

Graficzne programowanie mikrokontrolerów

M5Stack / UIFlow

Laboratorium 5

RFID i integracja systemu dostępu

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

Cel ćwiczenia

Celem ćwiczenia jest zaprojektowanie i uruchomienie systemu kontroli dostępu na platformie M5Stack. Student odczytuje UID kart RFID, tworzy listę uprawnień i logikę decyzji, sygnalizuje stan na matrycy RGB LED 8×8, steruje przekaźnikiem i rejestruje zdarzenia. Integruje wejścia, GUI, logikę procesu i wyjścia w modularnym programie z automatem stanów, diagnostyką, testami błędów oraz dokumentacją projektu.

1. Wprowadzenie

Wyposażenie: Core + Demo Board z czytnikiem MFRC522, matrycą RGB 8×8 i przekaźnikiem; trzy karty/tagi; sprawdzona konfiguracja sprzętu.

Organizacja pracy

Zapoznaj się z opisem stosowanych elementów, wykonaj cztery zadania częściowe, 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 częściowych i zadania końcowego. Opisane możliwości rozbudowy pozwalają pogłębić materiał. Wyniki testów i zdjęcia stanowiska zbierz podczas zajęć, a sprawozdanie opracuj po zajęciach.

Identyfikacja użytkowników i sygnalizacja

RFID i UID

Demo Board posiada czytnik RFID oparty na układzie MFRC522. W ćwiczeniu karta lub tag jest używany przede wszystkim jako nośnik identyfikatora UID. Program porównuje odczytany UID z listą identyfikatorów zapisanych w aplikacji.

Obsługę układu **RFID MFRC522** należy realizować za pomocą bloków dostępnych po podpięciu modułu **Unit RFID**

UID nie należy traktować jako bezwarunkowo bezpiecznego mechanizmu uwierzytelniania. Celem laboratorium jest nauka obsługi urządzenia, identyfikacji i logiki decyzji, a nie budowa certyfikowanego systemu bezpieczeństwa.

Matryca RGB LED 8×8 na Demo Board

Na płytce Demo Board znajduje się 64-elementowa matryca RGB LED. Diody są fizycznie rozmieszczone w układzie 8×8, ale elektrycznie tworzą jeden szeregowy łańcuch adresowalnych diod. Oznacza to, że mikrokontroler nie steruje osobnymi liniami wierszy i kolumn. Do całej matrycy doprowadzona jest jedna linia danych, a informacja o kolorze kolejnych diod jest przesyłana szeregowo przez cały łańcuch.

W przykładzie producenta matryca jest sterowana z linii **GPIO15**, zawiera 64 elementy, a sterowanie jest zgodne z układami klasy WS2811/NeoPixel. W UIFlow należy wykorzystywać funkcje z grupy **Hardware** → **RGB**, a matrycę traktować jako łańcuch 64 kolejno numerowanych diod.

Budowa logiczna matrycy

GPIO15 → LED 0 → LED 1 → LED 2 → ... → LED 63

Numer diody w łańcuchu nie jest tym samym co para współrzędnych (**wiersz**, **kolumna**). Przed tworzeniem znaków graficznych należy sprawdzić rzeczywistą kolejność diod na płytce przez kolejne zapalanie indeksów 0–63. Na tej podstawie można przygotować mapę przejścia z pozycji w macierzy 8×8 na numer diody w łańcuchu.

Zasada sterowania

Dla każdej diody określany jest kolor RGB. Po przygotowaniu wartości dla wybranych elementów należy uaktualnić stan matrycy. W ćwiczeniu nie należy używać maksymalnej jasności wszystkich 64 diod jednocześnie; do sygnalizacji stanu wystarcza niewielka lub średnia jasność.

Przykładowa reprezentacja wzorca 8×8 może być zapisana jako osiem wierszy wartości binarnych, gdzie 1 oznacza diodę świecącą, a 0 – wygaszoną. Kolor jest ustalany oddzielnie dla całego wzorca.

Przykład – czerwony znak X:

```
10000001
01000010
00100100
00011000
00011000
00100100
01000010
10000001
```

Logika dostępu

Typowy algorytm:

karta → UID → wyszukanie → decyzja → GUI/wyjście

Uwaga

Przełącznik w tym ćwiczeniu symuluje rygiel lub lampę sygnalizacyjną. Używaj wyłącznie bezpiecznego niskiego napięcia. Nie podłączaj napięcia sieciowego.

Architektura i integracja systemu

Zadaniem zespołu jest zintegrowanie czytnika, interfejsu i elementów wykonawczych 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 przełącznikami zespół musi przedstawić prowadzącemu schemat działania i sposób zatrzymania układu. Nie wolno pomijać ograniczeń bezpieczeństwa tylko po to, aby skrócić program.

Wskazówki do wariantu realizowanego na zajęciach

W podstawowym projekcie dwa znane identyfikatory i trzecia nieznana karta wystarczają do testu poziomów dostępu. Rozbudowana lista użytkowników powinna zawierać co najmniej trzy rekordy. Specyfikację wejść i wyjść, automat stanów, podział na funkcje oraz testy opracowuj dla tego samego systemu RFID.

Identyfikacja i decyzja

Czytnik MFRC522 odczytuje UID karty. W tym ćwiczeniu UID jest kluczem do rekordu użytkownika zawierającego nazwę i poziom dostępu. Sam odczyt identyfikatora nie załącza wyjścia; decyzja wynika z uprawnień i aktualnego stanu systemu.

Jest to demonstrator laboratoryjny logiki dostępu. Porównanie samego UID nie stanowi zabezpieczenia rzeczywistego zamka. Rolę elementu wykonawczego reprezentuje przełącznik z bezpiecznym obciążeniem stanowiska.

Przygotowanie i organizacja

Prowadzący sprawdza działanie Unit RFID, wyjścia przełącznikowego i matrycy. Z ćwiczeń 1–2 wykorzystaj funkcję GUI, obsługę zdarzeń i sterowanie wyjściem. Projekt integracyjny ma stały zakres: RFID + GUI + RGB + przełącznik. Dodatkowa komunikacja jest możliwością rozbudowy projektu.

UID i rekord użytkownika

UID zapisuj konsekwentnie: każdy bajt jako dwie cyfry szesnastkowe, stała kolejność bajtów i jednakowa wielkość liter. Nie zamieniaj braku odczytu na UID złożony z zer. Utwórz dwa uprawnione rekordy o poziomach 1 i 2, a trzecią kartę pozostaw nieznaną. Wymagany poziom funkcji wynosi 2.

Matryca RGB i przełącznik

Matryca Demo Board zawiera 64 adresowalne diody na GPIO15 i korzysta z obsługi Hardware → RGB. Potwierdź konfigurację dla stanowiska. Zapalając pojedyncze diody, rozpoznaj numerację; nie zakładaj liniowego przebiegu kolejnych wierszy. Używaj małej jasności.

Stan	Sygnalizacja	Wyjście
OCZEKIWANIE	Niebieski punkt / napis na LCD	Wyłączone
PRZYZNANY	Zielony znak potwierdzenia	Włączone przez 2 s
ODMOWA	Czerwony X / przyczyna	Wyłączone
BLAD	Czerwone miganie / komunikat	Wyłączone

Przełącznik jest sterowany z jednej funkcji wyjściowej. Po starcie, błędzie lub zatrzymaniu musi być wyłączony. Odmierzanie 2 s realizuj przez znacznik czasu, aby przyciski i logika błędu działały także podczas aktywnego wyjścia.

Jedno przyłożenie — jedno zdarzenie

Po obsłudze karty zablokuj kolejne decyzje do wykrycia braku karty przez co najmniej 300 ms. Trzymana karta nie może odnawiać czasu przełącznika. Zakończenie sygnalizacji i ponowne uzbrojenie czytnika to dwa oddzielne warunki.

Podział programu na funkcje

```
odczytaj_wejscia()
sprawdz_karte_i_uprawnienia()
aktualizuj_stan()
ustaw_wyjscia()
aktualizuj_gui()
```

Stanowisko demonstracyjne

Do styków przełącznika nie podłączaj napięcia sieciowego. Połączenia zmieniaj przy wyłączonym zasilaniu.

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.

Algorytm kontroli dostępu

```
karta → UID → wyszukanie rekordu → decyzja: dostęp przyznany / odmowa
dostępu → przełącznik + log
```

Zadanie 1. Odczyt i zapis identyfikatorów

Polecenie zadania

Uruchom Unit RFID zgodnie z konfiguracją stanowiska. Odczytaj UID trzech kart i nadaj im robocze nazwy. Wyświetl identyfikator w spójnym formacie. Po odsunięciu karty pokaż BRAK KARTY.

Warunek odbioru: Powtórne odczyty tej samej karty dają identyczny tekst. Pusty odczyt nie jest rozpoznawany jako użytkownik.

Sposób wykonania i przykłady

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

Odczyt karty

Odczytaj UID dostarczonej karty i wyświetl go na LCD w jednoznacznym formacie, np. jako kolejne bajty szesnastkowe. Sprawdź odczyt dla co najmniej dwóch identyfikatorów.

Sposób realizacji i weryfikacja

1. Zamień każdy bajt UID na dwucyfrowy zapis szesnastkowy i zachowaj stałą kolejność bajtów.
2. Sprawdź kilka kolejnych odczytów tej samej karty - UID powinien być identyczny; różnice oznaczają błąd formatowania lub odczytu.
3. Nie traktuj pustego/brakującego odczytu jako UID złożonego z zer.

Pseudokod / zapis algorytmu:

```
uid_tekst = " ".join(hex(bajt) for bajt in uid_bajty)
```

Zadanie 2. Uprawnienia i odmowa

Polecenie zadania

Utwórz rekordy UID/nazwa/poziom dla dwóch kart: poziomy 1 i 2. Dla wymaganego poziomu 2 wyznacz PRZYZNANY albo ODMOWA. Trzecia karta daje NIEZNANY i ODMOWA. Na ekranie wyświetl rozpoznaną nazwę i przyczynę decyzji. Wyjście pozostaje wyłączone.

Warunek odbioru: Karta poziomu 2 ma dostęp; poziomu 1 oraz nieznana otrzymują odmowę. Poprzednia nazwa nie pozostaje jako wynik nowej, nieznanej karty.

Dokumentacja projektu

Zapisz tabelę wejść i wyjść: źródło UID, przycisk zatrzymania, stan czytnika, LCD, RGB i przełącznik. Przy każdym wyjściu określ stan bezpieczny. Ta tabela jest początkiem dokumentacji projektu integracyjnego.

Sposób wykonania i przykłady

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

Lista uprawnionych kart

Utwórz w programie listę co najmniej trzech rekordów: UID, nazwa użytkownika i poziom dostępu. Po przyłożeniu karty wyświetl nazwę rozpoznanego użytkownika albo komunikat NIEZNANA KARTA.

Sposób realizacji i weryfikacja

1. Przechowuj rekord użytkownika jako zestaw powiązanych danych: UID, nazwa, poziom; unikaj kilku niezależnych list o tej samej długości.
2. Wyszukiwanie kończy się pierwszym dopasowaniem UID; jeśli brak dopasowania, zwróć stan NIEZNANY bez nazwy użytkownika.
3. Dodaj co najmniej dwa poziomy dostępu, aby później decyzja zależała nie tylko od samej obecności UID na liście.

Pseudokod / zapis algorytmu:

```
dla uzytkownik in uzytkownicy:
    if uid == uzytkownik.uid:
        zwroc uzytkownik
zwroc NIEZNANY
```

Decyzja: dostęp przyznany / odmowa dostępu

Dla karty spełniającej wymagany poziom dostępu wyświetl DOSTĘP PRZYZNANY, a dla karty nieznannej lub o niewystarczającym poziomie wyświetl ODMOWA DOSTĘPU. Wynik decyzji zapisz w zmiennej stanu. Na tym etapie komunikat jest prezentowany na LCD; w kolejnym zadaniu ten sam stan zostanie odwzorowany na matrycy RGB LED.

Sposób realizacji i weryfikacja

1. Zdefiniuj wymaganą klasę dostępu dla chronionej funkcji i porównuj ją z poziomem użytkownika.
2. Decyzję przechowuj jako zmienną stanu, a dopiero potem aktualizuj GUI i licznik statystyk.
3. Dla NIEZNANY zawsze wybieraj ODMOWA; nie pozostawiaj poprzedniego wyniku na ekranie jako aktualnego.

Pseudokod / zapis algorytmu:

```
if uzytkownik != NIEZNANY and uzytkownik.poziom >= wymagany_poziom:
    wynik = PRZYZNANY
else:
    wynik = ODMOWA
```

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/przekaźnik + 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.

Sprawdzenie: Prowadzący akceptuje zakres projektu przed rozpoczęciem integracji.

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.

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

Zadanie 3. Sygnalizacja i czas wyjścia**Polecenie zadania**

Przetestuj numerację LED, przygotuj niebieski punkt oczekiwania, zielone potwierdzenie i czerwony X. Po PRZYZNANY załącz przełącznik na 2 s; po ODMOWA tylko pokaż czerwony wzorec przez 2 s. Po upływie czasu wyłącz wyjście i wróć do niebieskiego wzorca.

Warunek odbioru: Wynik decyzji jest zgodny na LCD i RGB. Każda odmowa utrzymuje wyjście wyłączone; zielony stan nie trwa dłużej niż zaplanowany impuls.

Sposób wykonania i przykłady

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

Sygnalizacja stanu na matrycy RGB LED

Wykorzystaj matrycę 8×8 jako główny sygnalizator stanu terminala kontroli dostępu. Przygotuj co najmniej trzy jednoznacznie różne wzorce graficzne:

- **OCZEKIWANIE NA KARTE** – niebieski wzorec o małej jasności, np. ramka, punkt środkowy lub prosta animacja,
- **DOSTĘP PRZYZNANY** – zielony znak potwierdzenia,
- **ODMOWA DOSTĘPU** – czerwony znak X.

Sposób realizacji i weryfikacja

1. Skonfiguruj sterowanie matrycą jako łańcuchem 64 adresowalnych diod RGB na GPIO15. Ustaw ograniczoną jasność odpowiednią do pracy laboratoryjnej.
2. Wykonaj test numeracji: wygaszaj całą matrycę, a następnie zapalaj kolejno pojedyncze diody o indeksach 0–63. Zapisz kolejność fizycznego położenia diod.
3. Na podstawie testu przygotuj funkcję lub fragment programu, który dla wzorca 8×8 ustawia odpowiednie indeksy LED. Nie zakładaj, że indeksy kolejnych wierszy zawsze rosną od lewej do prawej.
4. Po uruchomieniu programu matryca pokazuje stan OCZEKIWANIE. Po zakończeniu identyfikacji karta powoduje przejście do zielonego lub czerwonego wzorca, zależnie od wyniku decyzji.
5. Po ustalonym czasie sygnalizacji wynikowej wyczyść matrycę i przywróć wzorec OCZEKIWANIE. Poprzedni wynik nie może pozostawać aktywny podczas następnej próby.

Pseudokod / zapis algorytmu:

```
ustaw_jasnosc(20)
pokaz_wzorzec(OCZEKIWANIE, NIEBIESKI)

if wynik == PRZYZNANY:
    pokaz_wzorzec(POTWIERDZENIE, ZIELONY)
else:
    pokaz_wzorzec(KRZYZ, CZERWONY)

po upływie czasu_sygnalizacji:
    pokaz_wzorzec(OCZEKIWANIE, NIEBIESKI)
```

Wskazówka projektowa

Dobłą praktyką jest oddzielenie danych wzorca od koloru. Ten sam mechanizm rysowania może wówczas wyświetlać różne symbole i barwy bez kopiowania kodu sterującego 64 diodami.

Sterowanie przekaźnikiem

Po prawidłowej autoryzacji załącz przekaźnik na krótki, ustalony czas, następnie automatycznie go wyłącz. Przekaźnik nie może pozostać aktywny po restarcie programu ani po nieznanej karcie.

Sposób realizacji i weryfikacja

1. Przekaźnik załączaj dopiero po decyzji PRZYZNANY; nie bezpośrednio w procedurze odczytu UID.
2. Zastosuj timer, po którym wyjście wraca do WYLACZONE bez konieczności kolejnego zdarzenia.
3. W procedurze inicjalizacji i w każdym stanie błędu wymuś WYLACZONE.

Pseudokod / zapis algorytmu:

```
if wynik == PRZYZNANY:
    wlacz_przekaznik()
    odblokuj_do = czas_biezacy + czas_otwarcia
if czas_biezacy >= odblokuj_do:
    wylacz_przekaznik()
```

Zadanie 4. Blokada ponownego odczytu i rejestr

Polecenie zadania

Wprowadź uzbrajanie odczytu dopiero po braku karty przez 300 ms. Dla jednego zdarzenia zapisz numer, UID, nazwę lub NIEZNANY, decyzję i czas od uruchomienia. Przechowuj tylko pięć ostatnich zdarzeń; zwiększaj liczniki PRZYZNANY i ODMOWA raz na zdarzenie.

Warunek odbioru: Trzymanie karty przez 5 s daje jeden wpis i jeden impuls 2 s. Po odsunięciu na co najmniej 300 ms ta sama karta może utworzyć kolejne zdarzenie.

Rejestr jest przechowywany w pamięci RAM i zeruje się po restarcie. Zapis do pliku ani pełna baza użytkowników nie należą do zakresu obowiązkowego.

Sposób wykonania i przykłady

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

Ochrona przed wielokrotnym odczytem

Jedno przyłożenie karty nie powinno tworzyć wielu zdarzeń. Wprowadź blokadę ponownej obsługi tego samego UID do chwili odsunięcia karty lub upływu ustalonego czasu.

Sposób realizacji i weryfikacja

1. Po obsłudze UID ustaw stan KARTA_OBECNA; kolejny odczyt tego samego UID nie tworzy nowego zdarzenia.
2. Powrót do OCZEKIWANIE_KARTY następuje po wykryciu braku karty przez określony czas lub po ustalonym przekroczeniu czasu oczekiwania.
3. Sprawdź długie przytrzymanie karty oraz szybkie odsunięcie i ponowne przyłożenie.

Pseudokod / zapis algorytmu:

```
OCZEKIWANIE_KARTY -> odczytaj UID -> PRZETWARZANIE -> KARTA_OBECNA
KARTA_OBECNA -> karta_usunieta -> OCZEKIWANIE_KARTY
```

Rejestr zdarzeń

Przechowuj licznik udanych i odrzuconych prób. Na ekranie dodaj stronę statystyk: ostatni UID, użytkownik, wynik, liczba PRZYZNANY i liczba ODMOWA.

Sposób realizacji i weryfikacja

1. Dla każdego zdarzenia zapisz co najmniej UID, rozpoznaną nazwę/NIEZNANY, wynik i numer kolejny lub czas od uruchomienia.
2. Liczniki PRZYZNANY i ODMOWA zwiększaj raz na obsłużone zdarzenie, po mechanizmie anti-powtórzeniowym.
3. Ekran statystyk nie może sterować przełącznikiem; jest wyłącznie prezentacją zgromadzonych danych.

Pseudokod / zapis algorytmu:

```
licznik_zdarzen += 1
if wynik == PRZYZNANY: przyznane += 1
else: odmowy += 1
```

Rezultat przykładu

System rozpoznaje uprawnione identyfikatory, odrzuca pozostałe, jednoznacznie sygnalizuje stan na matrycy RGB LED, steruje wyjściem tylko po poprawnej decyzji i nie zlicza wielokrotnie jednego przyłożenia karty.

Architektura projektu integracyjnego

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

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
```

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()
```

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.

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

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.

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

Rezultat przykładu

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

3. Zadanie końcowe

Polecenie zadania końcowego

Kompletny demonstrator kontroli dostępu

Połącz przygotowane funkcje w automat OCZEKIWANIE, PRYZNANY, ODMOWA i BLAD. Dodaj nadrzędną obsługę zatrzymania i błędu czytnika oraz wykonaj testy projektu integracyjnego.

- C w każdym stanie natychmiast wyłącza przełącznik i ustawia BLAD z komunikatem ZATRZYMANIE OPERATORA. B symuluje błąd odczytu i także wymusza BLAD.
- A kasuje BLAD wyłącznie wtedy, gdy czytnik działa i karta jest odsunięta; dla błędu symulowanego A usuwa flagę testową. Powrót następuje do OCZEKIWANIE, bez załączenia wyjścia.
- W BLAD odczyty kart nie powodują decyzji dostępu. Po resecie potrzebne jest nowe przyłożenie karty. Nie dopisuj zatrzymania operatora do licznika odmów kart.

- Na ekranie stale pokazuj stan, ostatnią decyzję, rzeczywiste polecenie wyjścia i oba liczniki. Student wskazuje przepływ wejście → decyzja → wyjście.

Testy i odbiór

Próba	Oczekiwana reakcja
Karta poziomu 2 / poziomu 1 / nieznana	Przyznanie / odmowa / odmowa.
Karta trzymana 5 s	Jeden wpis i impuls 2 s, bez przedłużania.
B lub C podczas impulsu	BLAD i natychmiastowe wyłączenie.
A przy karcie / po jej odsunięciu	Reset zablokowany / OCZEKIWANIE.

Kryteria zaliczenia

Poprawna identyfikacja i poziomy dostęp, RGB, czasowy przełącznik, blokada powtórzeń, rejestr pięciu zdarzeń, automat i dwa testy błędów.

Sprawozdanie

Zapisz projekt UIFlow, tabelę wejść/wyjść, diagram stanów, zdjęcia oraz tabelę testów: warunki, oczekiwany wynik, wynik rzeczywisty i sposób powrotu. Sprawozdanie PDF zredaguj po zajęciach.

Możliwość rozbudowy: przesyłanie rejestru do Core2 przez ESP-NOW z wykorzystaniem ćwiczenia 4.

Możliwości rozbudowy zadania końcowego

Zakres rozszerzeń

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

Rozbudowana kontrola dostępu z rejestrem zdarzeń

Wykorzystanie przygotowanych funkcji

Rozszerzenie projektu jest bezpośrednim rozwinięciem zadań 1–4. Odczyt UID z zadania 1 zasila listę użytkowników z zadania 2, decyzja dostępu z zadania 2 wybiera wzorzec matrycy RGB z zadania 3 i warunkuje sterowanie przełącznikiem z zadania 3. Automat z zadania 4 zapobiega wielokrotnemu zaliczaniu tej samej karty, a rejestr z zadania 4 gromadzi wyniki wszystkich prób.

Zbuduj terminal kontroli dostępu z co najmniej trzema użytkownikami. Każdy rekord zawiera UID, nazwę i poziom dostępu. Wybrany przełącznik może zostać aktywowany tylko dla osoby o odpowiednim poziomie. Na ekranie używaj polskich komunikatów, np. **DOSTĘP PRZYZNANY**, **ODMOWA DOSTĘPU**, **NIEZNANA KARTA**. Matryca RGB LED ma równolegle sygnalizować stan terminala: oczekiwanie na kartę, dostęp przyznany oraz odmowę dostępu za pomocą różnych wzorców i kolorów. Po zakończeniu sygnalizacji wyniku system automatycznie wraca do stanu oczekiwania. Dodaj licznik wszystkich prób, licznik prób zaakceptowanych i odrzuconych oraz ekran ostatniego zdarzenia.

Rozszerzenie: umożliw wyzerowanie statystyk wyłącznie po użyciu karty o poziomie administratora i dodatkowym potwierdzeniu przyciskiem.

Integracja dodatkowych funkcji systemu

Wykorzystanie przygotowanych funkcji

Projekt integracyjny jest końcową wersją rozwiązania rozwijanego w zadaniach 2 i 4 opisanych w tej instrukcji. Nie należy po specyfikacji rozpoczynać nowej implementacji: lista wejść i wyjść z zadania 2 ma zostać bezpośrednio wykorzystana w automacie z zadania 4, funkcje z zadania 4 realizują jego działanie, a przypadki z zadania 4 stają się obowiązkowymi testami odbiorowymi. Zadanie 4 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.

Dodatkowe testy przy rozbudowie zadania końcowego

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

- stabilny odczyt UID,
- lista co najmniej trzech rekordów,
- decyzja: dostęp przyznany / odmowa dostępu,
- poprawna konfiguracja i test numeracji 64 diod matrycy RGB,
- czytelna sygnalizacja stanów OCZEKIWANIE, DOSTĘP PRZYZNANY i ODMOWA DOSTĘPU na matrycy 8×8,
- automatyczny powrót matrycy do stanu oczekiwania,
- bezpieczne sterowanie przekaźnikiem,
- eliminacja wielokrotnego odczytu,
- rejestr i prezentacja statystyk.
- 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.

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,
- krótki opis zastosowanego algorytmu wraz ze schematem blokowym, diagramem stanów lub pseudokodem,
- opis sposobu testowania programu,
- wnioski.

Dokumentacja projektu obejmuje także opis celu, schemat blokowy, tabelę wejść i wyjść, diagram stanów, zrzuty programu UIFlow, zdjęcia stanowiska, opis testów normalnych i awaryjnych, wyniki oraz wnioski. Dołącz plik projektu.