

Graficzne programowanie mikrokontrolerów

M5Stack / UIFlow

Laboratorium 4

Komunikacja między urządzeniami

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

Celem ćwiczenia jest wymiana danych między M5Stack Core i Core2 przez UART, RS485 i ESP-NOW. Student konfiguruje transmisję, tworzy i interpretuje komunikaty tekstowe, odczytuje i zapisuje rejestry Modbus RTU oraz buduje dwukierunkową komunikację bezprzewodową. Przesyła wartości joysticka i polecenia sterujące, waliduje dane, numeruje ramki, wykrywa utratę łączności i realizuje przejście do stanu bezpiecznego.

1. Wprowadzenie

Wyposażenie: Core + Demo Board, Core2 z zewnętrznym interfejsem RS485/UART, przewody magistrali i projekt joysticka z ćwiczenia 2.

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.

Transmisja szeregową i protokoły przewodowe

UART

UART przesyła dane asynchronicznie. Obie strony muszą używać zgodnych parametrów, przede wszystkim prędkości transmisji, liczby bitów danych, parzystości i liczby bitów stopu. W najprostszym połączeniu wykorzystuje się linie TX i RX oraz wspólny potencjał odniesienia.

RS485

RS485 definiuje warstwę elektryczną wykorzystującą transmisję różnicową. W typowej konfiguracji dwuprzewodowej transmisja jest półdupleksowa: w danej chwili węzeł nadaje albo odbiera. Sama warstwa RS485 nie określa znaczenia komunikatów, dlatego w ćwiczeniu zostanie zbudowany prosty protokół aplikacyjny.

Przykładowe polecenia:

```
USTAW:SILNIK:50
POBIERZ:TEMPERATURA
```

POBIERZ: STAN
ZATRZYMAJ

Modbus RTU jako protokół nad RS485

RS485 określa warstwę elektryczną transmisji, natomiast **Modbus RTU** określa znaczenie przesyłanych bajtów. W typowym układzie jeden węzeł pełni rolę klienta/urządzenia nadrzędnego i inicjuje wymianę, a urządzenie serwer/urządzenie podrzędne odpowiada na żądania skierowane pod jego adres. W tym ćwiczeniu przyjmij:

Core + Demo Board: urządzenie nadrzędne Modbus (master/client) $\longleftrightarrow$ RS485
A/B $\longleftrightarrow$ Core2 + interfejs RS485: urządzenie podrzędne Modbus (slave/server)

Core2 należy połączyć z magistralą przez odpowiedni transceiver/Unit RS485 korzystający z UART. **Nie należy mechanicznie instalować Core2 na Demo Board.** Oznaczenia A/B potrafią różnić się pomiędzy producentami transceiverów, dlatego należy kierować się dokumentacją konkretnego modułu i potwierdzić komunikację testem.

Minimalna ramka Modbus RTU zawiera:

Pole	Znaczenie
Adres	adres urządzenia podrzędnego, 1 bajt
Kod funkcji	rodzaj operacji, np. 03 – odczyt rejestrów przechowujących (Holding Registers), 06 – zapis jednego rejestru przechowującego
Dane	adres rejestru, liczba rejestrów albo zapisywana wartość
CRC-16	kontrola poprawności ramki; w RTU bajt młodszy CRC jest transmitowany jako pierwszy

W podstawowej wersji ćwiczenia nie trzeba samodzielnie implementować obliczania CRC, jeżeli wykorzystywane bloki/klasz Modbus wykonują tę operację automatycznie.

Mapa rejestrów Core2 (adres urządzenie podrzędne = 2):

Offset	Nazwa	Zakres	Znaczenie
0	NASTAWA	0–100	nastawa zapisywana przez Core
1	TRYB	0–2	0=ZATRZYMANIE, 1=PRACA, 2=AUTOMAT
2	STAN	0–3	0=OK, 1=ALARM, 2=COMM_ERROR, 3=ZAJETY; logicznie tylko do odczytu
3	LICZNIK	0–65535	licznik aktualizowany lokalnie przez Core2

W ćwiczeniu używaj **offsetów 0,1,2,3**. Niektóre programy diagnostyczne prezentują rejestry przechowujące (Holding Registers) w notacji 40001, 40002 itd.

RS485 i terminacja

Na stanowisku laboratoryjnym stosuj topologię magistralową, krótkie odgałęzienia i zgodne parametry transmisji. Rezystory terminujące 120 Ω stosuje się na obu fizycznych końcach dłuższej magistrali; przed ich dodaniem sprawdź, czy moduł nie posiada już włączonej terminacji. Wszystkie połączenia wykonuj przy wyłączonym zasilaniu.

Uwaga

Nie łącz bezpośrednio sygnałów TTL UART z liniami A/B RS485. Do RS485 wykorzystuj dedykowany transceiver dostępny na Demo Board i właściwe zaciski magistrali.

Bezprzewodowa wymiana danych

Komunikacja peer-to-peer

ESP-NOW umożliwia bezpośrednią wymianę krótkich komunikatów pomiędzy urządzeniami ESP bez tworzenia typowego połączenia TCP/IP. W ćwiczeniu jeden M5Stack pełni rolę pilota, a drugi odbiornika. Do identyfikacji partnera wykorzystuje się adres MAC lub odpowiednią konfigurację peer.

Ramka aplikacyjna

Wartości wysyłane bezprzewodowo należy uporządkować w jednoznaczny format. Przykładowa ramka logiczna może zawierać:

```
nr_ramki, wychylenie_x, wychylenie_y, przycisk, nastawa, tryb
```

Pole `nr_ramki` jest numerem kolejnym ramki i ułatwia wykrywanie utraconych danych.

Bezpieczeństwo

Sterowanie napędem przez łącze bezprzewodowe wymaga stanu stan bezpieczny. Brak nowych ramek przez określony czas powinien prowadzić co najmniej do ustawienia polecenia ruchu na zero i sygnalizacji utraty komunikacji.

Wskazówki do wariantu realizowanego na zajęciach

Opis komunikacji zawiera pełną mapę Modbus z trybem AUTOMAT oraz pełne zadania obu protokołów. W planie obowiązkowym użyj uproszczonego wariantu z TRYB=0 lub 1. Pełny analizator czterech poleceń, tryb AUTOMAT, szczegółowe liczniki utraty ramek i porównanie dwóch kompletnych sterowników należą do rozwinięć.

Warstwy komunikacji

UART określa szeregową transmisję danych, RS485 jej różnicową warstwę elektryczną, a protokół aplikacyjny — znaczenie danych. Przy połączeniu półdupleksowym nadaje tylko jeden węzeł naraz. ESP-NOW pozwala przysyłać krótkie komunikaty bez typowej sesji TCP/IP.

W zadaniach 1–2 Core jest klientem, a Core2 serwerem. W zadaniach 3–4 Core przysyła dane joysticka, Core2 je prezentuje i odsyła polecenia. Logika walidacji oraz wykrywania braku danych jest potrzebna niezależnie od medium.

Przygotowanie przed zajęciami

Prowadzący przygotowuje sprawdzone projekty startowe obu urządzeń: konfigurację UART/RS485, działające bloki klienta i serwera Modbus oraz inicjalizację ESP-NOW. Studenci uzupełniają pola danych, walidację i logikę. CRC i sterowanie kierunkiem transceivera realizują gotowe biblioteki. Przygotowanie tych elementów pozwala skupić się na wymianie danych i logice programu.

RS485 i ramki tekstowe

Oba UART ustaw identycznie, np. 9600 bit/s, 8N1 zgodnie z konfiguracją stanowiska. Transceivery łącz liniami A/B według ich dokumentacji. Nie łącz A/B z TTL TX/RX i nie montuj Core2 na Demo Board. Połączenia wykonuj bez zasilania; terminację stosuj zgodnie ze schematem magistrali.

Ramkę tekstową kończy znak nowej linii. Zbieraj dane w buforze do końca linii, z limitem 64 bajtów i czasem kompletowania 500 ms. Nieznane polecenie, niepoprawna liczba albo przekroczony zakres nie mogą zmienić nastawy.

Modbus RTU

Ramka zawiera adres, funkcję, dane i CRC. Gotowa biblioteka obsługuje format oraz CRC. Core2 ma adres 2; FC03 odczytuje rejestry, FC06 zapisuje jeden rejestr. Używaj offsetów 0–3, nie numerów 40001–40004.

Offset	Nazwa	Znaczenie
0	NASTAWA	0–100; zapis i odczyt.
1	TRYB	0=ZATRZYMANIE, 1=PRACA.
2	STAN	0=OK, 1=ALARM, 2=BŁĄD ŁĄCZA; odczyt.
3	LICZNIK	0–65535, rośnie co sekundę; odczyt.

W podstawowym wariancie mapy rejestrów tryb AUTOMAT nie jest realizowany. Serwer odrzuca zapis poza zakresem; STAN i LICZNIK nie są zmieniane poleceniem operatora.

ESP-NOW i świeżość danych

Ustal partnerów po adresach MAC oraz zgodny kanał radiowy w projekcie startowym. Przyjmij tekstową ramkę DANE;nr;x;y;tryb oraz ramki POLECENIE;nr;akcja i STATUS;nr. Core2 wysyła STATUS co 200 ms jako sygnał obecności, również bez działań operatora. STATUS nie zmienia trybu pracy. Zakres x/y to -100...100, akcja to URUCHOM albo ZATRZYMAJ. Każda strona ma jeden rosnący licznik dla wszystkich swoich ramek, niezależnie od ich typu.

Waliduj nadawcę, typ, liczbę pól i zakresy. Dopiero nowa poprawna ramka odświeża czas ostatniego odbioru. Brak danych przez 2 s zeruje zdalne sterowanie i ustawia UTRATA ŁĄCZA. Duplikat nie jest nowym poleceniem.

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.

Warstwy komunikacji przewodowej

dane aplikacji → protokół (tekstowy / Modbus RTU) → UART →
nadajnik-odbiornik RS485 → A/B → drugi węzeł

Zadanie 1. Polecenia tekstowe przez RS485

Polecenie zadania

Uruchom tor UART przez RS485 w dostarczonym projekcie. Core wysyła USTAW:50 oraz POBIERZ:STAN, każdorazowo z końcem linii. Core2 waliduje argument USTAW w zakresie 0–100 i odsyła OK lub BLAD:powod. Odpowiada również na odczyt stanu. Core czeka na odpowiedź najwyżej 500 ms.

Warunek odbioru: Sprawdź 0, 100, 101 i tekst zamiast liczby. Błędny komunikat nie zmienia poprzedniej nastawy.

Sposób wykonania i przykłady

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

Konfiguracja UART

Skonfiguruj kanał komunikacji szeregowej z parametrami wskazanymi przez prowadzącego, np. 9600 lub 115200 bit/s, 8 bitów danych, bez parzystości, 1 bit stopu. Wysyłaj co sekundę licznik ramek.

Sposób realizacji i weryfikacja

1. Ustaw identyczne parametry po obu stronach: prędkość, 8 bitów danych, parzystość i bit stopu; zapisz konfigurację jako 8N1, jeżeli używasz braku parzystości i jednego bitu stopu.
2. Nadaj ramkę testową zawierającą numer sekwencyjny i znak końca linii; odbiornik powinien jednoznacznie rozdzielać kolejne komunikaty.
3. Porównaj zachowanie dla prawidłowej i celowo błędnej prędkości transmisji, aby rozpoznać typowe objawy niezgodnej konfiguracji.

Pseudokod / zapis algorytmu:

```
co 1 s:
    wyslij("RAMKA:" + nr_ramki + "\n")
    nr_ramki = nr_ramki + 1
```

Odbiór i odesłanie wiadomości

Program powinien odbierać linię tekstu i odesłać ją z prefiksem POWTORZ:. Dodaj licznik poprawnie odebranych komunikatów.

Sposób realizacji i weryfikacja

1. Odbieraj dane do bufora aż do znaku końca linii albo do osiągnięcia ustalonej długości maksymalnej.
2. Po złożeniu pełnej wiadomości usuń znak końca linii, zwiększ licznik RX i odeślij POWTORZ:....
3. Dodaj przekroczenie czasu oczekiwania bufora: niekompletna wiadomość nie może blokować programu bez końca.

Pseudokod / zapis algorytmu:

```
dopoki dane_dostepne:
    znak = odczytaj_znak()
    if znak == KONIEC_LINII:
        przetworz(bufor); bufor = ""
    else:
        bufor += znak
```

Analizator poleceń

Zaimplementuj rozpoznawanie poleceń POBIERZ:STAN, POBIERZ:WARTOSC, USTAW:WARTOSC:n oraz ZATRZYMANIE. Dla nieznanego polecenia zwróć BŁĄD:POLECENIE.

Sposób realizacji i weryfikacja

1. Najpierw rozpoznaj nazwę polecenia, a dopiero potem liczbę argumentów i ich typy.
2. Każde żądanie kończy się dokładnie jedną odpowiedzią: OK, danymi albo BLAD:<powod>.
3. Nieznane lub niepełne polecenie nie może zmieniać stanu urządzenia.

Pseudokod / zapis algorytmu:

```
pola = podziel(wiadomosc, ":")
if pola[0] == "POBIERZ": ...
elif pola[0] == "USTAW": ...
elif wiadomosc == "ZATRZYMAJ": ...
else: odpowiedz("BLAD:POLECENIE")
```

Walidacja argumentu

Dla USTAW:WARTOSC:n sprawdź, czy argument jest liczbą i należy do zakresu 0–100. W przypadku błędu zwróć odpowiedni komunikat bez zmiany poprzedniej wartości.

Sposób realizacji i weryfikacja

1. Konwertuj argument dopiero po sprawdzeniu, że polecenie ma właściwą liczbę pól.
2. Po konwersji sprawdź zakres 0–100; dla wartości spoza zakresu zwróć błąd bez zmiany poprzedniej nastawy.
3. Przetestuj: 0, 100, -1, 101, pusty argument oraz tekst zamiast liczby.

Pseudokod / zapis algorytmu:

```
nowa_wartosc = odczytaj_liczbe(pole)
if blad_konwersji: BLAD:TYP
elif nowa_wartosc < 0 or nowa_wartosc > 100: BLAD:ZAKRES
else: wartosc = nowa_wartosc; OK
```

RS485

Uruchom wymianę danych przez tor RS485. Jeden węzeł pełni rolę nadrzędną i wysyła żądanie, drugi odpowiada. Jeżeli dostępny jest tylko jeden węzeł RS485, prowadzący może zapewnić drugi terminal lub konwerter.

Sposób realizacji i weryfikacja

1. Połącz dwa węzły przewodami A/B zgodnie z oznaczeniami zastosowanych transceiverów; dla Core2 użyj zewnętrznego interfejsu RS485/UART, nie montuj Core2 mechanicznie na Demo Board.
2. Uruchom najpierw prostą wymianę żądanie–odpowiedź bez Modbus: Core wysyła żądanie, Core2 odpowiada. Sprawdź, czy tylko jeden węzeł nadaje w danej chwili.
3. Dodaj licznik zapytań i odpowiedzi; brak odpowiedzi w ustalonym czasie traktuj jako błąd komunikacji.

Pseudokod / zapis algorytmu:

```
CORE: wyslij(zadanie); czekaj_na_odpowiedz(limit_czasu)
CORE2: if odebrano_zadanie: przetworz(); wyslij(odpowiedz)
```

Zadanie 2. Rejestry Modbus RTU

Polecenie zadania

Zapisz wersję tekstową, a następnie uruchom wariant Modbus na tym samym torze. Z Core odczytaj cztery rejestry FC03, zapisz NASTAWA=40 przez FC06 i odczytaj ponownie. W Core2 uruchom lokalne zwiększanie LICZNIK. Dodaj licznik poprawnych odpowiedzi i przekroczeń czasu.

Warunek odbioru: Odczyt potwierdza 40 i zmienny LICZNIK. Zapis 101 jest odrzucony. Błędny adres urządzenia powoduje brak odpowiedzi, ale GUI reaguje.

Kolejność pracy

Protokół tekstowy i Modbus uruchamiaj oddzielnie. Nie pozostawiaj dwóch parserów obsługujących ten sam UART. Nie implementuj ręcznie CRC w czasie tego ćwiczenia.

Sposób wykonania i przykłady

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

Mapa rejestrów Modbus RTU

Skonfiguruj Core2 jako urządzenie podrzędne Modbus (slave/server) o adresie 2 i odwzoruj cztery rejestry z tabeli we wstępie. Na ekranie Core2 pokaż bieżące wartości NASTAWA, TRYB, STAN i LICZNIK. Rejestr LICZNIK zwiększaj lokalnie raz na sekundę, aby urządzenie nadrzędne mogło odczytywać zmieniającą się wartość.

Sposób realizacji i weryfikacja

1. Uruchom warstwę RS485 z identycznymi parametrami po obu stronach, np. 9600 bit/s, 8N1.
2. Zainicjalizuj urządzenie podrzędne Modbus (slave/server) pod adresem 2 i przypisz zmienne programu do offsetów 0–3.
3. Ustaw STAN=0 i TRYB=0 po starcie. Program powinien rozpocząć pracę w stanie bezpiecznym ZATRZYMANIE.
4. Zmień lokalnie LICZNIK; potwierdź na ekranie Core2, że zmienna rzeczywiście się zmienia przed testem odczytu z urządzenia nadrzędnego.

Odczyt i zapis rejestrów Core–Core2

Skonfiguruj Core jako urządzenie nadrzędne Modbus (master/client). Wykonaj cykliczny odczyt rejestrów Core2 kodem funkcji 03 oraz zapis NASTAWA lub TRYB kodem funkcji 06. Po każdej operacji wyświetl wynik i zwiększ odpowiedni licznik komunikacji.

Sekwencja testowa:

1. Nadrzędny: odczytaj rejestry 0..3 z urządzenia podrzędnego 2
2. Podrzędny: zwróć NASTAWA, TRYB, STAN, LICZNIK
3. Nadrzędny: zapisz rejestr 0 = 50
4. Nadrzędny: odczytaj rejestr 0 i sprawdź wartość == 50
5. Nadrzędny: zapisz rejestr 1 = 1 (PRACA)
6. Podrzędny: sprawdź polecenie i aktualizuj GUI

Testy obowiązkowe

Wykonaj: poprawny odczyt, poprawny zapis, odczyt z błędnego adresu urządzenia podrzędne, przerwę w przewodzie/wyłączenie drugiego węzła oraz próbę zapisu niedozwolonej wartości, np. NASTAWA=150. Niedozwolona wartość nie może zostać użyta przez logikę sterowania nawet wtedy, gdy warstwa protokołu dostarczy ją poprawnie.

Przekroczenie czasu oczekiwania i diagnostyka

Dodaj licznik czasu od ostatniej poprawnej ramki. Jeżeli przez ustalony czas nie pojawi się komunikat, ustaw stan BŁĄD KOMUNIKACJI. Na ekranie pokazuj liczbę odebranych ramek, liczbę błędów i czas od ostatniej ramki.

Sposób realizacji i weryfikacja

1. Mierz czas od wysłania żądania do poprawnej odpowiedzi oraz czas od ostatniej poprawnej ramki.
2. Rozróżnij co najmniej: przekroczenie czasu oczekiwania, błędny adres/funkcję, niepoprawną wartość oraz błąd CRC Modbus.
3. Po utracie komunikacji przejdź do bezpiecznego stanu i wymagaj ponownego URUCHOM po odzyskaniu łącza.

Pseudokod / zapis algorytmu:

```
if czas_biezacy - ostatnia_poprawna > limit_czasu_komunikacji:
    stan_komunikacji = BŁĄD_KOMUNIKACJI
    bezpieczne_zatrzymanie()
```

Rezultat przykładu

Dwa węzły wymieniają polecenia i odpowiedzi według ustalonego formatu, błędne polecenia nie zmieniają stanu urządzenia, a utrata komunikacji jest wykrywana.

Przepływ danych ESP-NOW

Core: joystick/enkoder → ramka + nr_ramki → ESP-NOW → Core2: GUI Core2:
stan → ESP-NOW → Core

Zadanie 3. ESP-NOW i ramka joysticka**Polecenie zadania**

Przejdź do projektu ESP-NOW. Zapisz MAC urządzeń i ustaw właściwych partnerów. Core wysyła co 200 ms DANE z numerem, skalibrowanym X/Y i trybem. Core2 sprawdza pola i wyświetla X/Y oraz numer ramki. Wykorzystaj pomiar joysticka z ćwiczenia 2.

Warunek odbioru: Oba kierunki osi są poprawne, wartości mieszczą się w -100...100. Niekompletna ramka oraz X=150 są odrzucane.

Sposób wykonania i przykłady

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

Identyfikacja urządzeń

Odczytaj i zapisz adresy urządzeń Core i Core2. Skonfiguruj je tak, aby mogły wymieniać dane przez ESP-NOW. Ustal role: nadajnik i odbiornik.

Sposób realizacji i weryfikacja

1. Odczytaj adres MAC obu urządzeń i zapisz, który będzie używany jako peer po drugiej stronie.
2. Nadaj urządzeniom role logiczne CORE_PILOT i CORE2_PANEL; rola nie musi oznaczać wyłącznie nadawania lub odbioru.
3. Przed wysyłaniem danych wyświetl na obu ekranach stan inicjalizacji peerów.

Sprawdzenie: Oba urządzenia identyfikują właściwego partnera i nie zależą od przypadkowego kolejnego uruchomienia.

Pierwsza wiadomość

Nadajnik wysyła co określony czas licznik ramek. Odbiornik wyświetla numer ostatniej ramki oraz liczbę odebranych komunikatów.

Sposób realizacji i weryfikacja

1. Ramka powinna zawierać co najmniej **nr_ramki** i pole typu wiadomości.
2. Nadaj w stałym, niezbyt krótkim okresie, np. 100–500 ms; zbyt szybkie nadawanie utrudnia analizę błędów.
3. Odbiornik zapisuje numer ostatniej ramki i czas jej odebrania, a GUI aktualizuje tylko po nowej wiadomości.

Pseudokod / zapis algorytmu:

```
ramka = {typ:"STAN", nr_ramki:nr_ramki}
wyslij(ramka)
nr_ramki = (nr_ramki + 1) mod 65536
```

Przesyłanie joysticka

Rozszerz ramkę o wartości X i Y joysticka Demo Board. Core2 powinien prezentować je liczbowo i za pomocą kursora na ekranie.

Sposób realizacji i weryfikacja

1. Przed wysłaniem zastosuj kalibrację, martwą strefę i ograniczenie wartości X/Y; wysyłaj dane użytkowe, nie surowe ADC, chyba że są potrzebne diagnostycznie.
2. Po stronie Core2 zweryfikuj obecność wymaganych pól i zakres -100...100 przed użyciem danych.
3. Kursor aktualizuj wyłącznie z ostatniej poprawnej ramki.

Pseudokod / zapis algorytmu:

```
wysylanie: {nr_ramki, x, y, przycisk}
odbior: if -100<=x<=100 and -100<=y<=100: aktualizuj_kursor()
```

Zadanie 4. Polecenie zwrotne i utrata łącza

Polecenie zadania

Na Core2 dodaj dotykowe URUCHOM i ZATRZYMAJ. Wyślij POLECENIE do Core; Core odsyła rzeczywisty tryb w kolejnej DANE. Dodaj cykliczne STATUS z Core2. Core sprawdza świeżość STATUS/POLECENIE, a Core2 świeżość DANE. Wprowadź kontrolę numeru ramki i wykrywanie braku nowych poprawnych ramek przez 2 s na obu urządzeniach. Po utracie łącza Core ustawia ZATRZYMANIE.

Warunek odbioru: Panel rozróżnia żądanie i potwierdzony tryb. Duplikat nie wywołuje drugiej akcji. Wstrzymanie nadawania wywołuje UTRATA ŁĄCZA.

W podstawowym wariancie użyj rosnącego licznika przez czas sesji, bez zawijania. Po restarcie jednego węzła ręcznie rozpocznij nową sesję na obu urządzeniach przy zatrzymanym sterowaniu i wyzeruj zapamiętane numery. Automatyczna obsługa restartów jest rozszerzeniem.

Sposób wykonania i przykłady

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

Sterowanie zwrotne

Dodaj komunikat wysyłany z Core2 do Core. Może to być polecenie zmiany trybu na RĘCZNY albo AUTOMATYCZNY, ewentualnie zmiana nastawy. W ten sposób oba urządzenia pełnią rolę nadajnika i odbiornika.

Sposób realizacji i weryfikacja

1. Zdefiniuj osobny typ ramki dla polecenia zwrotnego, np. POLECENIE; nie mieszaj znaczenia tych samych pól zależnie od kierunku transmisji.
2. Core po odebraniu żądania zmiany trybu sprawdza, czy przejście jest dozwolone, i odsyła stan potwierdzający stan rzeczywisty.
3. Na ekranach rozróżniaj wartość zadaną przez operatora i stan potwierdzony przez drugi węzeł.

Pseudokod / zapis algorytmu:

```
Core2 -> {typ:POLECENIE, zadany_tryb:AUTOMAT}
Core  -> {typ:STAN, rzeczywisty_tryb:AUTOMAT}
```

Numeracja i utrata ramek

Wykorzystaj pole `nr_ramki`. Odbiornik powinien wykrywać przeskok numeru ramki i zwiększać licznik potencjalnie utraconych komunikatów.

Sposób realizacji i weryfikacja

1. Przechowuj `ostatni_nr_ramki`; dla kolejnej ramki oblicz oczekiwaną wartość z uwzględnieniem przepełnienia licznika.
2. Jeżeli numer przeskoczył o więcej niż 1, zwiększ licznik utraconych ramek o liczbę brakujących numerów.
3. Nie traktuj duplikatu jako nowej ramki sterującej; zlicz go osobno lub pomiń.

Pseudokod / zapis algorytmu:

```
oczekiwany_nr = (ostatni_nr_ramki + 1) mod 65536
if nr_ramki != oczekiwany_nr:
    utraczone += roznica_nr(oczekiwany_nr, nr_ramki)
```

Sygnal okresowy i stan bezpieczny

Jeżeli przez 2 s nie zostanie odebrana poprawna ramka, ustaw stan UTRATA ŁĄCZA. W tym stanie wyzeruj polecenie sterujące napędem, a na ekranie pokaż alarm. Po wznowieniu komunikacji przejście do sterowania powinno wymagać świadomego polecenia URUCHOM.

Sposób realizacji i weryfikacja

1. Aktualizuj czas_ostatniego_odbioru tylko po pełnej, poprawnie zweryfikowanej wiadomości.
2. Po przekroczeniu 2 s ustaw UTRATA ŁĄCZA i natychmiast wyzeruj polecenie ruchu/wyjścia zależne od zdalnego sterowania.
3. Po powrocie ramek przejdź do stanu połączony, ale zachowaj ZATRZYMANIE do czasu nowego świadomego polecenia URUCHOM.

Pseudokod / zapis algorytmu:

```
if czas_biezacy - czas_ostatniego_odbioru > 2000 ms:
    lacze = UTRATA_LACZA
    polecenie = 0
    wymagaj_ponownego_uruchomienia = True
```

Rezultat przykładu

Core z Demo Board działa jako pilot, Core2 jako zdalny panel, komunikacja jest dwukierunkowa, a utrata ramek i całkowite zerwanie łączności są sygnalizowane.

3. Zadanie końcowe

Polecenie zadania końcowego

Bezprzewodowy panel manipulatora

Rozbuduj projekt ESP-NOW z zadań 3–4. Core2 pokazuje kursor sterowany odebranymi X/Y wyłącznie po potwierdzeniu PRACA przez Core. Core ma lokalny przycisk C o najwyższym priorytecie ZATRZYMAJ. Nie steruj napędem fizycznym.

- Zachowaj walidację ramek, numery i diagnostykę. Wyświetl stan łącza, potwierdzony tryb i wiek ostatniej poprawnej ramki.
- Utrata danych zatrzymuje kursor i zeruje używane wartości X/Y. Powrót ramek przywraca informację POŁĄCZONY, lecz nie uruchamia ruchu.
- URUCHOM akceptuj po aktywnym łączu i powrocie joysticka do środka. Polecenie oczekujące na potwierdzenie nie może samo uruchamiać kursora.
- Zachowaj osobno działające projekty RS485/tekst i Modbus. Nie łącz wszystkich trzech protokołów w jedną aplikację końcową.

Testy i odbiór

Próba	Oczekiwana reakcja
Poprawne dane i URUCHOM	Potwierdzony PRACA; kursor reaguje.
Ramka z X=150 lub duplikat	Brak aktualizacji sterowania i czasu świeżości.
Wstrzymanie transmisji na ponad 2 s	UTRATA ŁĄCZA, zatrzymanie kursora.
Powrót transmisji	POŁĄCZONY, nadal ZATRZYMANIE.

Kryteria zaliczenia

Poprawny test tekstowy i FC03/FC06, dwukierunkowa wymiana ESP-NOW, walidacja, potwierdzanie stanu i zatrzymanie po utracie łącza.

Sprawozdanie

Dołącz projekty obu węzłów, parametry UART, mapę rejestrów, format ramek, diagram przepływu danych i wyniki testów. PDF przygotuj po zajęciach na podstawie zapisów z demonstracji.

Możliwość rozbudowy: licznik utraconych ramek z obsługą zawijania lub automatyczna identyfikacja nowej sesji po restarcie.

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.

Sterowanie przez protokół tekstowy i Modbus RTU

Wykorzystanie przygotowanych funkcji

Rozbuduj tor komunikacyjny przygotowany w zadaniach 1 i 2. Wykorzystaj konfigurację UART i RS485, buforowanie, analizator poleceń oraz walidację z zadania 1. Dodaj obsługę rejestrów Modbus RTU i diagnostykę z zadania 2. Brak poprawnej odpowiedzi powinien prowadzić do stanu bezpiecznego.

Uruchom dwa warianty sterowania tym samym stanowiskiem po RS485. W pierwszym użyj polskiego protokołu tekstowego, np. **POBIERZ:STAN**, **POBIERZ:WARTOSC**, **USTAW:WARTOSC:n**, **ZATRZYMAJ**. W drugim użyj Modbus RTU pomiędzy Core jako urządzeniem nadrzędnym i Core2 jako urządzeniem podrzędnym. Rejestry nazwij **NASTAWA**, **TRYB**, **STAN** i **LICZNIK**. Core ma cyklicznie odczytywać rejestry, zapisywać nastawę 0–100 i tryb pracy oraz wykrywać przekroczenie czasu oczekiwania. Core2 waliduje wartości i po utracie komunikacji przechodzi do **ZATRZYMANIE** oraz wyświetla **BŁĄD KOMUNIKACJI**.

Rozszerzenie: licz oddzielnie poprawne zapytania, poprawne odpowiedzi i błędy oraz pokaż czas od ostatniej poprawnej wymiany.

Bezprzewodowe sterowanie ze stanem bezpiecznym

Wykorzystanie przygotowanych funkcji

Połącz konfigurację urządzeń i przesyłanie joysticka z zadania 3 z komunikacją zwrotną, numeracją ramek i kontrolą łącza z zadania 4. Na tej podstawie rozbuduj pilot i odbiornik, zachowując zatrzymanie sterowania po utracie transmisji.

Opracuj bezprzewodowy pilot. Joystick steruje dwoma parametrami na Core2, przycisk joysticka zmienia tryb, a enkoder ustala wartość zadaną. Core2 odsyła do pilota swój rzeczywisty stan. Oba ekrany prezentują **STAN ŁĄCZA**, numer ramki oraz licznik utraconych lub błędnych ramek. Po utracie transmisji sterowanie ma zostać wyzerowane, a ponowne uruchomienie wymaga świadomego polecenia użytkownika.

Rozszerzenie: pokaż różnicę pomiędzy trybem zadanym przez pilota i trybem rzeczywiście przyjętym przez Core2.

Dodatkowe testy przy rozbudowie zadania końcowego

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

- poprawna konfiguracja transmisji,
- nadawanie i odbiór danych tekstowych,
- analizator poleceń co najmniej czterech poleceń,
- walidacja argumentów,
- uruchomienie toru RS485,
- poprawna konfiguracja Modbus RTU Core–Core2 i mapa rejestrów,
- odczyt FC03 oraz zapis co najmniej jednego rejestru (np. FC06),
- diagnostyka i przekroczenie czasu oczekiwania komunikacji.
- poprawna konfiguracja dwóch urządzeń,
- przesyłanie ramki z kilkoma polami,
- prezentacja danych odebranych,
- komunikacja dwukierunkowa,
- numeracja i diagnostyka ramek,
- wykrywanie utraty łączności i stan bezpieczny.

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.