

Graficzne programowanie mikrokontrolerów

M5Stack / UIFlow

Laboratorium 8

ESP-NOW – komunikacja bezprzewodowa

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

Celem ćwiczenia jest uruchomienie bezpośredniej komunikacji bezprzewodowej ESP-NOW pomiędzy M5Stack Core i Core2. Student poznaje adresowanie peer-to-peer, buduje ramkę danych, przesyła wartości joysticka i polecenia sterujące oraz implementuje mechanizm wykrywania utraty łączności i przejście do stanu bezpiecznego.

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.

Przebieg ćwiczenia

Przepływ danych ESP-NOW

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

Zadanie 1. 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 wysłaniem danych wyświetl na obu ekranach stan inicjalizacji peerów.

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

Zadanie 2. 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
```

Zadanie 3. 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. 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}
```

Zadanie 5. 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 pomini.

Pseudokod / zapis algorytmu:

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

Zadanie 6. Sygnał 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
```

Wymagany rezultat

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

Zadanie końcowe**Integracja zadań etapowych**

Zadanie końcowe powstaje przez połączenie rezultatów zadań 1–6. Identyfikacja urządzeń z zadania 1 i pierwsza wiadomość z zadania 2 pozostają podstawą połączenia, dane joysticka z zadania 3 tworzą ramkę sterującą, zadanie 4 dodaje odpowiedź zwrotną, zadanie 5 numeruje ramki i wykrywa braki, a zadanie 6 zapewnia okresową kontrolę łącza i przejście do stanu bezpiecznego.

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.

Kryteria zaliczenia

- poprawna konfiguracja dwóch urządzeń,
- przesyłanie ramki z kilkoma polami,
- prezentacja danych odebranych,
- komunikacja dwukierunkowa,
- numeracja i diagnostyka ramek,
- wykrywanie utraty łączności i stan bezpieczny.

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.