Graficzne Programowanie Mikrokontrolerów

Ćwiczenie 5

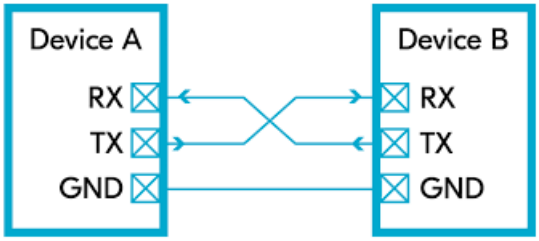
Przewodowe i bezprzewodowe przesyłanie danych
RS485 i EspNow

Komunikacja mikrokontrolera z otoczeniem (czujniki, urządzenia i inne mikrokontrolery) realizowana jest za pośrednictwem portów mikrokontrolera i protokołów komunikacyjnych. W niniejszym ćwiczeniu przećwiczona zostania komunikacja szeregowa **UART** i **RS485** oraz dostępna w mikrokontrolerach **M5STACK** (rodzina *M5Core* i *M5Stick*) łączność bezprzewodowa krótkiego zasięgu **EspNow**. Obydwie zapewniają nie tylko wymianę informacji pomiędzy mikrokontrolerami, ale jak w przypadku protokołów komunikacji szeregowej także z czujnikami i innymi urządzeniami jak mierniki, rejestratory, itp.

1 Komunikacja szeregowa UART

UART (Universal Asynchronous Receiver-Transmitter) to podstawowy protokół komunikacji szeregowej, używany do przesyłania danych między urządzeniami. Transmisja nie wymaga synchronizacji zegarowej, do jej poprawnej realizacji konieczne jest ustawienie przez urządzenia zgodności w:

- prędkości transmisji (boud rate) - liczbie bitów wysyłanych w ciągu 1 s, najczęściej *9600* lub *115200*
- strukturze ramki danych definiowanej przez cztery parametry:
  - **bit startu**: Oznacza początek transmisji, sygnał przechodzi z wysokiego stanu logicznego (1) na niski (0).
  - **bity danych**: Zazwyczaj 7 lub 8 bitów danych, ale mogą być inne konfiguracje (5 do 9 bitów).
  - **bit parzystości** (opcjonalnie): Używany do wykrywania błędów transmisji. Może być parzysty lub nieparzysty.
  - **bity stopu**: Informują o końcu transmisji, zwykle 1 lub 2 bity stopu.



Rys. 1: Połączenie dwóch urządzeń do transmisji UART

Transmisja realizowana jest dwukierunkowo za pomocą dwóch linii danych, **Tx** - wysyłania i **Rx** - odbierania. Wskazane jest dodatkowo stosowanie trzeciej linii GND w celu zapewnienia wspólnego punktu odniesienia dla odbiornika i nadajnika, redukuje także ewentualne zakłócenia w transmisji. Porty **Tx** i **Rx** jednego urządzenia należy w połączeniu krosowym, jak pokazano na Rys. 1, z portami **Rx** i **Tx** drugiego.

UART znajduje szerokie zastosowanie:

- programowaniu i debugowaniu mikrokontrolerów
- komunikacji z innymi urządzeniami - moduły GPS, bluetooth, modemy,...

Istotnym ograniczeniem metody transmisyjnej UART jest dopuszczalna długość linii łączącej urządzenia. Im dłuższe są przewody tym wpływ zakłóceń i tłumienia sygnału będzie wzrastał. W efekcie odbierane dane mogą ulec zniekształceniu lub nawet utracie. Istotnym czynnikiem wpływającym na określenie bezpiecznej długości linii transmisyjnej ma prędkość transmisji. Im większa tym krótsza może być linia. Przy prędkości *9600*bps długość linii może dochodzić do około 15 m, a przy *115200*bps do około 2 m. Poprawa parametrów linii transmisyjnej poprzez:

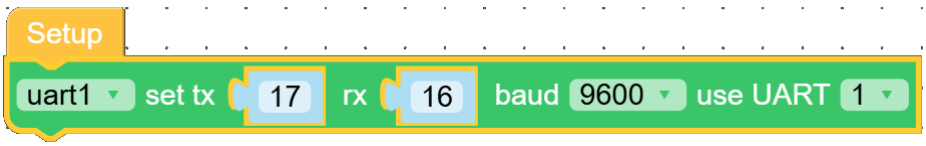
- ekranowanie linii,
- stosowanie wzmacniaczy sygnału lub repeaterów,
- zastosowanie z standardów transmisji opartych na UART jak **RS232** czy **RS485** które pozwalają na wydłużenie zasięgu nawet do około **1**km

może znacząco wpłynąć na zwiększenie długości linii transmisyjnej i poprawę jakości sygnału.

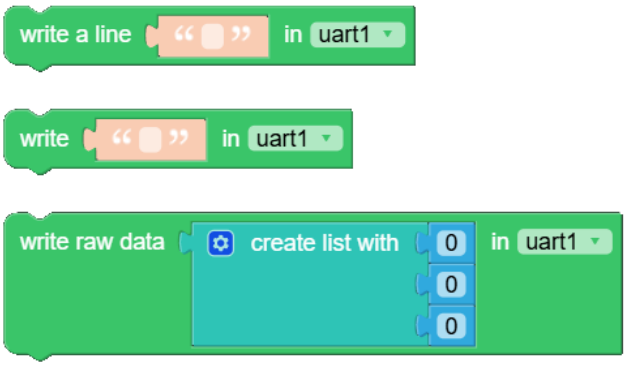
1.1 UART w UIFlow

Obsługa łączności szeregowej realizowana jest w środowisku UIFlow za pomocą bloków zgrupowanych w **UART** dostępnych w zakładce **HARDWARE**. W większości mikrokontrolery M5Stack pozwalają na zdefiniowanie dwóch kanałów komunikacyjnych. Dostępne bloki podzielono na trzy grupy:

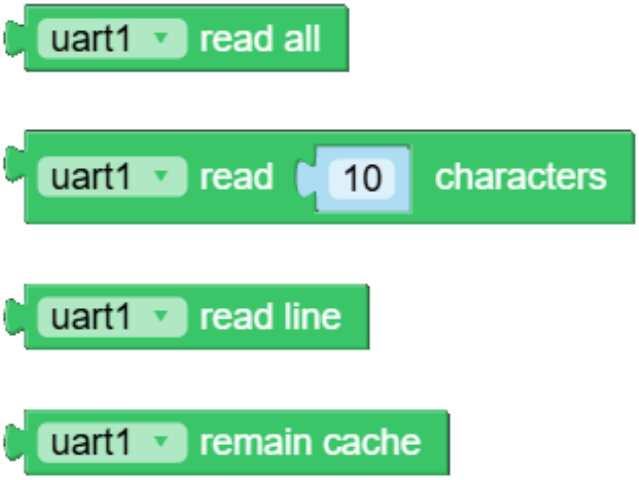
- *blok inicjujący* (Rys.2) definiujący kanał UART o zadanej prędkości transmisji korzystający wskazanych portów mikrokontrolera



Rys. 2: Blok inicjacji połączenia



Rys. 3: Bloki wysyłania danych



Rys. 4: Bloki odbioru danych

- *bloki wysyłania* (Rys.3), można podzielić na dwie grupy, wysyłania ciągu tekstowego (dwa bloki, w jednym ciąg zakończony jest znakiem końca linii) i wysyłania ciągu surowych danych w postaci listy bajtów.
- *bloki odbierania* (Rys.4), wydzielić można dwie grupy bloków, pierwsza odczytuje dane z portu szeregowego:
  1. odczytanie wszystkich danych z portu szeregowego i przekazanie ich w postaci ciągu znaków,
  2. odczytanie określonej liczby znaków z portu szeregowego,
  3. czytanie z portu do czasu natrafienia na znak końca linii,

  , druga grupę stanowi blok testowy zwracający wartość *true* gdy na porcie pojawią się dane.

Prawidłowa komunikacja zostanie zawiązana gdy na obu urządzeniach realizowana będzie transmisja o tej samej prędkości. Mikrokontroler M5Core posiada wyprowadzone na piny GPIO dwa porty UART, pierwszy korzysta z pinów GPIO**1** (**Tx**) i GPIO**3** (**Rx**), a drugi z pinów GPIO**17** (**Tx**) i GPIO**16** (**Rx**).

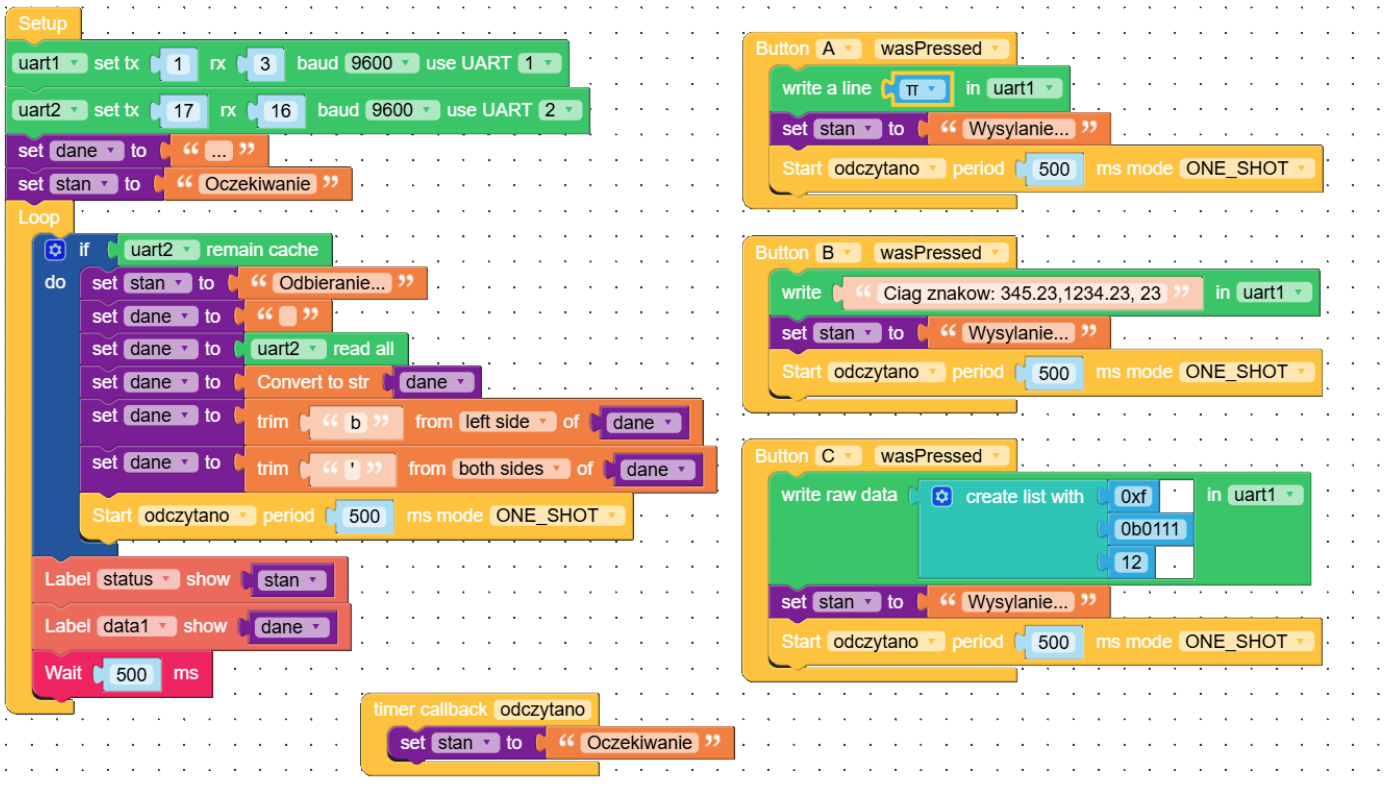
W przykładzie demonstrującym wykorzystanie komunikacji szeregowej UART, każdy z przycisków mikrokontrolera wysyłał będzie inny zestaw danych na kanale *uart1*:

A pojedynczą linie znaków,

B ciąg znaków,

C zestaw surowych danych,

odbierał je na kanale *uart2* i wyświetlał na ekranie. Kod realizujący podany problem pokazano poniżej na Rys.5, a efekt działania (po naciśnięciu przycisku **A**) pokazano na Rys.6.



Rys. 5: Kod przykładowego programu

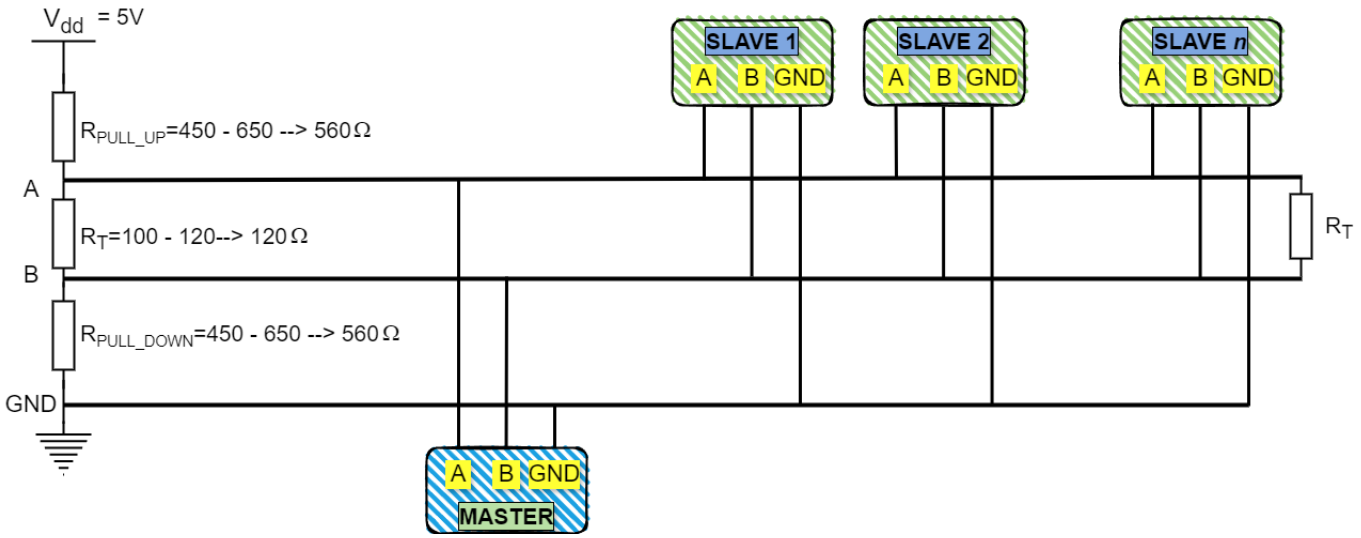
Należy zwrócić uwagę na bloki w pętli głównej trymujące dane odbierane po UART. Wysyłane dane kodowane są do ciągu tekstowego, uzupełnionego dodatkowo o znaki (`b'`) na początku i (`'`) na

końcu. Dodatkowo blok wysyłający linie na port dodaje na końcu tekstu znaki formatujące (`\r\n`). Rozwiązaniem optymalnym byłoby zastosowanie bloku *decode*, który oczyści ciąg tekstowy odebrany z portu UART z znaków formatujących. Jeżeli jednak dane wysyłane są jako surowe (RAW DATA) to próba ich zdekodowania zwróci błąd, w takim przypadku konieczne jest ręczne usunięcie znaków formatujących. Operacje trymowania można wykonać tylko na danych typu ***string***, dlatego konieczne jest prze konwertowanie odebranych danych na ten typ dedykowanym blokiem, jak pokazano na Rys.5.

Rys. 6: Efekt działania kodu z przykładu pokazanego na Rys.5

1.2 Protokół RS485

RS485 jest protokołem komunikacji szeregowej pozwalającym nie tylko na znaczne wydłużenie magistrali w stosunku do standardowego UART, ale także wiele urządzeń, jak pokazano na Rys.7. Standardowo do 32, ale przy zastosowaniu specjalnych sterowników możliwe jest rozszerzenie do 128, a nawet 256 urządzeń.



Rys. 7: Schemat połączeń dla komunikacji szeregowej po magistrali RS485

Magistrala RS485 pracuje w układzie **MASTER-SLAVE**, co oznacza że komunikacja odbywa się wyłącznie pomiędzy urządzeniem głównym MASTER (zarządzającym komunikacją) a wieloma urządzeniami **SLAVE**. Urządzenia **SLAVE** nie mogą komunikować się między sobą bezpośrednio. W pewnych możliwe jest działanie magistrali w trybie **Multi-MASTER** (*ModBus*), gdzie konieczne jest zaimplementowanie zaawansowanie kontroli kolizji transmisji. Urządzeniem typu master jest zazwyczaj mikrokontroler z podłączonym do jego portu szeregowego UART przetwornik RS485. Urządzenia

typu SLAVE zazwyczaj reagują na zapytania MASTER'a, pełniąc funkcje pomiarowe, sterownicze i wykonawcze:

- *Czujniki i mierniki* – urządzenia pomiarowe, które monitorują parametry fizyczne, takie jak temperatura, wilgotność, ciśnienie, poziom cieczy, przepływ czy wibracje. Mogą wysyłać odczytane dane na żądanie urządzenia master.
- *Sterowniki i kontrolery* – jednostki, które przyjmują polecenia od urządzenia master, na przykład do sterowania silnikami, pompami, zaworami, oświetleniem lub innymi elementami wykonawczymi. Wykonują one zadane operacje, takie jak włączanie, wyłączanie lub regulacja.
- *Moduły przekaźnikowe* – urządzenia sterujące, które umożliwiają przełączanie obwodów elektrycznych na podstawie sygnałów z magistrali. Mogą aktywować lub dezaktywować konkretne urządzenia, jak np. maszyny czy systemy zabezpieczeń.
- *Liczniki energii elektrycznej* – urządzenia używane do monitorowania zużycia energii elektrycznej, popularne w zakładach przemysłowych i systemach automatyki budynkowej. Master może odczytywać wartości licznika lub ustawiać konkretne tryby pracy.
- *Zdalne moduły I/O* – moduły rozszerzeń dla sygnałów wejścia/wyjścia, które umożliwiają monitorowanie stanu i sterowanie wyjściami cyfrowymi lub analogowymi. Mogą służyć do zdalnej kontroli różnych procesów.

W celu poprawy jakości transmisji po magistrali RS485 stosuje się, jak pokazano na Rys.7

- rezystory terminujące - umieszczane na końcach linii w celu eliminacji zjawiska odbicia sygnałów. Powinna być ona równa impedancji falowej linii, wynoszącej między 100 a 120 Ω. Zazwyczaj stosuje się rezystory o wartości 120 Ω.
- rezystory podciągające - **PULL_UP** i **PULL_DOWN** - zabezpieczające magistralę przed pływaniem potencjałów linii A i B gdy nie jest nadawana transmisja. Standardowo wartości tych rezystancji mieszczą się między 450 a 650 Ω, zazwyczaj stosuje się rezystory o wartości około 560 Ω.

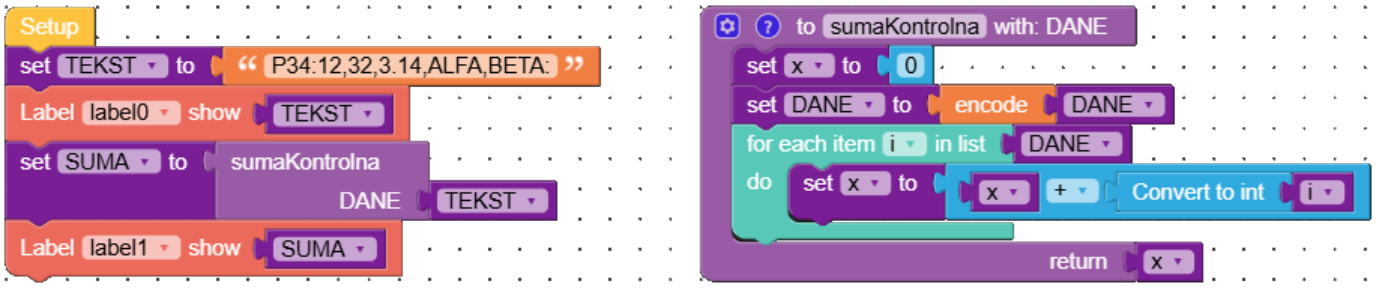
W odróżnieniu od standardowej komunikacji szeregowej (UART) ze względu na możliwość obsługi wielu urządzeń pakiet danych należy zebrać w zwarty blok, jak pokazano na Rys.8, zawierający informacje identyfikacyjne i kontrolne. Poszczególne człony takiego bloku można dodatkowo w sposób jednoznaczny rozdzielić, a całośc pakietu zakończyć.

Rys. 8: Przykład pakietu danych transmisji RS485

ADRES: jednoznaczna identyfikacja nadawcy lub adresata pakietu danych,

DANE: sformatowany ciąg danych, w przypadku większych (ilościowo) poszczególne elementu w sposób jednoznaczny rozdzielone (np. przecinkiem) lub zakodowane za pomocą ustalonej długości słowa (liczby bitów),

CRC: suma kontrolna lub inna metoda pozwalająca na zweryfikowanie poprawności zrealizowanej transmisji. Metodyka wyznaczania wartości kontrolnej może być realizowana w dowolnie przyjęty sposób np. jako suma kontrolna CRC8, CRC16 czy CRC32 lub zgodnie z indywidualnie przyjętą metodyką. Argumentami ją określającymi są wszystkie elementy pakietu danych znajdujące się przed członem kontrolnym.



Rys. 9: Przykład generowania sumy kontrolnej

Na Rys.9 pokazano prosty algorytm generujący sumę kontrolną ciagu tekstowego w oparciu o wartości kodowe znaków ASCII.

Dane (12,32,3.14,ALFA,BETA) wysyłane lub odbierane są od adresata "P34", a wyznaczona dla nich suma kontrolna wyniosłą 1375.

1.3 UIFlow - RS485

Korzystanie z magistrali RS485 przez aplikację tworzoną w środowisku UIFlow wymaga dodania wybranego UNIT'u z konwerterem sygnału UART na RS485. Dostępne są dwa układy, na płytce DEMO-BOARD zaimplementowano standardowy układ RS485, jak pokazano na Rys.10. Układ standardowo należny podpiąć do UART na portach 17 i 16. Możliwe jest także ręczne zdefiniowanie portów np na 1 i 3. Obydwa po dodaniu dostarczają identyczny zestaw bloków funkcyjnych, umiejscowionych w zakładce **UNIT**.

Rys. 10: UNIT konwertera RS485

Dostępne bloki podzielono na pięć grup:

1. inicjacji - konfigurujący szybkość transmisji i parametry ramki danych

2-3 wysyłania i odbierania danych - w zestawie i o właściwościach identycznych jak te dostępne dla standardowej komunikacji UART, pokazanej na Rys. 3 i 4

4-5 obsługa standardu komunikacji **ModBus** po magistrali RS485 (*wysyłanie* i *odbieranie*)

2 Komunikacja bezprzewodowa EspNow

Alternatywą dla komunikacji szeregowej realizowanej dla mikrokontrolerów m5Stack może być komunikacja bezprzewodowa EspNow. Protokół umożliwia połączenie wielu urządzeń na zasadach zbliżonych do tych stosowanych np. dla myszki bezprzewodowej 2.4 GHz.

Protokół pozwala na dwukierunkową komunikację dwóch urządzeń lub rozsyłanie danych do wielu urządzeń będących w zasięgu sieci. Możliwe jest zdefiniowanie 14 niezależnych kanałów komunikacyjnych. Możliwe jest nawiązanie szyfrowanego połączenia.

Procedura połączenia wymaga określenia trybu pracy poszczególnych urządzeń (MASTER lub SLAVE) i skonfigurowania ich parametrów zgodnie z zamieszczonym na Rys.11 diagramem.

Na urządzeniu typu MASTER można także sparować z urządzeniem SLAVE i wysłać z niego dane. Inicjacja parowania wymaga określenia ID pary i trybu pracy i szyfrowania.

- **ESP_IF_WIFI_STA** – interfejs stacji (*STA*):
  - Jest to tryb, w którym urządzenie działa jako klient sieci Wi-Fi, umożliwiając połączenie z punktem dostępu (np. routerem).
  - W ESP-NOW, interfejs STA jest zazwyczaj używany do komunikacji punkt-punkt lub punkt-wielopunkt (kiedy wiele urządzeń ESP komunikuje się między sobą bez pośrednictwa routera).
  - Działa głównie jako urządzenie odbiorcze, ale może również inicjować wysyłanie danych do innych urządzeń w zasięgu.
- **ESP_IF_WIFI_AP** – interfejs punktu dostępu (*AP*):
  - W tym trybie urządzenie pełni rolę punktu dostępu, do którego mogą podłączać się inne urządzenia Wi-Fi.
  - W kontekście ESP-NOW ten tryb bywa używany, jeśli urządzenie działa jako centralna jednostka nadawcza, która przesyła dane do innych urządzeń działających w trybie STA.
  - Pozwala na pełnienie roli głównego nadajnika w konfiguracjach punkt-wielopunkt, gdzie dane rozgłaszane są do wielu urządzeń naraz.

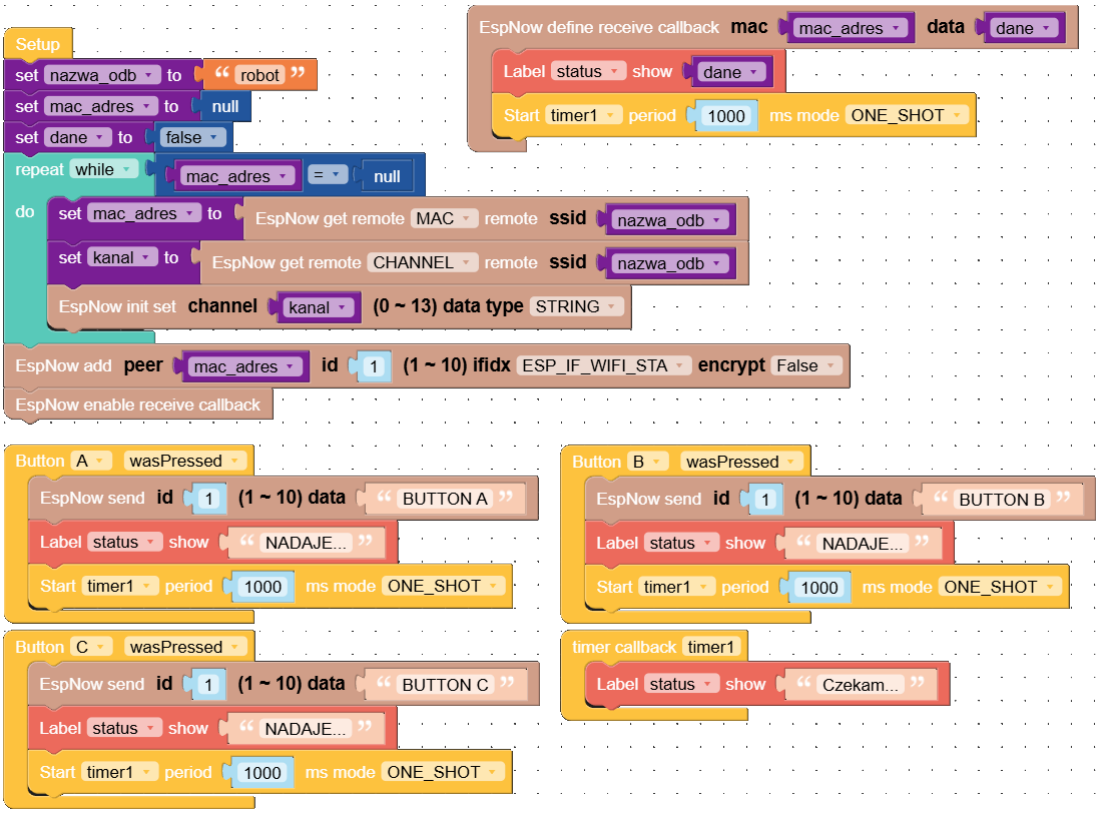
Rys. 11: Diagram inicjacji urządzeń MASTER i SLAVE

Działanie komunikacji bezprzewodowej EspNow pokazano na przykładzie realizującym zadanie opisane poniżej:

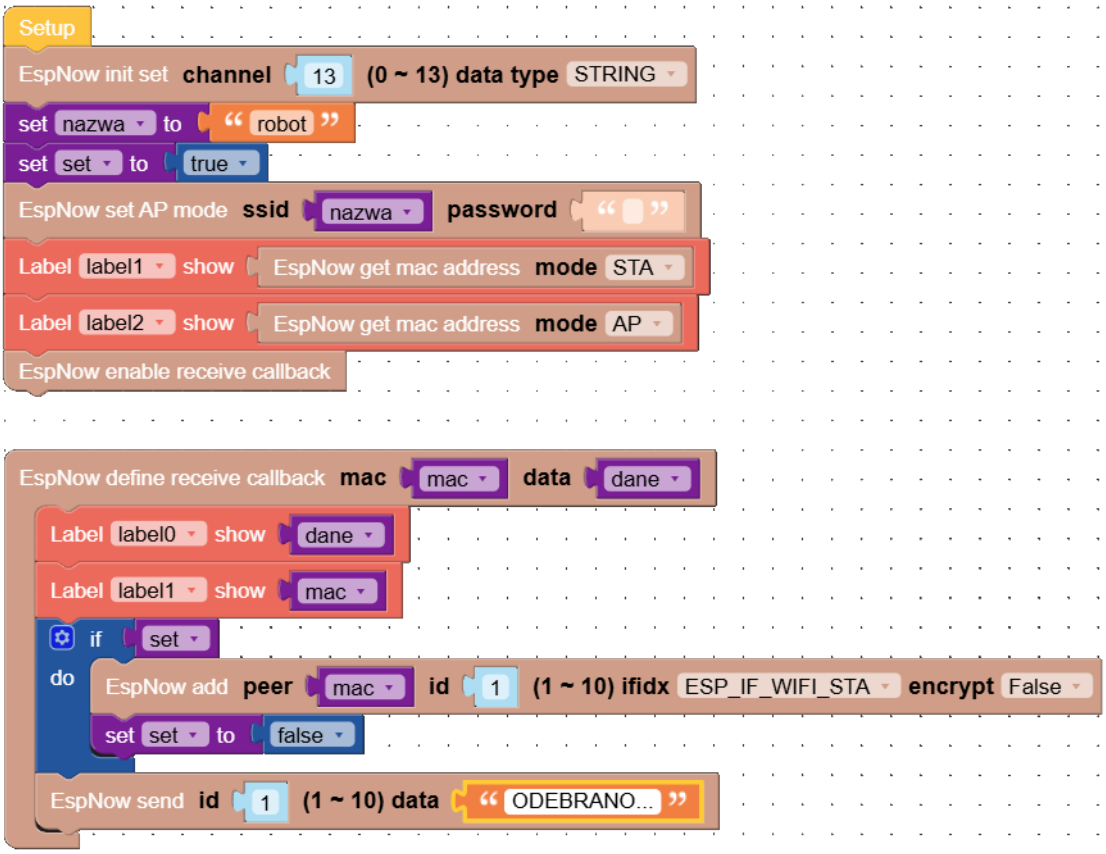
1. Urządzenie MASTER oczekuje na informację o przyciśnięciu przycisku na urządzenie SLAVE, wyświetla informacje o tym który z przycisków został naciśnięty i potwierdza informację o odebraniu danych

2. Urządzenie SLAVE nawiązuje połączenie z urządzeniem MASTER i wysyła do niego dane identyfikujące naciśnięcie wybranego przycisku. Wyświetla na ekranie aktualny status pracy (oczekiwanie, nadawanie, potwierdzenie dostarczenia danych.

   Na Rys.12 pokazano kod aplikacji MASTER a na Rys.13 kod aplikacji SLAVE.



Rys. 13: Kod aplikacji SLAVE



Rys. 12: Kod aplikacji MASTER

3 ZADANIA

Napisać aplikacji realizujące poniższe zadania:

Zad. 1: W oparciu o komunikację po RS485 napisać aplikację realizującą opisane poniższe zadania. Z sąsiadującą grupą ćwiczeniową przetestować działanie aplikacji.

Program w zależności od wybranego trybu pracy realizuje zadanie trybu **MASTER** (przycisk **A**) lub **SLAVE** (przycisk **B**). Przycisk **C** aktywuje wybrany tryb i wyświetla na ekranie informacje o trybie pracy.

MASTER: Program odczytuje parametry środowiskowe (temperaturę, ciśnienie i wilgotność) i wyświetla odczytane dane na ekranie mikrokontrolera. Aplikacja wysyła po porcie szeregowym RS485 wybrany parametr środowiskowy w odpowiedzi na otrzymane zapytanie. Na ekranie wyświetlana jest informacja o statusie transmisji (OCZEKIWANIE lub NADAWANIE), zapytaniu(TEMPERATURA, CIŚNIENIE lub WILGOTNOŚĆ) oraz czasowo wyświetlane potwierdzenie odebrania danych.

SLAVE: W zależności od wychylenia joystick'a program wysyła zapytanie o wybrany parametr środowiskowy, a naciśnięcie przycisku joystick'a wysyła zapytanie o wartość ciśnienia. Na ekranie wyświetlane są wartości poszczególnych parametrów środowiskowych, data i godzina ostatnie aktualizacji danych oraz informacje o statusie pracy (OCZEKIWANIA, ZAPYTANIE lub ODBIERANIE).

LEWO/GÓRA: Wybranym parametrem jest **temperatura**, wizualizacja danych temperatury.

PRAWO/DÓŁ: Wybranym parametrem jest **wilgotność**, wizualizacja danych wilgotności.

ŚRODEK: Dane nie są wizualizowane i nie są realizowane zapytania, serwomechanizm wraca po pozycji początkowej, a matryca led jest wygaszana.

Wizualizacja polega na wychyleniu ramienia serwomechanizmu proporcjonalnie do wartości aktywnego parametru, a na matrycy w układzie pionowym lub poziomym wyświetlana jest historia ośmiu ostatnio odczytanych jego wartości.

Zaimplementować kontrole poprawności transmisji, np. suma kontrolna oraz identyfikacje nadawcy, zapytania i odpowiedzi.

Pisząc kod programu realizujący przesyłanie danych pomiędzy dwoma urządzeniami w łączności przewodowej szeregowej czy bezprzewodowej musi formatować je do określonego standardu. Jest to szczególnie istotna gdy programy na obu urządzeniach piszą inne osoby. W przypadku realizowanego zadania konieczne jest ustalenia składni zapytań. Program przewiduje trzy typy komunikatów przesyłanych po magistrali RS485: ZAPYTANIE, ODPOWIEDŹ i POTWIERDZENIE. Trzeci z nich może być opcjonalny. Każdy komunikat powinien rozpoczynać się od fragmentu identyfikującego. Wskazane jest aby każde zapytanie zawierało weryfikującą sumę kontrolną.

- ZAPYTANIE - komunikat powinien zawierać w przypadku połączenia jeden do wielu lub wiele do wielu informacje o nadawcy i adresacie oraz kod identyfikujący cel zapytania. np: –> Z:M1:M2:T:A3;
- ODPOWIEDŹ - Analogicznie zawiera identyfikator typu komunikatu, celu, źródła, wielkości, wartości i sumy kontrolnej: np: –> O:M2:M1:T:24.12:FC;
- POTWIERDZENIE - opcjonalne komunikat potwierdzający dotarcie wysłanego wcześniej zapytania lub odpowiedzi: np: –> P:M1:M2:1C;

Zad. 2:

Zmodyfikować zadanie 1 aby komunikacja realizowana była za pomocą protokołu EspNow.