Arduino for STM32
Everything relating to using STM32 boards with the Arduino IDE

Arduino dla mikrokontrolerów STM32 (3)

W poprzednich artykułach z tego cyklu pokazano jak skonfigurować środowisko programistyczne Arduino do pracy z mikrokontrolerami STM32 oraz jak zrealizować aplikację używającą portów wejścia/wyjścia. Ta część kontynuuje wątek aplikacyjny demonstrując sposób wykorzystania kolejnego zasobu: interfejsu szeregowego UART.

Pracę należy rozpocząć uruchamiając środowisko programistyczne Arduino. Przygotowanie do pisania kodu obejmuje kilka kroków.

Arduino i STM32 – przygotowanie do pracy

W pierwszej kolejności konieczne jest stworzenie nowego projektu (zwanego sketchem w Arduino). W tym celu użytkownik wybiera z menu głównego środowiska programistycznego kolejno *File* i *New* (skrót klawiaturowy CTRL+N).

W następnym kroku należy wskazać używaną platformę sprzętową. Przykładowo autor wybrał płytkę NUCLEO-L476RG z 64-pinowym układem STM32L476RG. W tym celu należy ponownie użyć menu głównego środowiska programistycznego. Wybór platformy przebiega w dwóch etapach. Najpierw klikając *Tools* i *Board* wybrać należy *Nucleo-64*. Następnie, wybierając *Tools* i *Board Part Number* – należy wybrać *Nucleo L476*.

W ostatnim kroku należy zapisać projekt na dysku twardym. Jednocześnie warto zmienić nazwę projektu na bardziej intuicyjną, gdyż domyślna nazwa zawiera tylko słowo sketch oraz nazwę aktualnego miesiąca i dnia. Chcąc zapisać projekt oraz zmienić jego *nazwę* użytkownik znowu korzysta z menu głównego środowiska programistycznego wybierając kolejno *File* i *Save as…* (skrót klawiaturowy CTRL+Shift+S). W ten sposób otworzone zostanie okno, które pozwoli na wybranie ścieżki na dysku twardym, pod którą zapisany zostanie projekt. Jednocześnie okno pozwala na wpisanie nowej nazwy projektu.

Arduino i STM32 – struktura kodu

Każdy utworzony od nowa projekt (sketch) zawiera domyślnie kilka linii kodu tworzących szkielet aplikacji. Kod ten pokazano w **listingu 1**. Składa się on z dwóch funkcji. Pierwsza funkcja nosi nazwę *setup*. Jej przeznaczenie można łatwo rozszyfrować zapoznając się z komentarzem. Programista umieszcza w niej kod, który ma zostać wykonany tylko raz, na początku aplikacji, a więc po doprowadzeniu napięcia zasilania do mikrokontrolera. Są to głównie funkcje konfigurujące peryferia mikrokontrolera (np. ustawiające parametry przetwornika A/C lub interfejsu UART) lub inicjujące podzespoły zewnętrzne (np. moduł Bluetooth). Druga funkcja otrzymała nazwę *loop*. Ponownie w rozszyfrowaniu jej przeznaczenia pomaga komentarz. Jest to nieskończona pętla, w której programista umieszcza funkcje wykonywane cyklicznie (np. odczyt stanu przycisku lub odebranie danych z czujnika). Zatem programista tworzy kod aplikacji uzupełniając ciała funkcji *setup* i *loop* o dodatkowy kod języka Arduino. Kod ten składa się z typowych elementów języka programowania takich jak zmienne, funkcje, instrukcje warunkowe czy też operacje matematyczne.

Interfejs szeregowy UART – podstawowe informacje

Mając skonfigurowane środowisko programistyczne Arduino oraz znając podstawy programowania Arduino można przejść do kolejnego poziomu wtajemniczenia, jakim jest zrozumienie podstaw dotyczących interfejsu komunikacyjnego UART.

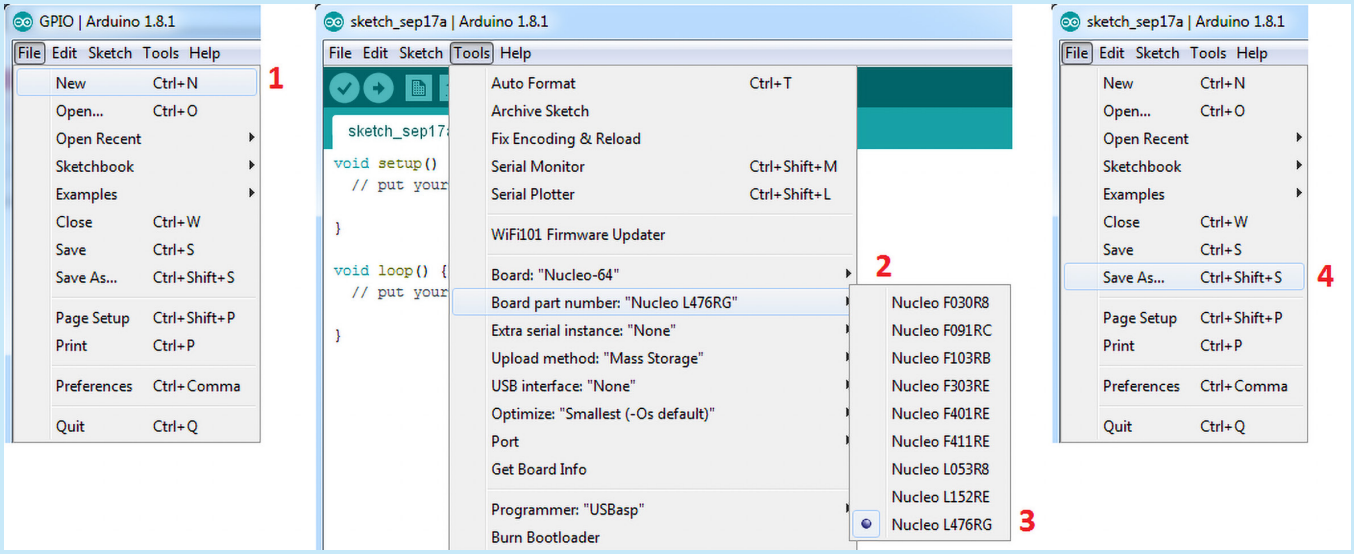
Podstawowym zasobem służącym mikrokontrolerowi do komunikacji ze światem zewnętrznym są porty wejścia/wyjścia. Sterowanie portami przez programistę z poziomu aplikacji dobrze sprawdza się w sytuacji, gdy komunikacja sprowadza się do wytwarzania i odczytywania pojedynczych stanów logicznych. W praktyce jednak wiele układów współtworzących system elektroniczny komunikuje się przez sekwencje bitów. Rozróżnia się dwa modele takiej komunikacji: transmisję szeregową (bity przesyłane są jeden po drugim przez jedną lub kilka linii sygnałowych) i transmisję równoległą (każdy bit słowa przesyłany jest przez oddzielną linię sygnałową). Realizacja takiej transmisji za pomocą portów w trybie GPIO nie jest optymalnym rozwiązaniem. Po pierwsze dlatego, że wiąże się to z określoną pracą implementacyjną, którą musi wykonać

Listing 1. Kod nowego projektu w środowisku programistycznym Arduino

```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Poprzednie części kursu i dodatkowe materiały dostępne są na na stronie www.media.avt.pl



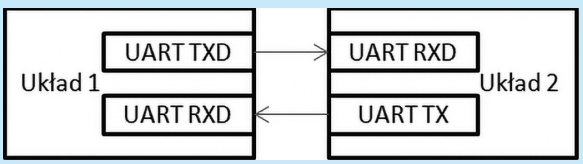
Rysunek 1. Kolejne kroki przygotowania środowiska Arduino do pisania kodu dla wybranej platformy z układem STM32

programista (mniej lub bardziej czasochłonną w zależności od poziomu skomplikowania transmisji). Po drugie w przypadku gdy aplikacja wymaga od mikrokontrolera częstego komunikowania się z układem lub układami, transmisja danych może okazać się niezbyt efektywna z punktu widzenia czasu wykonania (powód: implementacja ma charakter programowy, nie sprzętowy). Aby rozwiązać przedstawione problemy powszechnym rozwiązaniem stało się integrowanie w mikrokontrolerach kontrolerów odpowiedzialnych za sprzętową realizację komunikacji. Najpopularniejsze interfejsy komunikacyjne to I2C, SPI oraz będący przedmiotem tego artykułu UART.

W warstwie fizycznej komunikacja UART realizowana jest przy użyciu dwóch linii o nazwie TXD i RXD. Każda z nich przewidziana jest do jednokierunkowego przesyłania danych: TXD jest wyjściem nadajnika, a RXD jest wejściem odbiornika. UART w podstawowej formie jest komunikacją asynchroniczną, a więc linia z sygnałem zegarowym nie jest używana. Jednak w szczególnych przypadkach UART może wykorzystywać linię zegarową i wtedy ma miejsce komunikacja synchroniczna (USART). Ponadto UART jest komunikacją typu punkt-punkt, co oznacza, że przy użyciu jednego łącza mogą komunikować się tylko dwa układy. Schemat połączenia elektrycznego właściwego dla transmisji UART pokazano na **rysunku 2**.

Informacje w interfejsie UART przesyłane są zgodnie ze schematem przedstawionym na **rysunku 3**. Kolejno wyróżnić można: sygnalizujący rozpoczęcie ramki bit startu o poziomie niskim, od 5 do 8 bitów danych, opcjonalny bit parzystości służący do sprawdzenia poprawności danych oraz sygnalizujący zakończenie ramki bit (1, 1,5 lub 2) stopu o poziomie wysokim.

Popularnymi podzespołami używającymi interfejsu UART są moduły Bluetooth, GSM i GPS. Warto pamiętać, że zastosowanie transmisji UART nie ogranicza się tylko do komunikacji mikrokontrolera z komponentami otaczającymi go w systemie. Rozwiązanie to stanowi również podstawę standardów komunikacji między urządzeniami: standardu komputerowego RS-232 i sieciowych standardów przemysłowych RS-422 i RS-485 o topologii magistrali.



Rysunek 2. Schemat połączenia dwóch układów komunikujących się poprzez interfejs UART

Interfejs szeregowy UART – funkcje Arduino

Do sterowania interfejsem UART w Arduino wystarczy znajomość kilku funkcji. Nazywają się one następująco: *Serial.begin*, *Serial.println*, *Serial.available* oraz *Serial.readStringUntil*.

Funkcja *Serial.begin* odpowiada za skonfigurowanie interfejsu UART. Funkcja ta przyjmuje jeden argument, jakim jest prędkość transmisji, tak zwany baudrate. Warto w tym miejscu wspomnieć, że funkcja *Serial.begin* nie daje użytkownikowi możliwości wyboru który z interfejsów UART mikrokontrolera ma być wykorzystany. Wynika to z tego, że w Arduino przewidziano wykorzystanie tylko jednego interfejsu UART.

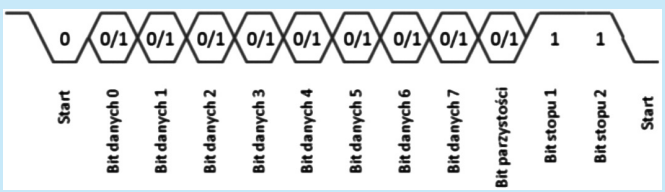
Funkcja *Serial.println* pozwala na wysłanie treści interfejsem UART. Funkcja ta przyjmuje jeden argument, który jest właśnie wspomnianą treścią. Treść ta najpierw jest przekazywana do nadajnika kontrolera UART, a następnie zostaje wysłana linią nadawczą TXD. Co istotne, argumentem funkcji *Serial.println* może być zarówno ciąg znaków ASCII podany wprost, jak również zawartość wcześniej zdefiniowanej zmiennej bądź też tablicy.

Funkcja *Serial.available* służy do sprawdzania czy odbiornik interfejsu UART dysponuje nową treścią. Funkcja ta nie przyjmuje żadnego argumentu, za to wartością przez nią zwracaną jest liczba otrzymanych przez odbiornik bajtów treści.

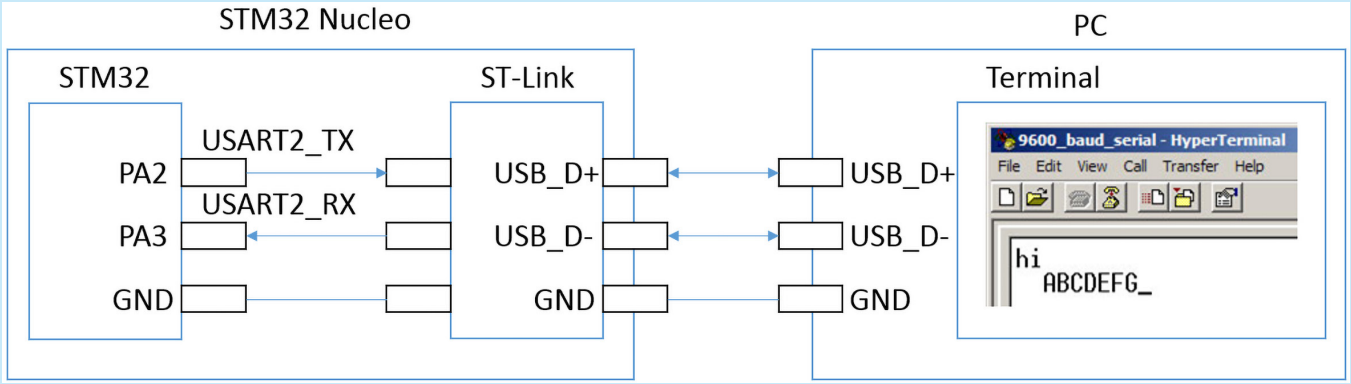
Funkcja *Serial.readStringUntil* daje możliwość odczytania treści przechowywanej przez odbiornik interfejsu UART. Funkcja zwraca odebraną treść kończąc na znaku, który podany jest jako argument funkcji (tak zwany terminator).

Interfejs szeregowy UART – przykładowa aplikacja

Płytka NUCLEO-L476RG dysponuje programatorem/debugerem ST-Link, którego dodatkową funkcją jest pełnienie roli mostu pomiędzy interfejsem UART i interfejsem USB emulującym port COM. Dzięki takiemu rozwiązaniu aplikacja mikrokontrolera może komunikować się poprzez interfejs UART z terminalem komputerowym. Po stronie mikrokontrolera transmisja realizowana jest przez porty



Rysunek 3. Format komunikacji w interfejsie UART



Rysunek 4. Schemat blokowy do komunikacji szeregowej na płytce NUCLEO-L476RG

Listing 2. Kod programu realizującego dwukierunkową transmisję danych przez interfejs UART

```
String Data = „”;

void setup()
{
    // put your setup code here, to run once:
    Serial.begin(9600);
}

void loop()
{
    // put your main code here, to run repeatedly:
    if(Serial.available() > 0)
    {
        Data = Serial.readStringUntil('\n');
        Serial.println(„STM32 odebral: „ + Data);
    }
}
```

PA2 (USART2_TX) i PA3 (USART2_RX). Odpowiedni schemat pokazano na **rysunku 4**.

Po zapoznaniu się najpierw z funkcjami Arduino do obsługi interfejsu szeregowego UART, a następnie po przeanalizowaniu schematu połączenia interfejsu szeregowego, jesteśmy gotowi do napisania programu realizującego dwukierunkową transmisję danych. Zaprezentowany na **listingu 2** kod zawiera kolejno:

- Przed funkcją *setup*: deklarację łańcucha znaków *Data*.
- Wewnątrz funkcji *setup*: wywołanie funkcji *Serial.begin* ustawiającej prędkość transmisji interfejsu UART.
- Wewnątrz funkcji *loop*:
  - użycie instrukcji warunkowej, która poprzez wywołanie funkcji *Serial.available* sprawdza czy odbiornik interfejsu UART dysponuje nową treścią,
  - w przypadku informacji o odebraniu nowych danych:
    - wywołanie funkcji *Serial.readStringUntil* odczytującej treść z odbiornika interfejsu UART i zapisującej ją w łańcuchu znaków *Data*,
    - wywołanie funkcji *Serial.println*, która wysyła do komputera prostą wiadomość tekstową oraz odsyła odebraną wcześniej treść.

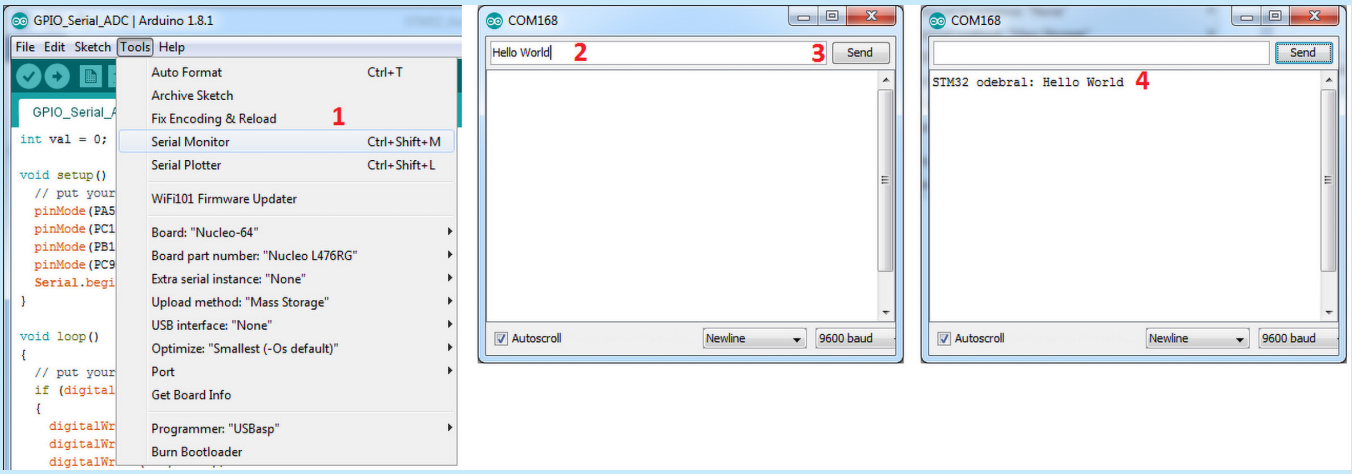
Po napisaniu kodu program należy skompilować wybierając przycisk *Verify* (skrót klawiaturowy CTRL+R). Po zakończeniu procesu kompilacji należy wgrać program do pamięci mikrokontrolera wybierając przycisk *Upload* (skrót klawiaturowy CTRL+U).

W efekcie działania tej aplikacji każdorazowo po wysłaniu z terminala komputerowego treści zakończonej znakiem nowej linii, mikrokontroler odsyła tą treść z powrotem do terminala dodając informację, że treść ta pochodzi od układu STM32. Działanie aplikacji można obserwować z poziomu środowiska programistycznego Arduino IDE, które dysponuje wbudowanym terminalem komputerowym portu COM. Terminal ten można włączyć z poziomu menu głównego wybierając kolejno *Tools* i *Serial Monitor* (skrót klawiaturowy CTRL+Shift+M). Kolejne kroki pracy z aplikacją pokazano na **rysunku 5**.

Podsumowanie

Trzeci artykuł z cyklu traktującego o Arduino i STM32 kontynuuje wątek aplikacyjny koncentrując się tym razem na interfejsie szeregowym UART. W materiale zaprezentowano sposób konfiguracji interfejsu (ustawienie prędkości transmisji) oraz przedstawiono jak wysyłać i odbierać dane. Kolejne artykuły rozwijać będą aspekt prostych aplikacji korzystających z zasobów mikrokontrolera. Tematyka obejmować będzie przetwornik A/C (aplikacja: odczytywanie napięcia na pinie mikrokontrolera) oraz Timer (wytwarzanie na pinie sygnału PWM).

Szymon Panecki
STMicroelectronics
szymon.panecki@st.com



Rysunek 5. Praca z aplikacją realizującą komunikację UART