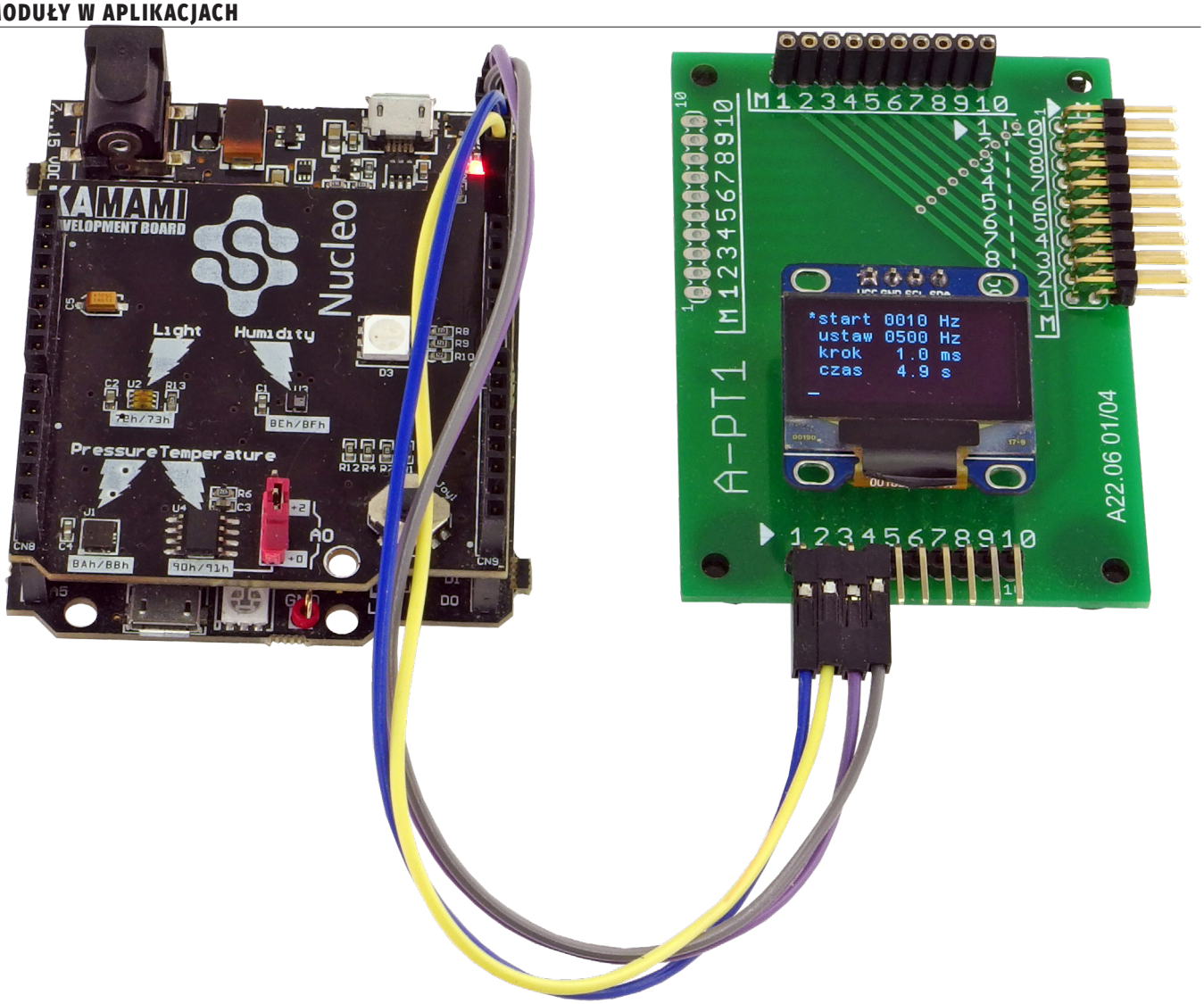


Generator symulujący impulsy przyśpieszenia

Projekt powstał na potrzeby takiego zadania: należało skonstruować układ kontrolujący działanie dużej, trudno dostępnej maszyny. Mechanizm maszyny podczas pracy przyśpiesza i zwalnia. Czujnik zintegrowany z mechanizmem generuje impulsy, których okres trzeba rejestrować i mierzyć. Pisząc oprogramowanie dla układu nadzorującego, chcemy na bieżąco przeprowadzać testy i usuwać błędy. Ponieważ ciągły dostęp do maszyny jest niemożliwy, potrzebujemy symulatora, który odwzoruje sygnał z czujnika maszyny.

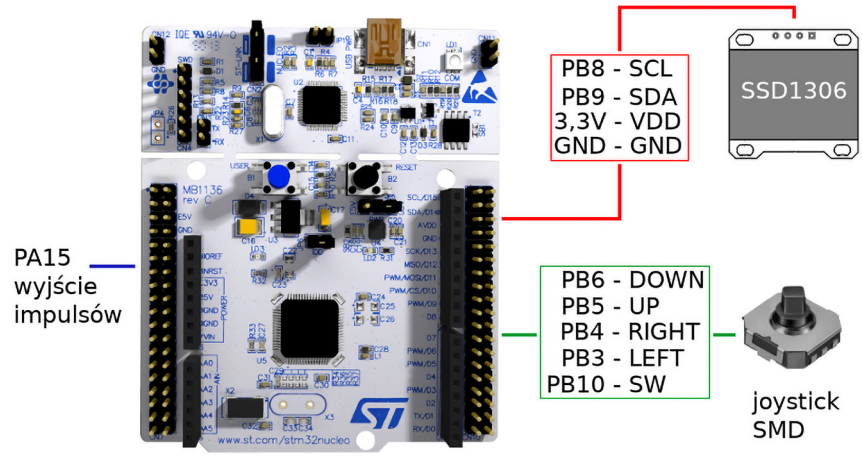
Założenia

Ruch prawdziwej maszyny, np. wału silnika napędowego, nie jest stały, ale stopniowo i płynnie przyśpiesza lub zwalnia. Nasza symulacja będzie wystarczająco dokładna, jeżeli na potrzeby testu utrzyma stały przyrost szybkości, pomijając etapy rozpędzania i hamowania. Zakres częstotliwości generowanych impulsów także nie będzie szeroki: od 10 Hz do 500 Hz. Powinna istnieć możliwość wyboru tempa liniowego rozpędzania lub hamowania. Przyjmijmy, że będzie ono ustawiane w zakresie od 0,1% do 10% przyrostu pomiędzy jednym okresem impulsów a kolejnym.

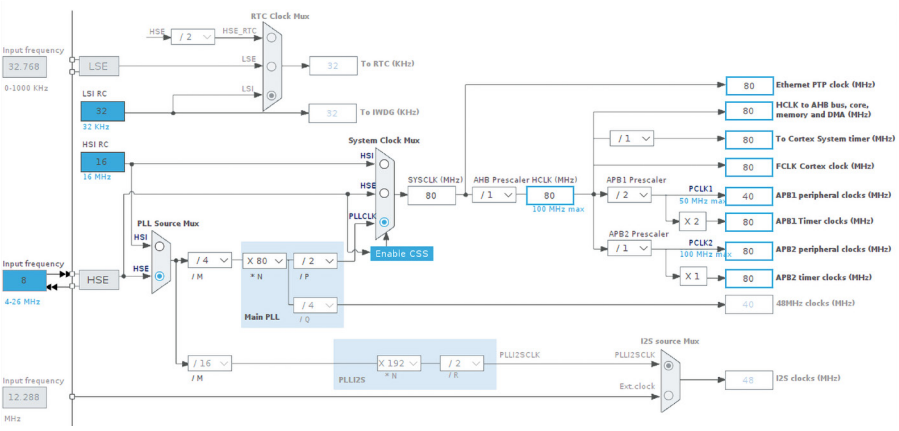
Jako część sprzętowa generatora-symulatora posłuży kontroler w postaci płytki Nucleo z STM32F411. Miniaturowy wyświetlacz OLED i mikrojoystik będą pełniły funkcję interfejsu użytkownika służącego do wprowadzania nastaw i wyświetlania bieżących parametrów pracy symulatora. Na **rysunku 1** pokazano schemat połączeń pomiędzy trzema elementami symulatora.

Generowanie impulsów

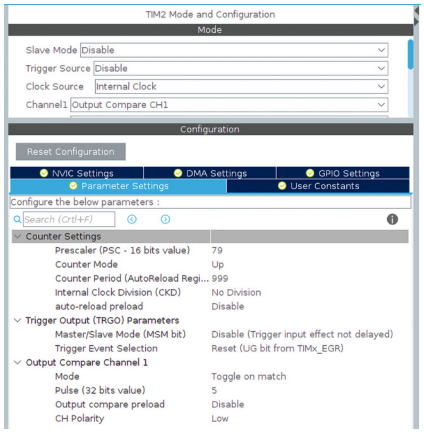
Kontrolery STM32 są doskonale przygotowane do generowania impulsów. Kilka wewnętrznych programowalnych liczników (*Timer*) pozwala na wytwarzanie zarówno pojedynczych impulsów, jak i przebiegów o stałej częstotliwości i zmiennym wypełnieniu (PWM). Do generownaia przebiegu o zmiennym okresie symulującym impulsy wytwarzane przez maszynę użyjemy licznika Timer2 i jego wyjścia OC1 (*Output Compare*). Timer2 jest 32-bitowym licznikiem,



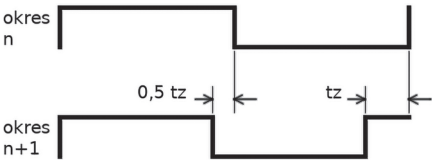
Rysunek 1. Połączenia pomiędzy częściami składowymi symulatora



Rysunek 2. Ustawienia sygnałów zegarowych w STM32CubeMX



Rysunek 3. Ustawienia parametrów Timera2 w STM32CubeMX



Rysunek 4. Impulsy wyjściowe generowane przez symulator

co pozwala bez większych komplikacji używać go do generowania zarówno szybkich przebiegów, jak i wolnych, o długim okresie.

Dla zapewnienia dobrej rozdzielczości ustawień licznik powinien być taktowany szybkimi impulsami zegarowymi o okresie 1 µs, co odpowiada częstotliwości 1 MHz. Na **rysunku 2** pokazano ustawienia wewnętrznych przebiegów zegarowych procesora przeprowadzone za pomocą programu narzędziowego STM32CubeMX. Główny sygnał zegarowy HCLK ma częstotliwość 80 MHz. Preskaler Timera2 ustawiony na 79 wytwarza z sygnału HCLK impulsy zegarowe dla Timera2 o częstotliwości 1 MHz. Pozostałe ustawienia STM32CubeMX dla Timera2 i wyjścia OC1 są widoczne na **rysunku 3**. Dla ustawień parametrów: *Counter Period* 999, *Mode Toggle on match*, *Pulse* 5, symulator będzie gotowy do generowania przebiegu o okresie 2 ms (500 Hz) i wypełnieniu 50%. Timer2, po osiągnięciu wartości ustawionej w *Counter Period*, zeruje się i zaczyna liczyć od początku. Po osiągnięciu wartości zapisanej w *Pulse* wyjście OC1 wyprowadzone na port PA15 zmienia swój stan na przeciwny, co zadeklarowano ustawieniem parametru *Mode*. Do wygenerowania pełnego okresu impulsu potrzebne są dwa przejścia Timera2.

Symulacja przyśpieszania lub zwalniania maszyny wymaga, aby każdy następny generowany okres odpowiednio wydłużać lub skracać, co pokazano na **rysunku 4**. Żeby uzyskać skrócenie kolejnego okresu n+1 o wartość *tz*, należy skrócić jego półokresy o ½ *tz*. W oprogramowaniu symulatora maszyny dzieje się to podczas przerwania *HAL_TIM_OC_DelayElapsedCallback*, którego konstrukcję można obejrzeć na **listingu 1**. W zmiennej *okres_timera* przechowywana jest wartość, przy której następuje zerowanie Timera2, co odpowiada półokresowi generowanego przebiegu liczonego w mikrosekundach. Rejestr ARR określa zakres zliczania Timera2 i właśnie dlatego wartość zmiennej *okres_timera* przepisywana jest do tego rejestru. W procedurze wywoływanej w przerwaniu *GeneratorKrokiRampy()* obliczana jest nowa wartość zmiennej *okres_timera*. Jeżeli będzie symulowane zwalnianie prędkości maszyny, następny okres Timera2 będzie wydłużany o ustawioną wartość procentową z przedziału 0,1% (powolne zwalnianie) do 10% (bardzo szybkie hamowanie). Do symulowania przyśpieszenia kolejne okresy Timera2 będą skracane.

Większa część oprogramowania służy jako interfejs do wyświetlania stanu symulatora maszyny oraz do wprowadzania nastaw. Na miniaturowym wyświetlaczu OLED SSD1306 można odczytać informacje o bieżącej i ustawianej częstotliwości oraz o ustawionej szybkości zmian przyśpieszenia wyrażonej jako wartość procentowa przyrostu dla kolejnych okresów. Wyświetlany jest także wyliczony czas w sekundach potrzebny na osiągnięcie zadanej częstotliwości. Mikrojoystik służy do przemieszczania kursora na wyświetlaczu oraz do wprowadzania nowych ustawień i wyzwalania cyklu zmiany przyśpieszenia. Do przełączania pomiędzy kolejnymi ekranami menu służy dodatkowy przycisk podłączony do portu PB12.

Na **fotografii tytułowej** pokazano działający model symulatora. Do budowy zastosowano płytkę KA-NUCLEO-F411 będącą odpowiednikiem NUCLEO-F411RE firmy ST. Ze względu na zamontowany mikrojoystik użyto także płytki KA-NUCLEO-Weather.

Ryszard Szymaniak
biuro@ars.info.pl

Listing 1

```
//---------------------------------
//obsługa przerwania kanałów OC TIM2
void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef * htim)
{
    if (htim->Instance ==TIM2){
        //obsługa przerwania TIM2 CC1 odliczania okresu ustawionej
        //częstotliwości, rozdzielczość 1us
        if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1){
            if (zmiana_polowki_okresu ==TRUE)
                zmiana_polowki_okresu =FALSE;
            else {
                zmiana_polowki_okresu =TRUE;
                if (ustaw_nowy_okres_timera==TRUE){
                    TIM2-> ARR = okres_timera-1;
                    ustaw_nowy_okres_timera =FALSE;
                    GeneratorKrokiRampy();
                }
            }
        }
    }
}
```