Sterowniki robotów

Laboratorium – ADC, DAC i DMA

Przetwoniki analogowo–cyfrowe, cyfrowo–analogowe oraz bezpośredni dostęp do pamięci

Wojciech Domski

Dokument `sr_lab04.pdf`, Wersja 1.1.2

Spis treści

1 Wprowadzenie

Oprócz liczników [3] mikrokontrolery wyposażone są w przetworniki analogowo–cyfrowe (ang. *ADC*), a także w przetworniki cyfrowo–analogowe (ang. *DAC*). ADC pozwala na zmianę mierzonego napięcia na postać bitową, natomiast DAC generuje napięcie z zadaną rozdzielczością na podstawie wartości bitowej w odpowiednim rejestrze. Bardzo istotnym elementem mikrokontrolerów firmy ST jest układ DMA, który pozwala na przepisywanie danych znajdujących się w różnych obszarach pamięci bez udziału rdzenia.

2 Opis ćwiczenia

Uwaga! Przykłady implementacji różnych peryferiów mikrokontrolera można znaleźć w plikach biblioteki HAL dla danej rodziny układów. Pliki te znajdują się zazwyczaj w ustaloenj ścieżce, np. *C:/Users/Wojciech Domski/STM32Cube/Repository/*. Jednakże położenie tych plików może być różne w zależności od systemu operacyjnego, jak i miejsca instalacji biblioteki. Aby zweryfikować katalog przechowywania repozytorium należy uruchomić program `STM32CubeMX`, a następnie wybrać *Help* → *Updater settings ...* i odczytać zawartość pola *Repository folder*. Przykładami znajdują się w podkatalogu *STM32Cube_FW_L4_V1.7.0/Projects/STM32L476RG-Nucleo/Examples/*. podzielonym na podfoldery ze względu na wykorzystywane peryferium.

W instrukcji zostanie zaprezentowana obsługa przetwornika ADC, DAC oraz DMA. Dzięki tym peryferiom mikrokontrolera możliwa jest wydajna komunikacja (DMA) z światem zewnętrznym. Przetworniki ADC pozwalają na pomiar napięcia, które konwertowane jest do liczby binarnej, dzięki czemu umożliwia pomiar fizycznych wartości z zadaną dokładnością. DAC w odróżnieniu do ADC jest wyjściem, które można wykorzystać do generowania sygnału napięciowego z zadaną rozdzielczością. Dzięki temu w aplikacjach, w których wymagana jest kontrola pokrywająca więcej poziomów wyjściowych napięcia niż stan wysoki i niski, przetwornik DAC jest niezastąpiony.

3 Zadania do wykonania

Uwaga! Pamiętaj, aby wyłączyć optymalizację kodu (-O0), a także wykorzystać kompilację równoległą (*parallel*) np. -j8 [4]. Ponadto, ustaw automatyczny zapis przed wykonaniem kompilacji, aby uniknąć problemów związanych z rozbieżnością kodu, a plikiem wynikowym wgranym na mikrokontroler. Pamiętaj również, że gdy powtórnie generujesz projekt może okazać się konieczne jego wyczyszczenie jak i przebudowanie indeksu. W tym celu rozwiń menu kontekstowe dla projektu i wybierz *Clean project*, a nastepnie powtórz operację oraz wybierz *Index* → *Rebuild*.

Należy wykonać trzy ćwiczenia, każde z nich wykorzystuje jedno z peryferiów, bądź ich połączenie.

3.1 Przetwornik analogowo–cyfrowy ADC

Jak już wspomniano na wstępie przetwornik ADC wykorzystywany jest do zmiany mierzonej wartości fizycznej (napięcia) na liczbę całkowitą z przedziału $[0, 2^n - 1]$, gdzie n jest rozdzielczością przetwornika. Innymi słowy mierzone napięcie podlega kwantyzacji, gdzie liczba poziomów wynosi odpowiednio 2^n.

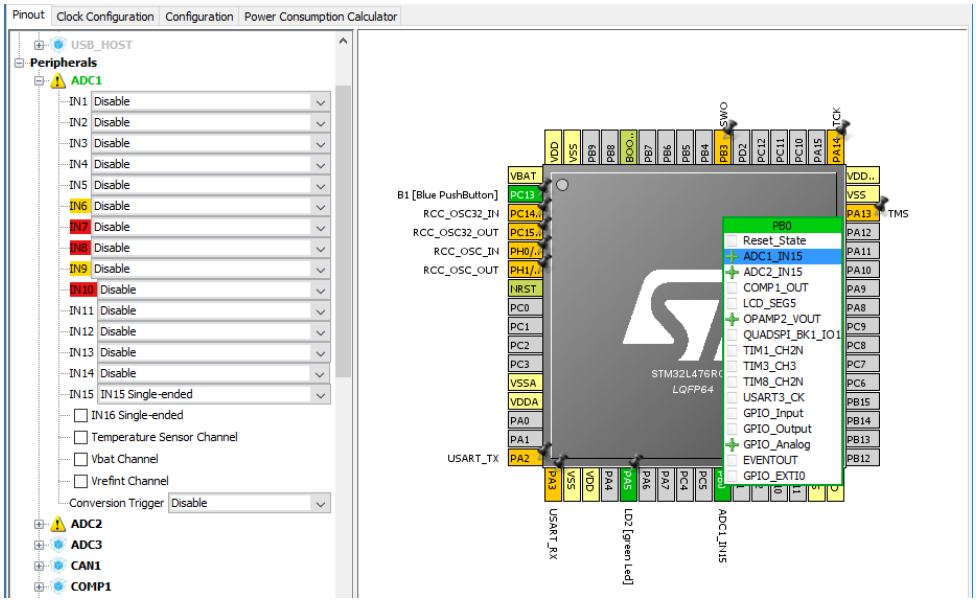
Mikrokontroler firmy ST, jak np. STM32L476RGT6 [6], [12] posiada aż 3 przetworniki analogowo–cyfrowe. Zazwyczaj każdy z przetworników posiada od kilku do kilkunastu kanałów pomiarowych. Dzięki wewnętrznemu multipleksowaniu kanałów możliwy jest pomiar od kilku do kilkunastu różnych napięć na różnych pinach mikrokontrolera. Pomiar więcej niż jednego kanału w obrębie pojedynczego przetwornika nie jest możliwy. Aby osiągnąć taką funkcjonalność wymagane jest wykorzystanie różnych przetworników ADC, jak to jest w przypadku wykorzystywanego mikrokontrolera.

ADC może być skonfigurowane na wiele różnych sposobów pracy. Jednym z podstawowych jest konfiguracja jednorazowego pomiaru w trybie pojedynczego wejścia wyzwalanego programowo. W przypadku, gdy występuje potrzeba pomiaru więcej niż jednego kanału możliwe jest skonfigurowanie przetwornika w taki sposób, że pomiar kolejnych kanałów jest wykonywany automatycznie. Możliwe jest również ustawienie ciągłego skanowania mierzonych wejść analogowych dzięki czemu zmierzone napięcia na różnych portach są aktualizowane bez ingerencji w konfigurację. Ponadto, możliwa jest takie ustawienie rejestrów konfiguracyjnych ADC, aby wykorzystywane było DMA, które odciąża rdzeń podczas przepisywania pomiarów.

Przygotuj nowy projekt dla płytki deweloperskiej NUCLEO-L476RG [9], [10] w programie `STM32CubeMX` [7], [11]. Celem ćwiczenia jest napisanie aplikacji, która nieustannie mierzy wartość napięcia na jednym z kanałów ADC. Kanał pomiarowy jest połączony za pomocą przewodu z portem

PC13, który wykorzystywany jest jako wejście cyfrowe. Naciskając niebieski przycisk na płytce wartość mierzona na porcie ADC PB0 ulega zmianie.

Wykorzystaj pin PB0, jako wejście analogowe. Zostało to pokazane na rysunku 1. Należy najpierw wybrać tryb pracy portu, a następnie wybrać funkcję pomiarową jednego wejścia (ang. *Single-ended*) w peryferium ADC1.



Rysunek 1: Konfiguracja wejścia numer 15 dla przetwornika ADC1

Po wyborze kanału przetwornika ADC należy przystąpić do konfiguracji peryferium. Przykładowa konfiguracja została pokazana na rysunku 2. Zwróć uwagę na czas próbkowania wyrażony w cyklach.

Uwaga! Jak długo będzie trwał pomiar wartości analogowej dla zaproponowanej konfiguracji z rysunku 2 [12]? Od czego zależy czas trwania pomiaru?

Kolejnym krokiem jest włączenie przerwań dla ADC. Zostało to zaprezentowane na rysunku 3. Skonfiguruj również USART2 w celu późniejszego przekierowania funkcji *printf*() [2].

Wygeneruj kod z odpowiednimi ustawieniami [4], a następnie przystąp do modyfikacji pliku *main.c*. W owym pliku należy umieścić poniższy kod wewnątrz nieskończonej pętli `while(1)`:

```
if (adc_flag == 1) {
  printf("Zmierzona wartosc to: %d\r\n", adc_value);
  adc_flag = 0;

  HAL_ADC_Start_IT(&hadc1);
}

HAL_Delay(1000);
```

W odpowiednim miejscu przed wejściem do nieskończonej pętli `while(1)` wywołaj funkcję `HAL_ADC_Start_IT()`.

Uwaga! Co się stanie gdy powyższa funkcja nie zostanie wywołana przed wejściem do pętli `while(1)`?

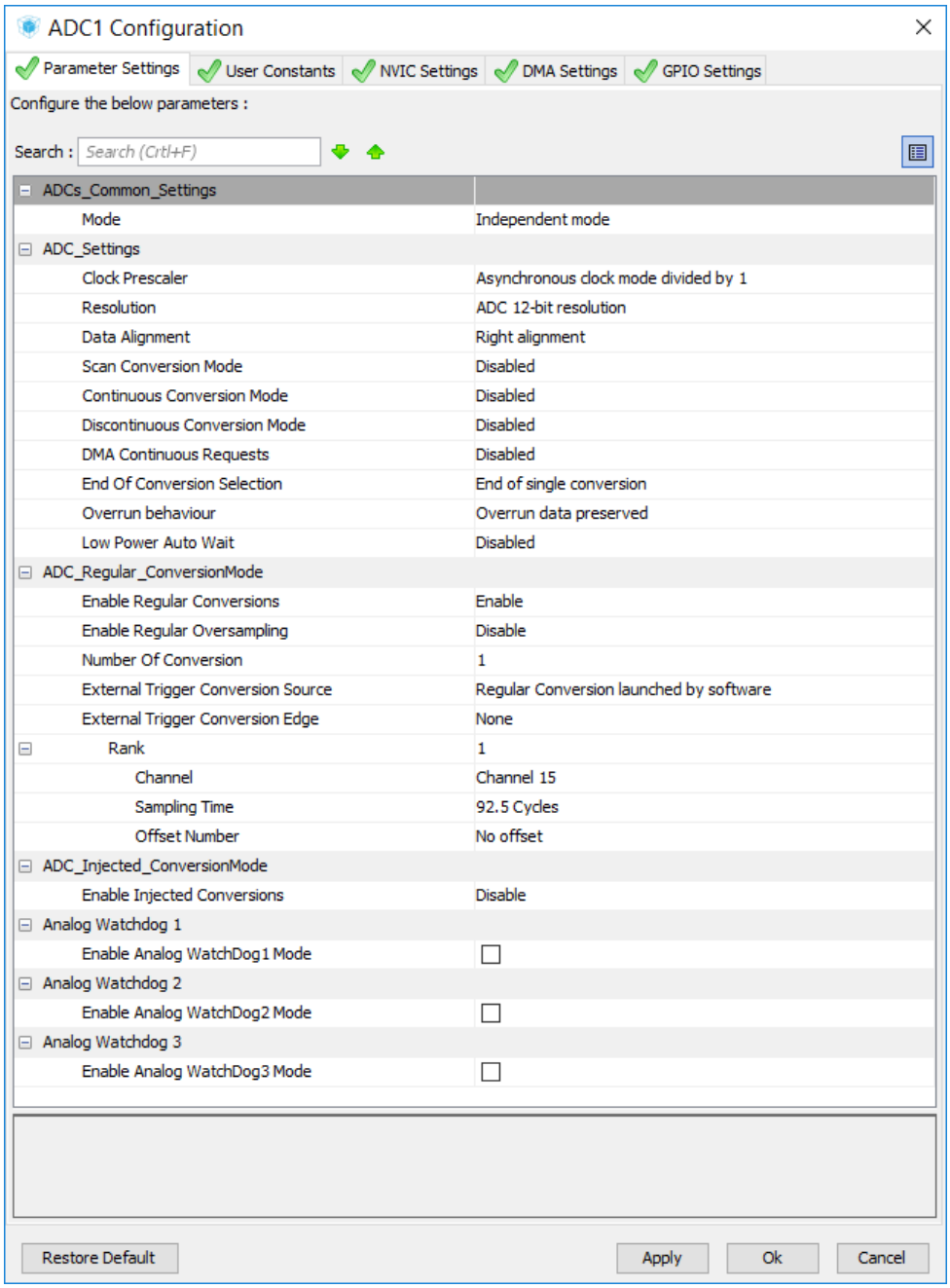
Na podstawie powyższego kodu stwórz w programie odpowiednie zmienne. Ponadto, zdefiniuj funkcję `HAL_ADC_ConvCpltCallback()`, w której będzie sprawdzane, czy funkcja zwrotna dotyczy odpowiedniego peryferium, a także będzie odczytywana wartość zmierzonego napięcia i ustawiana flaga. Do odczytu wartości zmierzonej na danym kanale przetwornika analogowo–cyfrowego wykorzystaj funkcję `HAL_ADC_GetValue()`. Nie zapomni również o redefinicji funkcji `_write()`.

Przystąp do testowania oprogramowania. W tym celu odłącz płytkę od komputera, a następnie połącz przewodem piny PC13 oraz PB0. Prawidłowo podłączony przewód do płytki został przedstawiony na zdjęciach 4(a) i 4(b).

Uwaga! Poinformuj prowadzącego o przygotowaniu płytki testowej!

Do odbierania danych z portu szeregowego wykorzystaj np. program Termite [1], bądź PuTTY [13]. Zaobserwuj działanie aplikacji podczas naciskania przycisku, przykładowe działanie programu zostało zaprezentowane na rysunku 5 podczas, gdy niebieski przycisk podłączony do portu PC13 był naciskany.

Uwaga! Dlaczego wartości zmierzone przez ADC zmieniają się?



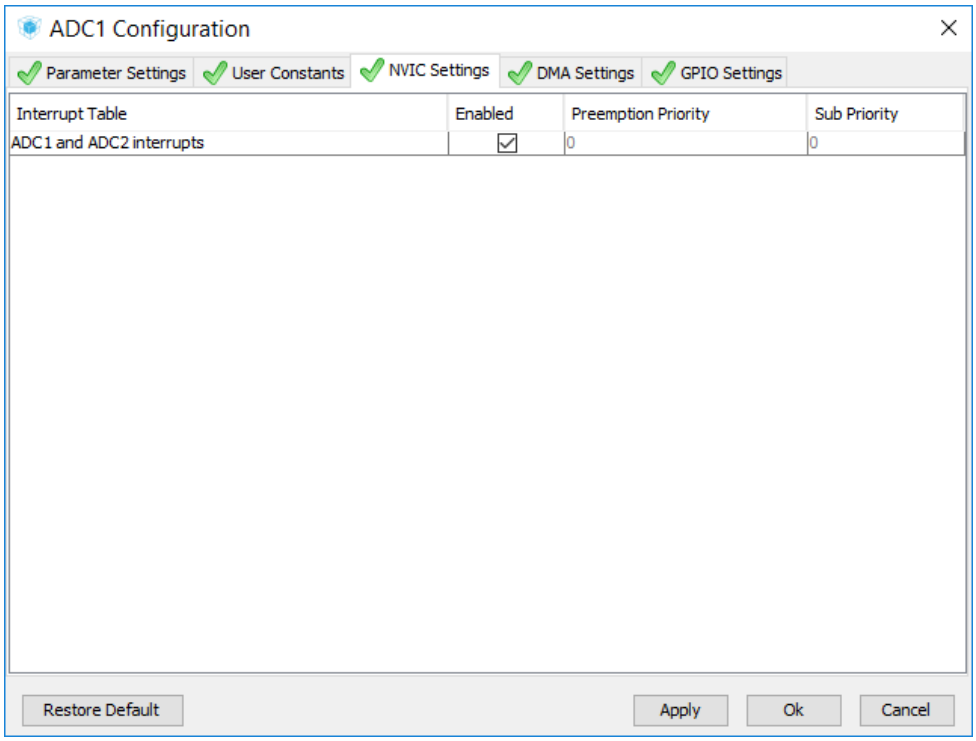
Rysunek 2: Konfiguracja parametrów przetwornika ADC1

Zadanie dodatkowe Skonfiguruj ADC w taki sposób, aby wykonywany był pomiar ciągły bez potrzeby cyklicznego wyzwalania procesu. Nie wykorzystuj przy tym przerwania od ADC, czy DMA.

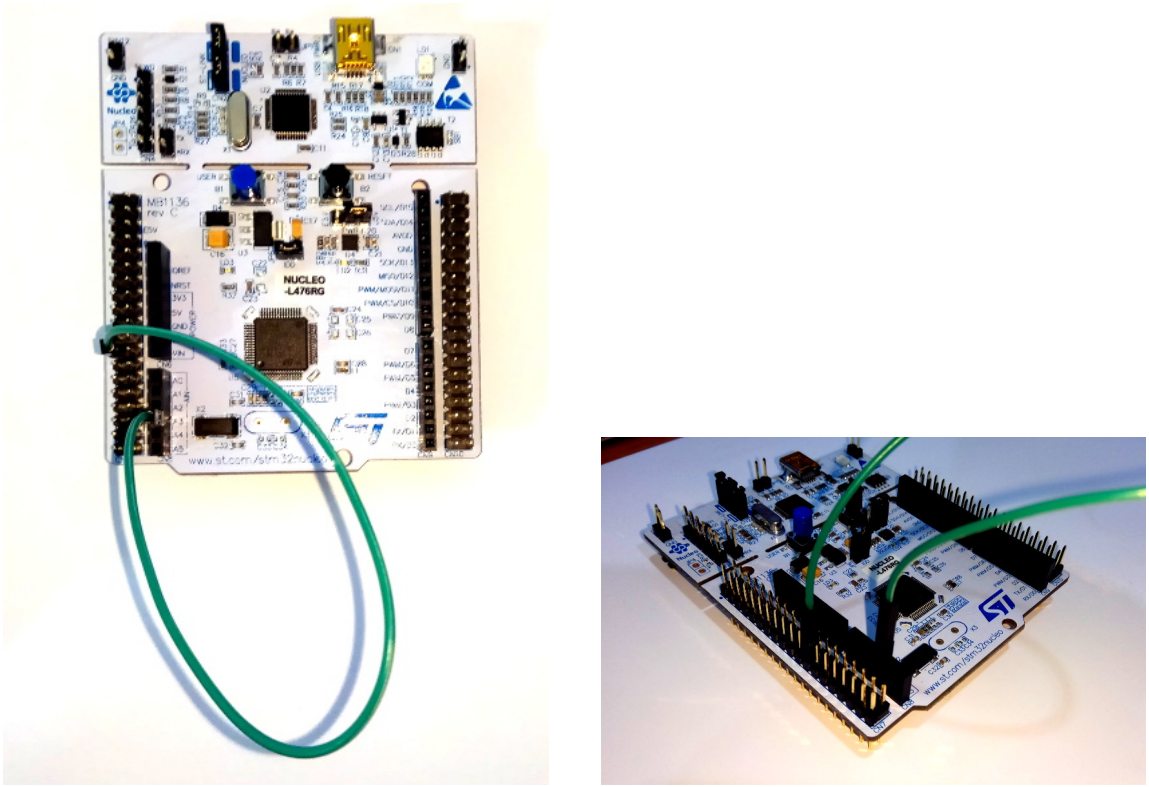
3.2 Przetwornik cyfrowo–analogowy DAC

Celem tego ćwiczenia jest zaprezentowanie działania przetwornika DAC. Aby pokazać, że owy przetwornik cyfrowo–analogowy działa poprawnie połączono wyjście tego przetwornika z kanałem ADC. Program z interwałem sekundowym generuje kolejne poziomy na wyjściu DAC, które to są od razu mierzone za pomocą przetwornika ADC. Zadana wartość na DAC, jak i wartość zmierzona na kanale ADC jest wypisywana za pomocą przekierowanej funkcji `printf()`. Ponadto, wykorzystano aplikację `STMStudio` [8], [5].

Bazując na poprzednim programie zmodyfikuj konfigurację projektu w aplikacji `STM32CubeMX`, tak, aby wykorzystywany był przetwornik DAC. Konfiguracja pinu została pokazana na rysunku 6. Podobnie, jak poprzednio na pinie PA4 wybierz tryb pracy *DAC1_OUT1*, a następnie z listy peryferiów wybierz tryb pracy pierwszego wyjścia przetwornika cyfrowo–analogowego na *Connected to external pin only.*



Rysunek 3: Konfiguracja przerwań dla ADC1

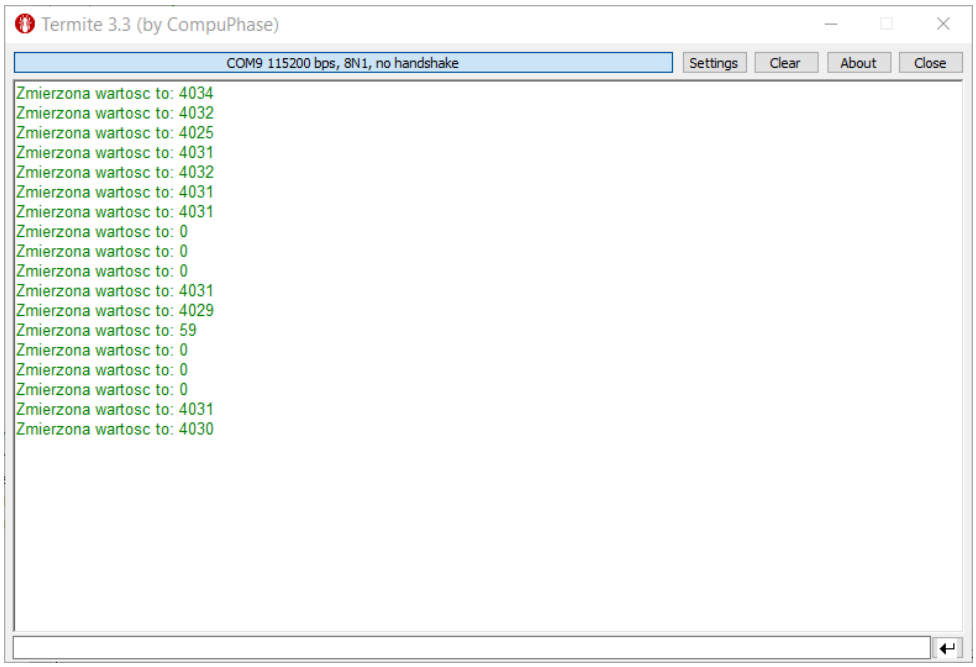


(a) (b)

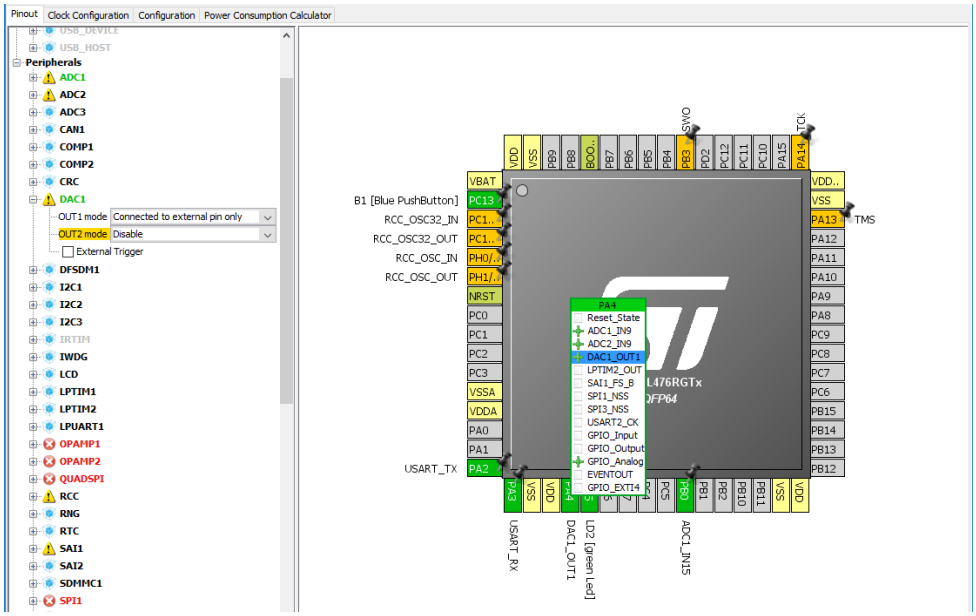
Rysunek 4: Zdjęcia przedstawiające prawidłowo podłączony przewód do płytki

Po skonfigurowaniu układu DAC przejdź do ustawień parametrów tego peryferium i upewnij się, że jest ona taka, jak na rysunku 7.

Wygeneruj kod z programu `STM32CubeMX`. W przypadku, gdy modyfikowany jest program z wcze-



Rysunek 5: Zapis sesji z portu szeregowego dla przetwornika ADC



Rysunek 6: Konfiguracja pinu dla kanału DAC1

śniejszego ćwiczenia wystarczy wprowadzić niewielkie modyfikacje. Pierwszą z nich jest zastąpienie kodu znajdującego się w nieskończonej pętli głównej programu `while(1)` następującym fragmentem:

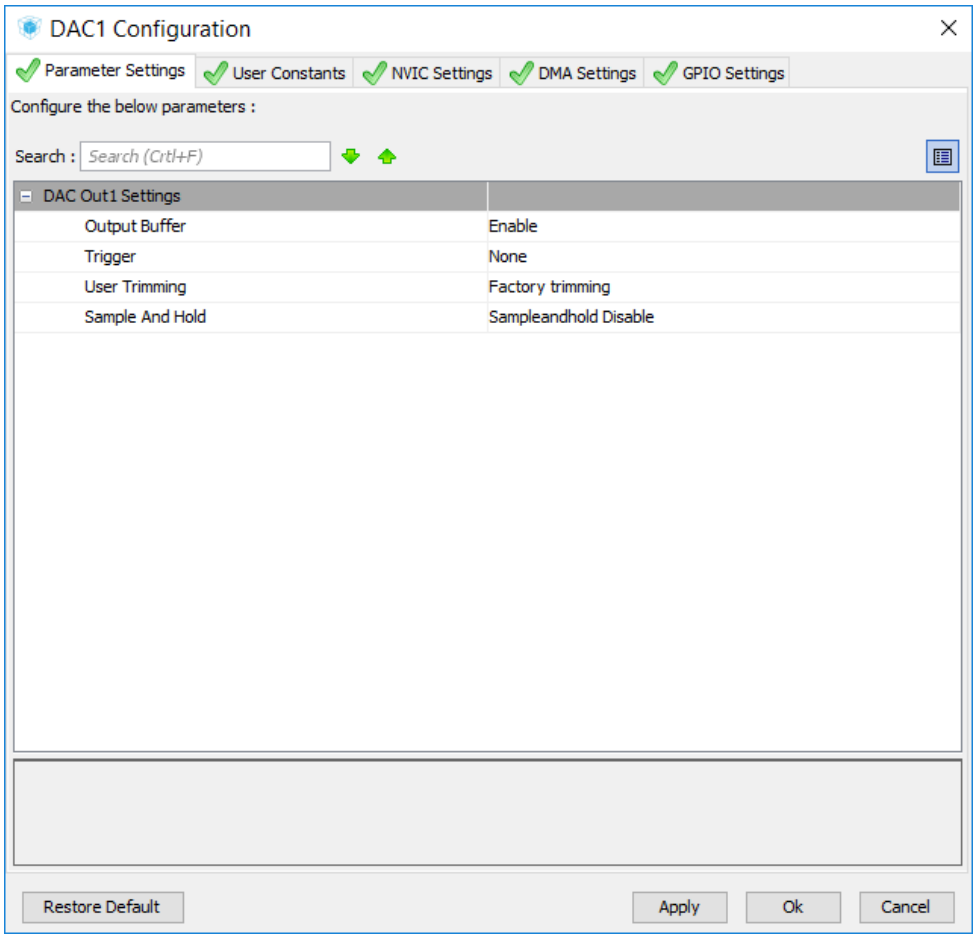
```
if (adc_flag == 1) {
  printf("Zadana warotsc to: %d\r\n", dac_value);
  printf("Zmierzona wartosc to: %d\r\n", adc_value);
  adc_flag = 0;

  dac_value += 50;
  dac_value %= 300;
  HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_8B_R, dac_value);

  HAL_ADC_Start_IT(&hadc1);
}

HAL_Delay(1000);
```

Uzupełnij program o brakujące zmienne. Pamiętaj, aby były one odpowiedniego typu i dostęp do



Rysunek 7: Konfiguracja pinu dla kanału DAC1

nich nie był optymalizowany.

Podobnie, jak poprzednio należy odpowiednio uruchomić DAC. Dodaj następujący fragment kodu:

```
1 HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_8B_R, 0);
2 HAL_DAC_Start(&hdac1, DAC_CHANNEL_1);
```

przed wywołaniem funkcji `HAL_ADC_Start_IT()` znajdującym się powyżej pętli `while(1)`. Tak przygotowany program skompiluj i wgraj do pamięci Flash mikrokontrolera. Po wgraniu programu odłącz płytkę oraz połącz przewodem porty PA4 (DAC) oraz PB0 (ADC). Prawidłowo podłączony przewód do płytki został przedstawiony na zdjęciach 8(a) i 8(b). Przed ponownym podłączeniem płytki do komputera za pomocą przewodu mini USB poinformuj prowadzącego o podłączeniu portów mikrokontrolera.

Do wizualizacji danych, poza prostym przekierowaniem funkcji `printf()`, wykorzystaj program `STMStuidio` do prezentacji zawartości zmiennych *adc_value* oraz *dac_value* w czasie (rysunek 9).

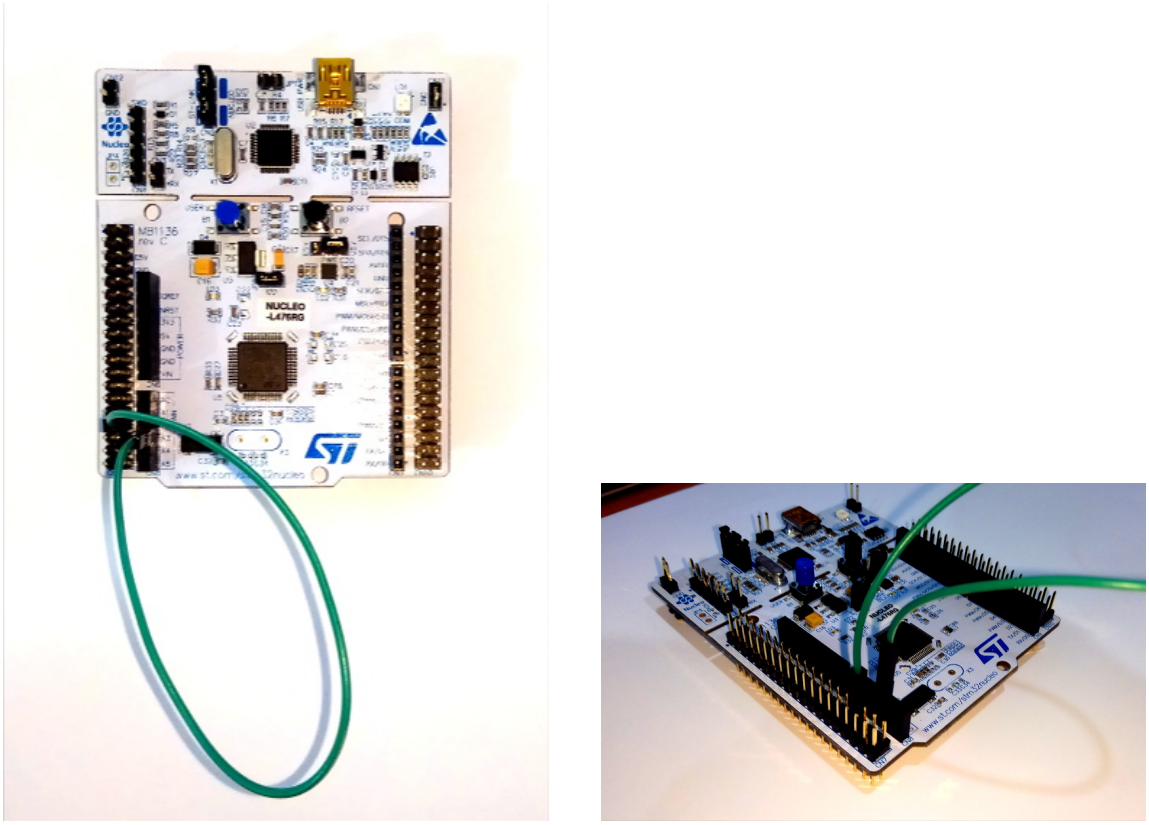
Efekt w postaci danych wypisywanych na port szeregowy można zaobserwować na rysunku 10.

Uwaga! Dlaczego wartości zadanej dla DAC odpowiada inna wartość zmierzona za pomocą ADC?

3.3 Bezpośredni dostęp do pamięci DMA

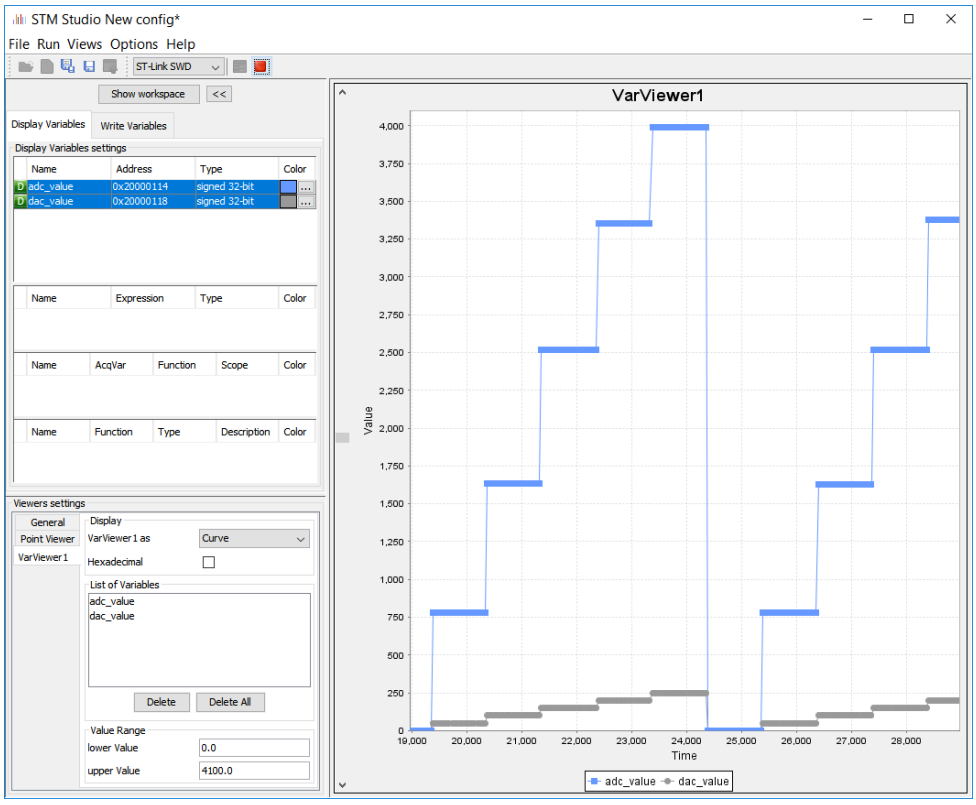
W dwóch poprzednich ćwiczeniach należało skonfigurować ADC oraz DAC. Wszelkie dane odczytywane z rejestru danych ADC były odczytywane ręcznie z udziałem procesora, podobnie nowe próbki, które miały zostać zamienione z postaci cyfrowej na analogową były zapisywane do odpowiedniego rejestru DAC ręcznie wykorzystując do tego celu rdzeń układu. Możliwe jest ominięcie udziału CPU pośredniczącego w tym procesie, a tym samym zaoszczędzenie czasu jednostki wykonawczej. Aby uzyskać zamierzony efekt należy wykorzystać zarządzanie bezpośrednim dostępem do pamięci (ang. *Direct Memory Access*, DMA).

DMA jest wyspecjalizowanym układem, który jest bezpośrednio podłączony do pamięci Flash, SRAM, magistrali APB oraz AHB [12]. Pozwala on na transfer danych bez udziału procesora w trzech różnych konfiguracjach:

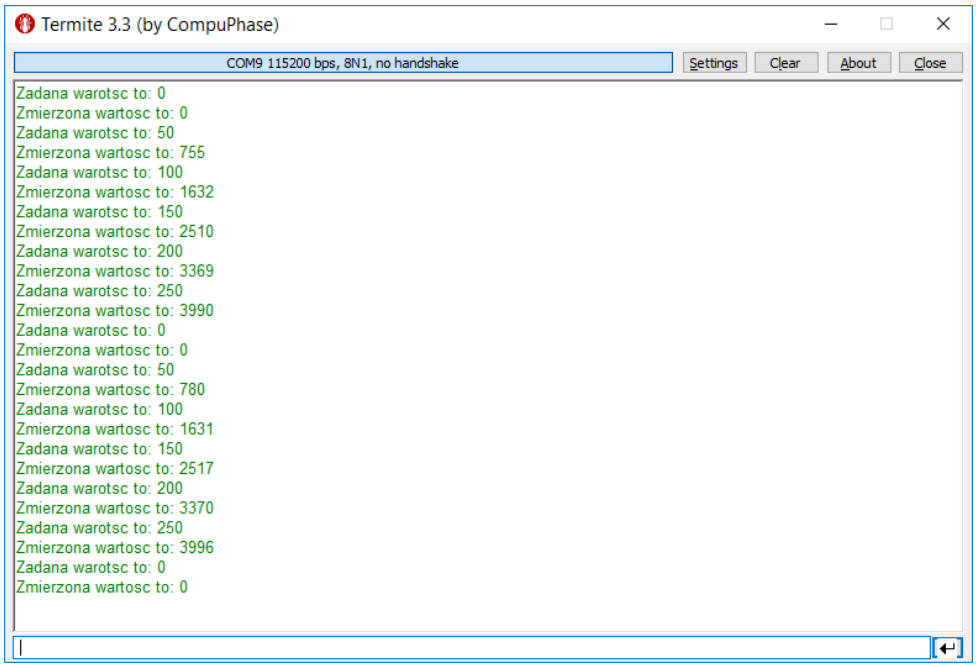


(a) (b)

Rysunek 8: Zdjęcia przedstawiające prawidłowo podłączony przewód do płytki



Rysunek 9: Wizualizacja danych w STMStudio dla przetwornika DAC



Rysunek 10: Zapis sesji z portu szeregowego dla przetwornika ADC oraz DAC

- z pamięci do pamięci,
- z peryferium do pamięci,
- z pamięci do peryferium.

Ponadto, dostarcza on przerwania, które informują system o połowicznym zakończeniu transmisji, całościowym zakończeniu transmisji, jak i o wystąpieniu błędu podczas transferu danych. Każdy z kontrolerów bezpośredniego dostępu do pamięci jest podzielony na niezależne kanały dostępu, które przypisane są do peryferiów mikrokontrolera. Jedną z dodatkowych cech DMA jest możliwość jego konfiguracji w tzw. trybie bufora cyklicznego, po którego zapełnieniu dane są nadpisywane od jego początku.

Zadaniem w tym ćwiczeniu jest skonfigurowanie DAC z wykorzystaniem DMA w trybie bufora cyklicznego wyzwalanym okresowo sygnałem z licznika. Pozwala to na zaimplementowanie prostego generatora sygnału o zadanej częstotliwości. Zasada działania jest podobna do wcześniejszego programu – dane zapisywane są do rejestru danych przetwornika DAC, a następnie mierzone jest wygenerowane napięcie za pośrednictwem ADC. Rożnica polega na tym, że dane do DAC będą automatycznie przepisywane z tablicy zawierającej wcześniej ustalone stałe wartości sygnału (tzw. *look–up table*).

Zacznij od wygenerowania tablicy przechowującej wartości sygnału sinusoidalnego w jednym okresie. Możesz do tego celu wykorzystać jedną linię programu w języku skryptowym Python:

```
print ''.join(['%d, ' % (scipy.sin(a*scipy.pi/8)*127+128) for a in range(16)])
```

lub przepisać gotową tablicę:

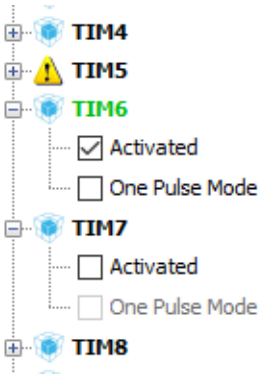
```
char dac[] = {128, 176, 217, 245, 255, 245, 217, 176, 128, 79, 38, 10, 1, 10, 38, 79, };
```

Wykorzystaj poprzedni program lub stwórz nowy projekt w `STM32CubeMX`. Aktywuj jeden z liczników (rys. 11), który będzie wyzwalaczem (ang. *trigger*) powodującym przepisanie danej z tablicy *dac* do rejestru danych przetwornika DAC.

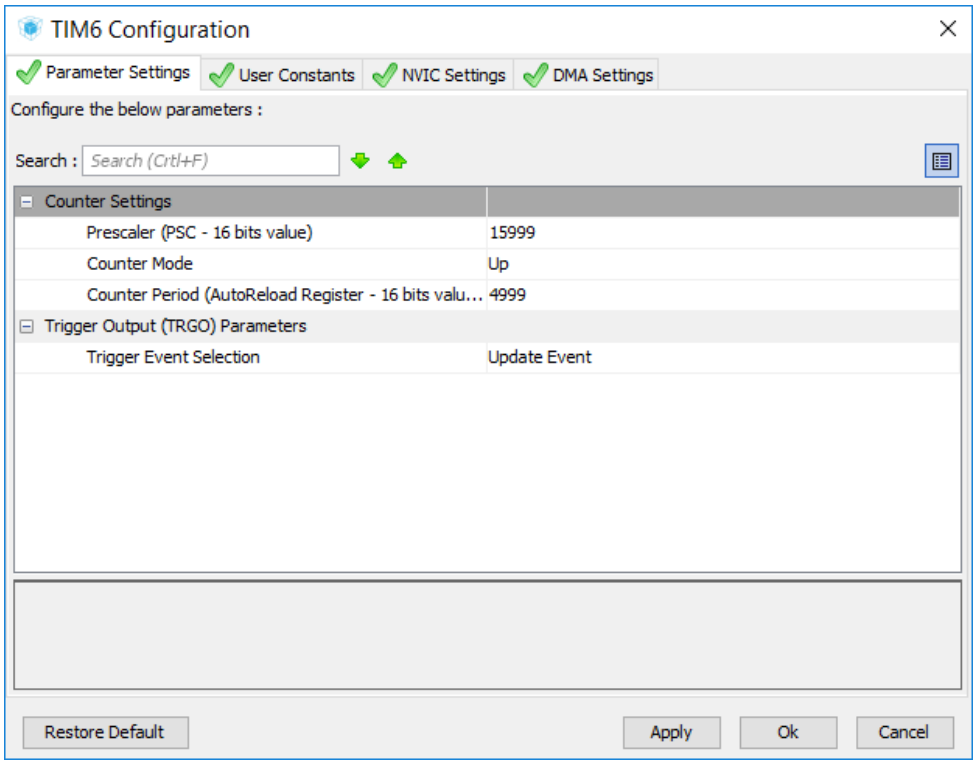
Ustaw dzielnik częstotliwości, okres licznika oraz generowany tyb zdarzenia dla licznika TIM6, tak jak zostało to pokazane na rysunku 12.

Po skonfigurowaniu pracy licznika TIM6 przejdź do konfiguracji ustawień przetwornika DAC1. Rozpocznij od zmiany parametrów, a w szczególności od zmiany źródła wyzwalacza na *timer 6 Trigger Out event*. Prawidłowo ustawione podstawowe parametry pracy przetwornika cyfrowo–analogowego zostały zaprezentowane na rysunku 13.

Kolejnym krokiem jest konfiguracja kanału DMA. W tym samym oknie przejdź do zakładki z ustawieniami DMA *DMA Settings*. Dodaj żądanie DMA za pomocą przycisku *Add*, a następnie w kolumnie *DMA Requst* kliknij na pusty rekord i wybierz pozycję *DAC_ CH1*. Skonfiguruj żądanie (ang. *request*) w



Rysunek 11: Aktywacja licznika TIM6



Rysunek 12: Aktywacja licznika TIM6

trybie pracy cyklicznej wybierając tryb (*Mode*), jako *Circular*. Ustaw automatyczną inkrementację dla pamięci. Spowoduje to, że po każdorazowym transferze danych adres wskazujący na następną komórkę pamięci, z której mają być pobrane zostanie odpowiednio zwiększony. Należy również ustalić wielkość przesyłanych danych. W tym przypadku zarówno dla peryferium, jak i pamięci wybierz pojedynczy bajt (ang. *Byte*). Zapisz ustawienia i wygeneruj kod źródłowy programu.

Uwaga! W jakim kierunku przesyłu danych został skonfigurowany kanał DMA dla DAC? Jak należało by skonfigurować żądanie DMA w przypadku, gdybyśmy chcieli odczytywać dane z ADC do pamięci?

Zastąp poprzedni kod inicjalizujący pracę peryferiów ADC i DAC przed nieskończoną pętlą `while(1)` nastepującym:

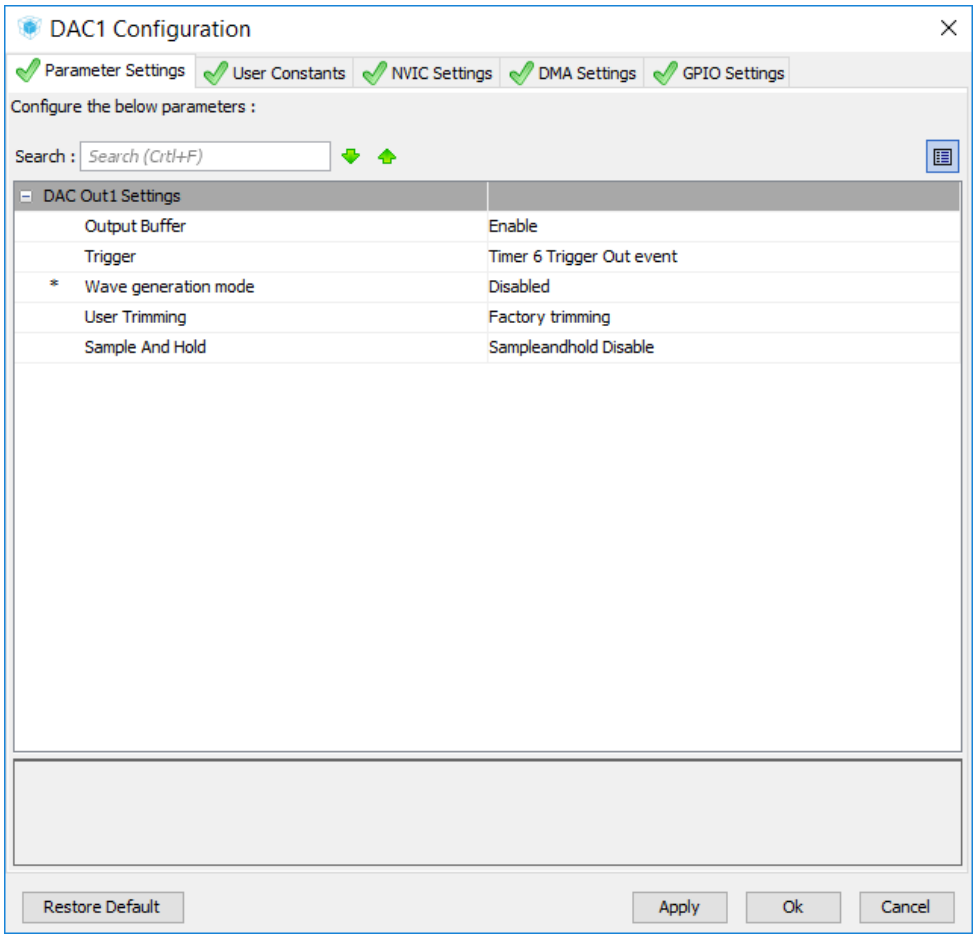
```
HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_8B_R, 0);
HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, &dac[0], 16, DAC_ALIGN_8B_R);

HAL_TIM_Base_Start(&htim6);

HAL_ADC_Start_IT(&hadc1);
```

Wewnątrz nieskończonej pętli `while(1)` umieść poniższy kod:

```
if (adc_flag == 1) {
```



Rysunek 13: Aktywacja licznika TIM6

```
2    printf("Zmierzona wartosc to: %d\r\n", adc_value);
3    adc_flag = 0;
4
5   HAL_ADC_Start_IT(&hadc1);
6  }
7
8  HAL_Delay(500);
```

Efekt w postaci danych wypisywanych na port szeregowy można zaobserwować na rysunku 14. Czy analizując zmierzone wartości za pomocą przetwornika ADC możesz zidentyfikować, jaki generowany jest sygnał?

Dodatkowo, tak, jak w poprzednim ćwiczeniu wykorzystaj program `STMStudio` do wizualizacji mierzonego napięcia przy użyciu przetwornika ADC. Przykładowy zrzut ekranu z generatorem funkcji sinus został przedstawiony na rysunku 15.

Uwaga! Z jaką częstotliwością następuje przepełnienie licznika TIM6?

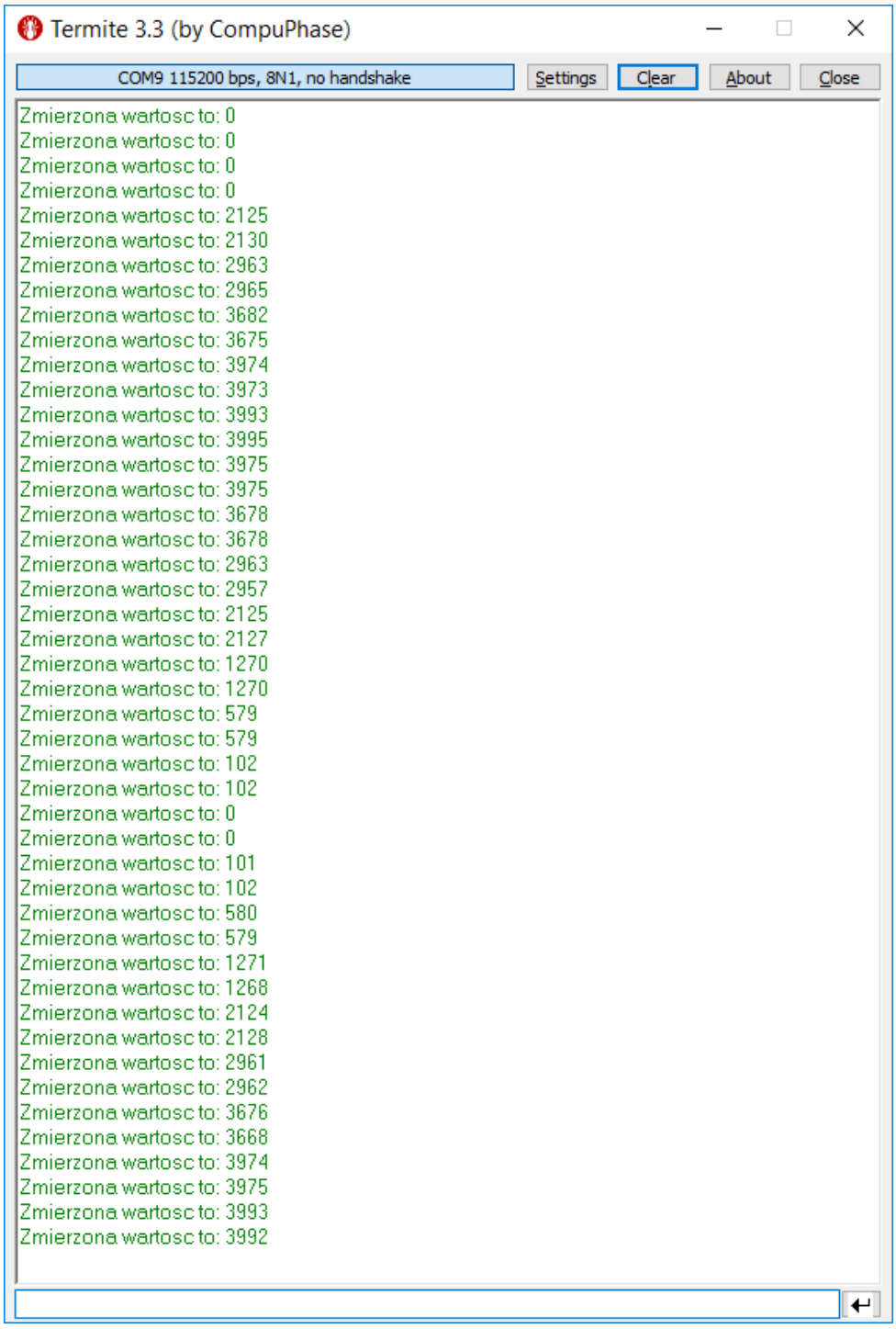
Uwaga! Jak często należy próbkować sygnał, aby sprawdzić, czy DAC rzeczywiście generuje odpowiedni przebieg?

Uwaga! Do jakich celów wykorzystywana jest tablica *look–up table*? Podaj co najmniej dwa przykłady.

Uwaga! W jaki sposób można zmieniać częstotliwość generowanego sygnału? Czy wymagane jest stworzenie nowej tablicy *look–up table*?

3.4 Uporządkowanie stanowiska

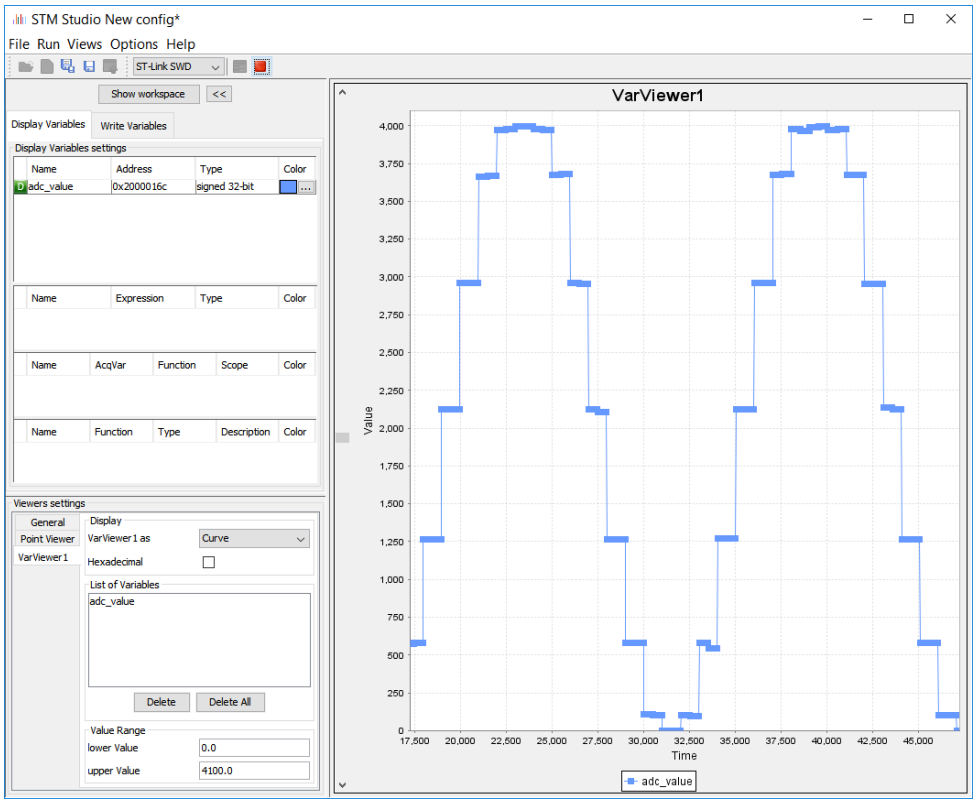
Odłóż płytkę i kabel na miejsce. Usuń projekt z AC6. Można to zrobić przez kliknięcie prawym przyciskiem myszki na projekt i wybranie opcji Usuń (Delete) z menu kontekstowego.



Rysunek 14: Zapis sesji z portu szeregowego dla przetwornika ADC oraz DAC z DMA

4 Podsumowanie

Ćwiczenie pokazuje w praktyczny sposób wykorzystanie przetwornika ADC do pomiaru napięcia. Pierwszy program wykorzystuje wyłącznie przetwornik analogowo–cyfrowy, gdzie do jednego z kanałów tego przetwornika został podłączony niebieski przycisk znajdujący się na płytce prototypowej. Drugie ćwiczenie wykorzystuje dodatkowo przetwornik cyfrowo–analogowy zamieniający zadaną wartość cyfrową na sygnał analogowy w postaci odpowiedniego poziomu napięcia. Sygnał tej jest próbkowany za pomocą przetwornika ADC. W tym ćwiczeniu wykorzystano wizualizację danych pomiarowych w czasie rzeczywistym za pomocą programu `STMStudio`. Ostatnie ćwiczenie rozszerza zakres poprzedniego o wykorzystanie kontrolera DMA dzięki, któremu w wydajny sposób można zrealizować prosty generator funkcyjny. Rów-



Rysunek 15: Wizualizacja danych w STMStudio dla przetwornika DAC z wykorzystaniem kanału DMA

nież i w tym zadaniu wykorzystano program `STMStudio` do prezentacji pomiarów.

Literatura

[1] CompuPhase. Termite: a simple RS232 terminal. `https://www.compuphase.com/software_termite.htm`.

[2] W. Domski. Sterowniki robotów, Laboratorium – Debugowanie, Zaawansowane techniki debugowania. Marzec, 2017.

[3] W. Domski. Sterowniki robotów, Laboratorium – Liczniki i przerwania, Tryby pracy licznika oraz obsługa przerwań. Marzec, 2017.

[4] W. Domski. Sterowniki robotów, Laboratorium – Wprowadzenie, Wykorzystanie narzędzi STM32CubeMX oraz SW4STM32 do budowy programu mrugającej diody z obsługą przycisku. Marzec, 2017.

[5] ST. *Getting started with STM-STUDIO, User manual.*, Październik, 2013.

[6] ST. *STM32L476xx, Ultra-low-power ARM® Cortex®-M4 32-bit MCU+FPU, 100DMIPS, up to 1MB Flash, 128 KB SRAM, USB OTG FS, LCD, analog, audio.*, Grudzień, 2015.

[7] ST. *STM32 configuration and initialization C code generation.*, Kwiecień, 2016.

[8] ST. *STM Studio run-time variables monitoring and visualization tool for STM32 microcontrollers.*, Marzec, 2016.

[9] ST. *STM32 Nucleo-64 board.*, Listopad, 2016.

[10] ST. *STM32 Nucleo-64 board, User manual.*, Listopad, 2016.

[11] ST. *STM32CubeMX for STM32 configuration and initialization C code generation, User manual.*, Marzec, 2017.

[12] ST. *STM32L4x5 and STM32L4x6 advanced ARM®-based 32-bit MCUs, Reference Manual.*, Marzec, 2017.

[13] PuTTY. Download PuTTY - a free SSH and telnet client for Windows. `http://www.putty.org/`.

