

# WSYW LAB2b

## Timery

Uwaga:

Efekty pracy każdego laboratorium powinny znaleźć się w repozytorium w oddzielnym katalogu (proszę skorzystać z gotowej struktury katalogów z repozytorium wzorcowego i nie tworzyć oddzielnych gałęzi (branchy) na poszczególne ćwiczenia). Katalog ten powinien zawierać podkatalog z kodem i podkatalog z dokumentacją, zawierający opis realizacji ćwiczenia i wyjaśnienia dotyczące implementacji kodu. Oczywiście stosowne komentarze w kodzie są również mile widziane.

Użyteczne informacje:

\* Semihosting: <https://msalamon.pl/semihosting-stm32/>

\* Dokumentacja HAL: [https://www.st.com/resource/en/user\\_manual/um1725-description-of-stm32f4-hal-and-lowlayer-drivers-stmicroelectronics.pdf](https://www.st.com/resource/en/user_manual/um1725-description-of-stm32f4-hal-and-lowlayer-drivers-stmicroelectronics.pdf)

\* Reference Manual do naszego modelu mikrokontrolera:

[https://www.st.com/resource/en/reference\\_manual/rm0383-stm32f411xce-advanced-armbased-32bit-mcus-stmicroelectronics.pdf](https://www.st.com/resource/en/reference_manual/rm0383-stm32f411xce-advanced-armbased-32bit-mcus-stmicroelectronics.pdf)

\* Obsługę przerwań implementujemy definiując odpowiednie HAL\_(...)Callback w main.c

\* Do obsługi timerów przydatne będą zarówno funkcje w języku C jak i predefiniowane makra (czyli te instrukcje zaczynające się od dwóch podkreślników \_\_HAL\_TIM\_(...)).

**Wykonanie każdego zadania należy zgłosić prowadzącemu.**

**Na koniec każdego ukończonego zadania należy wykonywać commit w git (dodając każdorazowo pliki ze zmianami do tychże commitów (pomocne będzie git add -u, tylko zawczasu trzeba dodać pliki projektowe do śledzenia zmian)). Do kodu z poprzednich zadań trzeba będzie czasem wracać na laboratoriach.**

**Na koniec zajęć zmiany w repozytorium lokalnym muszą się znaleźć również w repozytorium zdalnym na gitlabie (git push).**

Przygotowanie do zadań:

Skonfigurować semihosting tak aby można było dokonywać wydruków z STM32 w konsoli.

Zadanie 1:

Za pomocą kanałów timera w trybie Output Compare zrealizować 4 diody mrugające co sekundę, każda kolejna przesunięta względem poprzedniej o 90 stopni.

Zadanie 2:

Wykorzystać timer aby na jednej z diod uzyskać efekt łagodnego tętnienia poprzez regulację wypełnienia sygnału PWM. Kolejne wartości wypełnienia sygnału PWM proszę przysyłać w procedurze obsługi przerwania timera **z tablicy o odpowiedniej długości, wstępnie wypełnionej w funkcji main (proszę użyć <math.h> i stosownych funkcji)**. Niech kolejne wartości wypełnienia będą próbkami sinusa z przedziału połówki okresu.

### Zadanie 3:

Zaimplementować grę. Gra polega na tym, że użytkownik próbuje zmierzyć czas świecenia diody poprzez wciśnięcie przycisku. Wygrywa jeśli czas wciśnięcia przycisku jest zbliżony (z podaną dokładnością) do czasu świecenia.

Przy pomocy jednego timera z kanałem w trybie Input Capture mierzyć czas wciśnięcia przycisku.

Drugi timer ustawić na 4 sekundy i w ciągu tego czasu na losowo długi moment (ale nie dłużej niż 3 sekundy i nie krócej niż pół sekundy) zapalać diodę.

Jeśli użytkownik poprawnie odczyta czas świecenia diody z dokładnością do  $\pm 100$  ms to wygrywa. W razie potrzeby dokładność można zmniejszyć (czyli przyjąć większą tolerancję odczytu).

Żeby ułatwić sobie zadanie, niech czas świecenia diody zmienia się co rundę. Runda zaczyna się od ustalenia w losowy sposób długości czasu świecenia diody na czas trwania rundy, a kończy na prawidłowym odgadnięciu czasu świecenia diody.

Zmierzony czas wydrukować w konsoli i porównać do faktycznego czasu świecenia diody.

### Dokumentacja sprzętowa:

[https://www.st.com/resource/en/user\\_manual/um1842-discovery-kit-with-stm32f411ve-mcu-stmicroelectronics.pdf](https://www.st.com/resource/en/user_manual/um1842-discovery-kit-with-stm32f411ve-mcu-stmicroelectronics.pdf)

<https://www.st.com/resource/en/datasheet/stm32f411ve.pdf>

### Git:

git pull <adres repozytorium>

git add <nazwa pliku / katalogu>

git commit -m "<opis zmian np. Dodany projekt lab1>"

git push

**Uwaga: dopiero polecenie git push skutkuje wgraniem zmian na serwer!**

### Przydatne funkcje HAL:

HAL\_TIM\_(...)

np.:

HAL\_TIM\_Base\_Start(...)

HAL\_TIM\_Base\_Stop(...)

\_\_HAL\_TIM\_(...)

np.:

\_\_HAL\_TIM\_GET\_COMPARE(...)

Więcej funkcji w dokumentacji HAL w dziale z timerami.