WSYW LAB1

GPIO

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/
* `HAL_GetTick()` --- zwraca czas systemowy w milisekundach.
* Obsługę przerwań implementujemy definiując odpowiednie `HAL_(...)Callback` w `main.c`

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:
Do dyspozycji mamy cztery diody LED. Możemy zatem wyświetlić na nich binarnie 4-bitową liczbę. W przypadku kodowania bez bitu znaku (typy unsigned) można zapisać maksymalnie liczbę (2^4)-1 = 15 (binarnie 1111).
Zaimplementować licznik liczący od 0 do 15 (zapętalany), inkrementujący wartość co sekundę. Przyporządkować do każdego bitu jedną z diod. Jeśli wartość bitu, do którego przyporządkowana jest dioda ma wartość 1 to niech ta dioda będzie zapalona, jeśli 0 to zgaszona.

Zadanie 2:
Zmodyfikować kod z zadania 1. tak aby to wciśnięcie przycisku inkrementowało licznik. Jedno wciśnięcie przycisku powinno zawsze odpowiadać inkrementacji licznika o 1. Wydrukowywać w konsoli wartość licznika co wciśnięcie przycisku. Nie wykorzystywać przerwania.

Zadanie 3:
Zmodyfikować kod z zadania 2. tak aby skorzystać z przerwania do obsługi przycisku. Pamiętać o problemie drgania styków. Może przydać się `HAL_GetTick()` do implementacji debouncingu. Czy można go użyć w kodzie obsługi przerwania?

Dokumentacja sprzętowa:
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_Delay
HAL_GPIO_(...)

Semihosting:
https://msalamon.pl/semihosting-stm32/