Politechnika Poznańska

Wydział Automatyki, Robotyki i Elektrotechniki

Instytut Robotyki i Inteligencji Maszynowej

Zakład Sterowania i Elektroniki Przemysłowej

ZADANIE ZALICZENIOWE

Mikroprocesorowy system sterowania i pomiaru.

Systemy mikroprocesorowe

Materiały do zajęć laboratoryjnych

dr inż. Dominik Łuczak, mgr inż. Adrian Wójcik

Dominik.Luczak@put.poznan.pl
Adrian.Wojcik@put.poznan.pl

I. Cel ćwiczenia

Umiejętności

Celem zadania zaliczeniowego jest nabycie umiejętności programistycznych pozwalających na:

- implementację zamkniętej pętli regulacji dla prostego obiektu sterowania, w tym:
  - programową obsługę elementu wykonawczego,
  - programową obsługę elementu pomiarowego wraz z niezbędną obróbką sygnału pomiarowego,
  - implementację algorytmu sterowania,
- implementację interfejsu użytkownika dla mikroprocesorowego systemu sterowania i pomiaru,
- parametryzacji systemu wbudowanego,
- implementacji niezawodnych protokołów komunikacji systemu wbudowanego z systemem nadrzędnym w oparciu o komunikację szeregową,
- zaprojektowanie i implementację dedykowanych aplikacji desktopowych, skryptów i modeli symulacyjnych, stanowiących graficzny interfejs użytkownika dla systemu wbudowanego.

Kompetencji społecznych

Celem realizacji zadania zaliczeniowego jest kształtowanie właściwych postaw programistycznych:

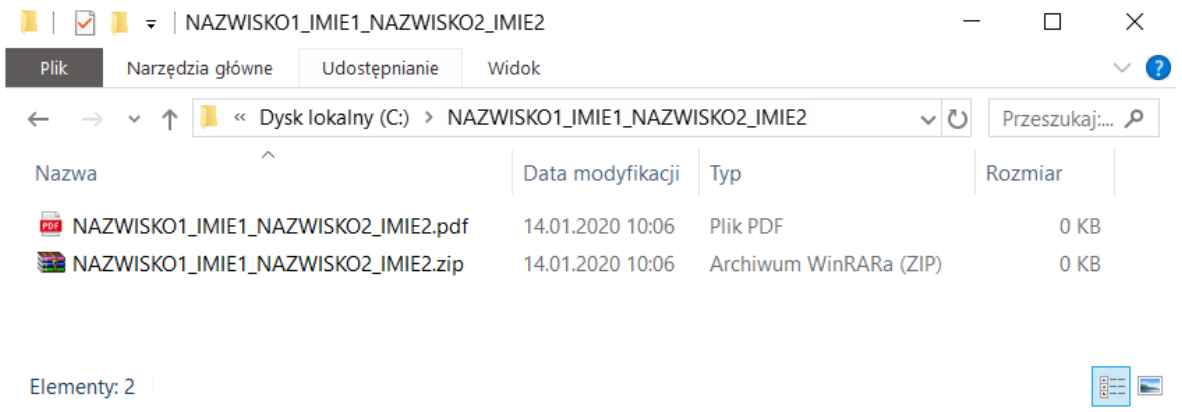
- tworzenie kodu programu w sposób czytelny,
- tworzenie własnych procedur i struktur danych zorganizowanych w niezależne moduły,
- stosowanie użytecznych dyrektyw preprocesora oraz makroinstrukcji,
- wykorzystanie systemu kontroli wersji,
- tworzenie profesjonalnej dokumentacji kodu.

II. Polecenia końcowe

W ramach zadania zaliczeniowego należy zrealizować mikroprocesorowy system sterowania i pomiaru dla *wybranego obiektu sterowania* wykorzystując jako platformę sprzętową NUCLEO-F746ZG. Zadanie należy zrealizować w **trzyosobowym zespole**. Zaliczenie obejmuje:

1. **Prezentację działania systemu prowadzącemu.**
   Prezentacja oceniana jest na podstawie liczby spełnionych wymagań.
   → **I termin**: 26 stycznia 2023
   → **II termin**: 24 lutego 2023

2. **Wykonanie raportu z realizacji zadania i przekazanie kodów źródłowych.**
   Raport oceniany jest w trzech kategoriach: *spełnienie wymogów redakcyjnych*, *poprawność merytoryczna opisu* oraz *zastosowany warsztat programistyczny*. Opis implementacji obejmować powinien zarówno część programistyczną (kody źródłowe), sprzętową (schematy elektryczne) jak i ewentualne modele symulacyjne, skrypty i aplikacje. Opisy testów powinny dokumentować prawidłowe działanie wszystkich zaimplementowanych funkcjonalności. Raport należy przesłać do systemu *eKursy* jako plik *.pdf*; pliki źródłowe (w tym kompletny projekt w środowisku STM32CubeIDE) należy przesłać jako archiwum *.zip*. Nazwać pliki wg wzorca na rys. 1.
   → **I termin**: 27 stycznia 2023
   → **II termin**: 24 lutego 2023



Rys. 1. Katalog projektowy: archiwum projektu w środowisku STM32CubeIDE oraz raport w formacie .pdf

III. Opis zadania

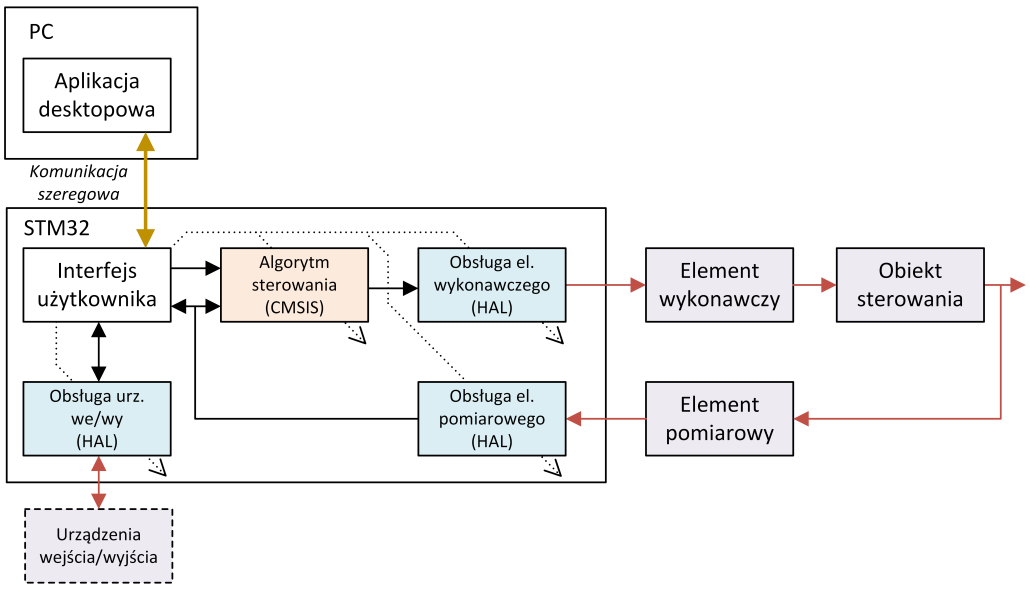
a) Zapoznanie z przepisami BHP

Wszystkie informacje dotyczące instrukcji BHP laboratorium są zamieszczone w sali laboratoryjnej oraz na stronie Zakładu [1]. Wszystkie nieścisłości należy wyjaśnić z prowadzącym laboratorium. Wymagane jest zaznajomienie i zastosowanie do regulaminu.

Na zajęcia należy przyjść przygotowanym zgodnie z tematem zajęć. Obowiązuje również materiał ze wszystkich odbytych zajęć.

b) Wstęp

Na rys. 2 zamieszczono schemat blokowy mikroprocesorowego systemu sterowania i pomiaru opartego o platformę STM32 z wykorzystaniem bibliotek HAL (obsługa urządzeń peryferyjnych) oraz CMSIS (algorytm sterowania).



Rys. 2. Schemat poglądowy mikroprocesorowego systemu sterowania i pomiaru opartego o platformę STM32 z wykorzystaniem bibliotek HAL i CMSIS.

Obsługa elementu pomiarowego: Element pomiarowy niezależnie od wybranego obiektu sterowania stanowić będzie **czujnik cyfrowy** lub **czujnik analogowy**. Programowa obsługa czujnika cyfrowego w praktyce zostanie sprowadzona do obsługi interfejsu cyfrowego (SPI, I2C, 1-wire) oraz implementacji interfejsu programowania aplikacji (ang. application programming interface, API) pozwalającego na odczyt pomiaru i konfigurację czujnika. Przykładem może być czujnik ciśnienia i temperatury BMP280 lub czujnik natężenia światła BH1750. Obsługa czujnika analogowego wymaga zaprojektowania i wykonania analogowego toru pomiarowego (z wykorzystaniem np. dzielników napięcia, wzmacniaczy oraz filtrów analogowych). Programowa obsługa czujnika analogowego sprowadza się do obsługi przetwornika ADC oraz w razie potrzeby filtracji cyfrowej. Pomiar regulowanej zmiennej może być wykonany zarówno w jednostkach fizycznych jak i w jednostkach względnych (0-100%).

Obsługa elementu wykonawczego: Element wykonawczy stanowi urządzenie, które pozwoli na zmianę wartości regulowanej zmiennej wykorzystując wyjście cyfrowe lub analogowe mikrokontrolera. Dla szerokiego wachlarza aplikacji element wykonawczy stanowić będzie tranzystor lub układ tranzystorów (np. mostek H) sterowany za pomocą sygnału PWM. Bardziej rozbudowane aktuatory (np. silniki elektryczne z dedykowanymi sterownikami) nierzadko wyposażone są w interfejsy cyfrowe: obsługa takiego elementu wykonawczego jest analogiczna do obsługi czujnika cyfrowego.

Implementacja algorytmu sterowania: Biblioteka CMSIS zawiera gotową implementację liniowego regulatora PID. Wykorzystanie tego algorytmu stanowić powinno domyślne rozwiązanie. Wykorzystanie własnej implementacji algorytmu sterowania (np. regulatora dwupołożeniowego) jest dozwolone - powinno zostać jednak uzasadnione w sprawozdaniu.

Komunikacja i interfejs użytkownika: Zrealizowany system mikroprocesorowy musi umożliwiać interakcje z użytkownikiem. W ramach zadania należy oprzeć interfejs użytkownika o komunikację szeregową. Rolą tego elementu systemu jest zapewnienie możliwości zadawania wartości referencyjnej dla układu regulacji oraz odczyt aktualnej wartości mierzonej. Interfejs powinien umożliwiać również modyfikacje parametrów systemu, np. nastaw regulatora, współczynników filtrów pomiarowych etc.

Komunikacja szeregowa powinna być odporna na podstawowe błędy i awarie. Problemy związane z fizycznym zerwaniem lub uszkodzeniem połączenia mogą być rozwiązane za pomocą implementacji *niezawodnego* protokołu komunikacji. Losowe błędy w przesłanych danych można wykryć stosując system sum kontrolnych (np. CRC).

Interfejs użytkownika może być rozbudowany o dodatkowe elementy w postaci urządzeń wejść (zadajników - potencjometry, impulsatory, klawiatury) oraz urządzeń wyjścia (wyświetlacze, diody, wyjścia analogowe).

c) Wymagania minimalne

W celu uzyskania oceny dostatecznej (50%) należy spełnić następujące wymagania minimalne:

1. System dokonuje pomiaru regulowanej zmiennej ze stałym okres próbowania.
2. System umożliwia sterowanie w bezpiecznym zakresie zmian regulowanej zmiennej.
3. System zapewnia uchyb ustalony na poziomie 5% zakresu regulacji (np. jeżeli przyjmiemy zakres regulacji temperatury jako 20-40^{o}C, uchyb ustalony w całym przedziale nie może przekraczać 1^{o}C).
4. System umożliwia zadawanie wartości referencyjnej za pomocą komunikacji szeregowej,
5. System umożliwia podgląd aktualnej wartości sygnału: *pomiarowego*, *referencyjnego* i *sterującego* za pomocą komunikacji szeregowej lub urządzenia wyjścia,

d) Wymagania dodatkowe

W celu uzyskania oceny bardzo dobrej (100%) należy spełnić wybrane 9 spośród 12 wymagań dodatkowych:

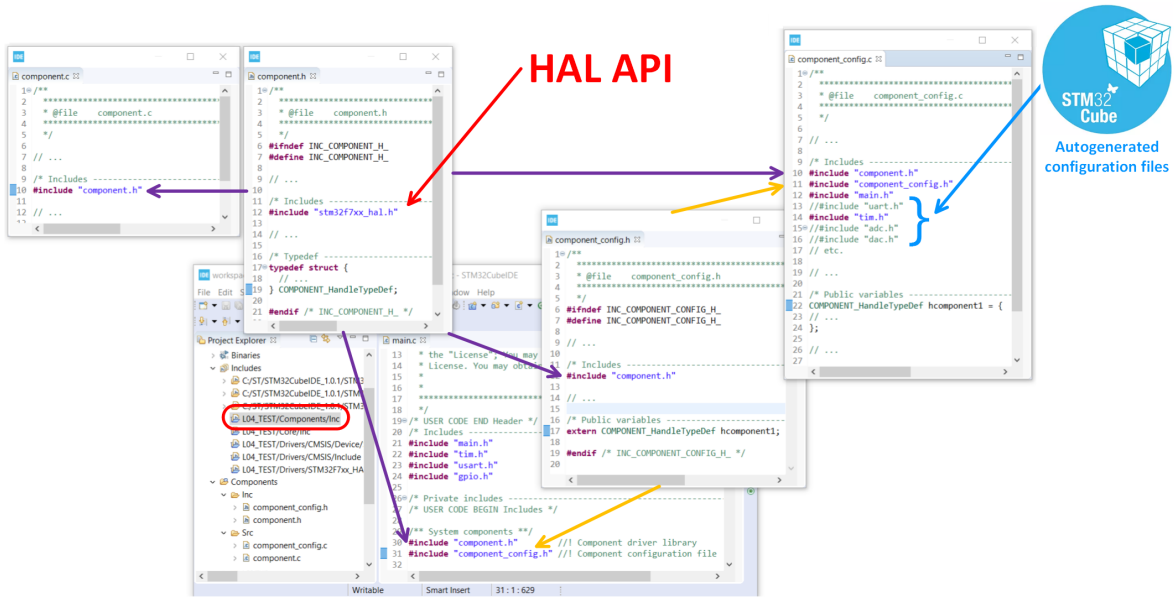
1. Podział kodu źródłowego na moduły i dokumentacja kodu zgodna ze standardem dokumentacji Doxygen. Separacja funkcjonalności i konfiguracji.
2. Wykorzystanie systemu kontroli wersji.
3. System zapewnia uchyb ustalony poziomie 1% zakresu regulacji,
4. Dedykowana aplikacja desktopowa jako graficzny interfejs użytkownika,
5. Dedykowane skrypty lub modele symulacyjne do logowania sygnałów sterujących i pomiarowych,
6. Dodatkowe urządzenie wyjścia użytkownika, np.: wyświetlacz LED lub LCD,
7. Dodatkowe urządzenie sterujące, np. 2. źródło światła w układzie regulacji oświetlenia, wentylator w układzie regulacji temperatury, etc.,
8. Dodatkowe urządzenie wejścia użytkownika, np.: enkoder, potencjometr, klawiatura numeryczna, etc,
9. Protokół komunikacji szeregowej wykorzystujący sumę kontrolną (np. CRC),
10. Wykorzystanie karty SD i systemu plików, np. do logowania danych lub przechowywania parametrów,
11. Wykorzystanie systemu czasu rzeczywistego np. FreeRTOS,
12. Wykorzystanie komunikacji sieciowej (np. złącza Ethernet i protokołu TCP) zamiast portu szeregowego.

e) System kontroli wersji

System kontroli wersji (ang. version/revision control system) – oprogramowanie służące do śledzenia zmian głównie w kodzie źródłowym oraz pomocy programistom w łączeniu zmian dokonanych w plikach przez wiele osób w różnym czasie [2]. Jednym z najpopularniejszych rozproszonych systemów kontroli wersji jest Git. System ten pozwala zapamiętać i synchronizować pomiędzy użytkownikami zmiany dokonywane na plikach źródłowych. Umożliwia przywołanie dowolnej wcześniejszej wersji kodu, a co najważniejsze, automatycznie łączy zmiany, które ze sobą nie kolidują, np. dokonane w różnych miejscach w pliku. Więcej informacji na stronie git-scm.com. Dla popularnych desktopowych systemów operacyjnych dostępne są aplikacje klienckie systemu. Istnieje kilka serwisów hostingowych umożliwiających trzymanie repozytoriów Gita, m.in Bitbucket, GitHub czy GitLab.

f) Organizacja kodu źródłowego

Prawidłowo napisany kod źródłowy powinien zawierać logiczną i konsekwentną strukturę. Na rys. 3 przedstawiono przykład organizacji kodu źródłowego użytkownika w środowisku STM32CubeIDE: każdy programowy lub sprzętowy komponent posiada parę plików .c/.h zawierające funkcjonalność komponentu (np. funkcje odczytu z sensorów) oraz parę plików .c/.h zawierających struktury konfiguracyjne komponentów (np. adresy i tryby pracy sensorów).



Rys. 3. Organizacja kodu źródłowego w środowisku STM32CubeIDE.

g) Dokumentacja kodu źródłowego

Kod źródłowy - zwłaszcza autorskie funkcje oraz struktury danych - muszą zostać opatrzone adekwatnym komentarzem zgodnym ze standardem Doxygen. Warto pamiętać, że całość biblioteki HAL jest udokumentowana zgodnie z tym standardem - pliki źródłowe biblioteki powinny więc stanowić użyteczną referencję dla tworzenia własnych komentarzy.

h) Dedykowana aplikacja desktopowa

W celu zapewnienia wygodnej obsługi systemu przez użytkownika należy zrealizować *dedykowaną* aplikację desktopową opartą o komunikację szeregową, na wybrany system operacyjny. Celem wykonania dedykowanego rozwiązania jest stworzenie graficznego interfejsu wykorzystującego widżety (pola edycji, suwaki, przyciski) ułatwiające komunikację z systemem mikroprocesorowym. Do realizacji tego zadania wykorzystać można dowolny framework programistyczny. Przykładowy *szkielet* aplikacji desktopowej napisanej w języku C# w środowisku Visual Studio dostępny jest w demonstracyjnym repozytorium.

Bibliografia

1. *Regulaminy porządkowe, instrukcje BHP* [online]. [B.d.] [udostępniono 2019-09-30]. Dostępne z: http://zsep.cie.put.poznan.pl/materialy-dydaktyczne/MD/Regulaminy-porz%C4%85dkowe-instrukcje-BHP/.
2. *System kontroli wersji* [online]. 2019 [udostępniono 2020-01-13]. Dostępne z: https://pl.wikipedia.org/w/index.php?title=System_kontroli_wersji&oldid=57005557. Page Version ID: 57005557.