(pieczęć wydziału) **KARTA PRZEDMIOTU**

1. Nazwa przedmiotu: PROGRAMOWANIE SYSTEMÓW STEROWANIA **2. Kod przedmiotu:**

3. Karta przedmiotu ważna od roku akademickiego: 2018/2019

4. Forma kształcenia: studia drugiego stopnia

5. Forma studiów: studia stacjonarne

6. Kierunek studiów: AUTOMATYKA I ROBOTYKA; WYDZIAŁ AEII

7. Profil studiów: ogólnoakademicki

8. Specjalność: Systemy pomiarowe i informacyjne

9. Semestr: 1

10. Jednostka prowadząca przedmiot: Instytut Automatyki, RAu1

11. Prowadzący przedmiot: dr hab. inż. Dariusz Bismor, prof. PŚ

12. Przynależność do grupy przedmiotów: przedmioty specjalnościowe

13. Status przedmiotu: obieralny

14. Język prowadzenia zajęć: polski

15. Przedmioty wprowadzające oraz wymagania wstępne: Zakłada się, że przed rozpoczęciem nauki niniejszego przedmiotu student posiada przygotowanie w zakresie: Programowanie obliczeń komputerowych, Programowanie obiektowe, Podstawy automatyki, regulacja PID.

16. Cel przedmiotu: Celem przedmiotu jest zapoznanie studentów z wybranymi narzędziami do programowania i symulacji układów sterowania, oraz z algorytmami najczęściej spotykanymi w systemach sterowania: podczas projektowania i symulacji, w dużych obiektach przemysłowych oraz w małych, dedykowanych sterownikach. Studenci powinni posiąść umiejętność zastosowania technik programowania orientowanego obiektowo do programowania systemów sterowania.

17. Efekty kształcenia:[1]

Nr	Opis efektu kształcenia	Metoda sprawdzenia efektu kształcenia	Forma prowadzenia zajęć	Odniesienie do efektów dla kierunku studiów
W1	Zna składnię języka C++ według standardu ISO i zawartość standardowych bibliotek.	SP,CL, RP	WM, L, P	K2A_W04, K2A_W11, K2A_W13
W2	Zna składnię s-funkcji Matlaba/Simulinka	SP,CL	WM, L	K2A_W11, K2A_W13
W3	Zna równania pozwalające na zaprogramowanie symulacji regulatora PID i GPC oraz sposoby strojenia regulatora PID	CL	WM, L	K2A_W04, K2A_W10, K2A_W16
U1	Potrafi napisać w stylu orientowanym obiektowo program implementujący techniki symulacji, identyfikacji, sterowania dyskretnego, komunikacji sieciowej	CL, RP	L, P	K2A_U01, K2A_U03, K2A_U08, K2A_U15
U2	Potrafi włączyć kod programu w języku C/C++ do środowiska symulacyjnego Matlab/Simulink	CL	L, P	K2A_U08, K2A_U19
U3	Potrafi samodzielnie skonfigurować platformę programową dla potrzeb programowania wybranego sprzętu	CL, RP	L, P	K2A_U03, K2A_U08, K2A_U25
K1	Potrafi samodzielnie dokonać podziału zadania programistycznego na współdziałające ze sobą obiekty	RP, PS	L, P	K2A_K01

[1] należy wskazać ok. 5 – 8 efektów kształcenia

K2	Rozumie zalety analizy, projektowania i programowania w stylu orientowanym obiektowo	RP,PS	L, P	K2_K04

18. Formy zajęć dydaktycznych i ich wymiar (liczba godzin)

W. : 15 L.: 15 P.: 15

Treść/tematy: Ćw./L./P./Sem.

Wykład

Wykład jest wykładem multimedialnym, bazującym na zawierających animacje slajdach. Wykładowca omawia wybrane zagadnienia wykładowe rozszerzając w razie potrzeby te zagadnienia, które najbardziej studentów interesują. Wybrane zagadnienia są dodatkowo ilustrowane demonstracjami fragmentów kodu, dokumentacji, działających programów bądź symulacji.

1. Systemy dokumentowania kodu programu. Doxygen: opcje konfiguracyjne, sposoby dokumentowania kodu, instrukcje strukturalne, formatowanie dokumentacji, odnośniki, kod HTML w dokumentacji.
2. Język C a C++. Typy rozróżniane przy przeładowaniu funkcji, etapy dopasowania. Konstruktory i destruktor, lista inicjalizacyjna. Nie-publiczny konstruktor, singletony.
3. Automatyczne i jawne konwersje typów, konstruktor jako konwerter, operator konwersji. Najczęściej przeładowywane operatory: operator przypisania, tablicowy, funkcyjny oraz odniesienia przez wskaźnik. Zręczne wskaźniki.
4. Wyjątki: specyfikacja wyjątków, wyjątki w konstruktorze i destruktorze, blok „try" na poziomie funkcji. Dziedziczenie i zawieranie klas: kolejność konstrukcji i destrukcji, spo-soby dziedziczenia, zagnieżdżona definicja klasy, dziedziczenie wielokrotne i wirtualne. Funkcje wirtualne: rzutowanie dynamiczne, funkcje czysto wirtualne i klasy abstrakcyjne.
5. Szablony: sposoby ukonkretniania, specjalizacja, szablony z wieloma parametrami, szablony a przyjaźń. Przestrzenie nazw: deklarowanie i używanie, dyrektywa a instrukcja „using", przestrzenie anonimowe.
6. Iteratory: typy, sposoby implementacji, iteratory predefiniowane. Pojemniki: typy pojemników, pojemniki standardowe, pojemnik „vector" – budowa, interfejs i wykorzystanie.
7. Pojemniki asocjacyjne: przykłady wykorzystania. Obiekty funkcyjne. Standardowe algorytmy. Klasa string: wykorzystanie i budowa wewnętrzna. Biblioteka do lokalizacji.
8. Standardowe wejście i wyjście. Strumienie. Flagi stanu formatowania. Pisanie własnych manipulatorów. Błędy i wyjątki w strumieniach. Wewnętrzna budowa strumieni, operacje nieformatowane. Strumienie plikowe i znakowe.
9. Wzorce projektowe w programowaniu orientowanym obiektowo.
10. Programowanie S-funkcji Simulinka: fazy symulacji, metody wywołania s-funkcji, s-funkcje w języku Matlaba. S-funkcje w języku C: wywołania metod w języku C, struktura „SimStruct", kompilacja s-funkcji.
11. Wejścia i wyjścia bloków Simulinka, stany, parametry, wektory robocze i czasy próbkowania bloków, detekcja przejść przez zero, raportowanie błędów. S-funkcje w języku C++.

Zajęcia laboratoryjne

Projektowanie i programowanie symulacji obiektu dyskretnego typu ARMAX.
Projektowanie interfejsu regulatora. Programowanie dyskretnej pętli symulacji.
Programowanie rekurencyjnej metody identyfikacji parametrycznej z wybraną modyfikacją.
Programowanie regulatora PID.
Programowanie regulatora predykcyjnego.
Wykorzystanie kodu zaprogramowanych regulatorów w s-funkcji Simulinka.

Zajęcia projektowe

Projekt polega na wykonaniu wybranego przez studenta w porozumieniu z prowadzącym projektu i programu w stylu orientowanym obiektowo, w języku C++ lub Java. Preferowane są programy wykonywane dla urządzeń dedykowanych, na przykład telefonów komórkowych (Symbian, Android). W projekcie wymagane jest wykorzystanie komunikacji sieciowej. W przypadku braku innego pomysłu możliwe jest rozszerzanie programu napisanego w ramach laboratorium o nowe funkcjonalności, np. komunikację sieciową, inne regulatory, itp.

21. Literatura podstawowa:

1. D. Bismor: *Programowanie systemów sterowania. Narzędzia i metody.* WNT, Warszawa, 2010
2. A. Koenig, B. Moo: *Potęga języka C++*, Helion, 2004.
3. B. Eckel: *Thinking in C++, Second Edition*, vol. 1 & 2, Upper Saddle River, NJ, 2000 & 2004.
4. S. Meyers: *Język C++ bardziej efektywny*, WNT, 1998

22. Literatura uzupełniająca:

1. M. Cline: *C++ FAQ Lite*, http://www.parashift.com/c++-faq-lite
2. D. van Heesch: *Doxygen Manual*, http://www.stack.nl/~dimitri/doxygen/manual.html
3. ISO/IEC JTC 1/SC22/WG21: *Programming Languages -- C++, Working Draft*, ISO, 2006
4. *Writing S-Functions, Version 6,* Podręcznik do Simulinka, Mathworks Inc., 2006.
5. A. Niederliński, J. Mościński, Z. Ogonowski: *Regulacja adaptacyjna*, PWN, 1995.

23. Nakład pracy studenta potrzebny do osiągnięcia efektów kształcenia

Lp.	Forma zajęć	Liczba godzin kontaktowych / pracy studenta
1	Wykład	15/10
2	Ćwiczenia	0/0
3	Laboratorium	15/10
4	Projekt	15/10
5	Seminarium	0/0
6	Inne	5/5
	Suma godzin	50/35

24. Suma wszystkich godzin: 85

25. Liczba punktów ECTS:[2] 2

26. Liczba punktów ECTS uzyskanych na zajęciach z bezpośrednim udziałem nauczyciela akademickiego: 2

27. Liczba punktów ECTS uzyskanych na zajęciach o charakterze praktycznym (laboratoria, projekty): 2

26. Uwagi:

Zatwierdzono:

…….
(data i podpis prowadzącego)

……
(*data i podpis dyrektora instytutu/kierownika katedry/ Dyrektora Kolegium Języków Obcych/kierownika lub dyrektora jednostki międzywydziałowej)*

[2] 1 punkt ECTS – 25-30 godzin.