KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Podstawowe narzędzia i metody programowania robotów autonomicznych [N2AiR1-RiSA>PNiMPRA]

Przedmiot

Kierunek studiów
Automatyka i robotyka

Rok/Semestr
1/1

Studia w zakresie (specjalność)
Roboty i systemy autonomiczne

Profil studiów
ogólnoakademicki

Poziom studiów
drugiego stopnia

Język oferowanego przedmiotu
polski

Forma studiów
niestacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
20

Laboratorium
20

Inne (np. online)
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

4,00

Koordynatorzy

dr hab. inż. Dominik Belter prof. PP
dominik.belter@put.poznan.pl

Wykładowcy

Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z podstaw robotyki i programowania. Powinien również posiadać umiejętność pozyskiwania informacji ze wskazanych źródeł oraz mieć gotowość do podjęcia współpracy w ramach zespołu

Cel przedmiotu

Przekazanie studentom wiedzy z zakresu narzędzi używanych do programowania robotów autonomicznych, poprawnego wykorzystania tych narzędzi i integracji systemów sterowania i dobierania narzędzi do rzeczywistych problemów.

Przedmiotowe efekty uczenia się

Wiedza
1. Ma zaawansowaną i pogłębioną wiedzę w zakresie metod analizy i projektowania systemów sterowania (K2_W7 [P7S_WG])
2. Ma uporządkowaną i pogłębioną wiedzę związaną z systemami sterowania i układami kontrolno-pomiarowymi (K2_W11 [P7S_WG])
3. Ma podstawową wiedzę o cyklu życia systemów automatyki i robotyki oraz układów kontrolno-

pomiarowych (K2_W13 [P7S_WG])
Umiejętności
1. Potrafi posługiwać się technikami informacyjno-komunikacyjnymi (K2_U08 [P7S_UK])
2. Potrafi zintegrować i zaprogramować specjalizowane systemy zrobotyzowane (K2_U12 [P7S_UW])
3. Potrafi dokonać krytycznej analizy sposobu funkcjonowania systemów sterowania i systemów robotyki; posiada także umiejętność doboru systemów automatyki z wykorzystaniem sterowników mikroprocesorowych (K2_U19 [P7S_UW])
4. Potrafi skonstruować algorytm rozwiązania złożonego i nietypowego zadania inżynierskiego i prostego problemu badawczego oraz zaimplementować, przetestować i uruchomić go w wybranym środowisku programistycznym dla wybranych systemów operacyjnych (K2_U25 [P7S_UW])
Kompetencje społeczne
1. Posiada świadomość konieczności profesjonalnego podejścia do zagadnień technicznych, skrupulatnego zapoznania się z dokumentacją oraz warunkami środowiskowymi, w których urządzenia i ich elementy mogą funkcjonować (K2_K4 [P7S_KR])

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:
Wiedza nabyta w ramach wykładu jest weryfikowana przez jeden 45-minutowy egzamin realizowany w sesji egzaminacyjnej. Egzamin składa się z 20-30 pytań (testowych) i do 5 pytań otwartych, różnie punktowanych. Próg zaliczeniowy: 50% punktów. Zagadnienia do egzaminu, na podstawie których opracowywane są pytania są udostępniane na wykładzie.
Umiejętności nabyte w ramach zajęć laboratoryjnych weryfikowane są podstawie kolokwium zaliczeniowego, składającego się z 20 pytań oraz sprawdzeniu realizacji zadania praktycznego realizacji problemu planowania ruchu. Próg zaliczeniowy: 50% punktów.

Treści programowe

Wykład:
1. Skrypty systemowe w bashu/python, cron, bashrc, services
2. Programowanie układu Discovery z poziomu Linuxa
3. udev rules, stałe nazwy dla urządzeń USB, low latency dla komunikacji USB
4. węzeł w ROSie do komunikacji w USB i publikujący dane
5. Przetwarzanie współbieżne w C++ (wątki, procesy)
6. CUDA (wykonywanie operacji na karcie graficznej)
7. Tensorflow + ROS (uruchomienie sieci wykrywającej obiekty w ROS)
8. Remote Master (ROS na wielu komputerach)
9 Wykrywanie obiektów ArUco do kalibracji
10 ROS bags (zebranie danych z kamery do kalibracji)
11 TFy w ROSie (odczytywanie przekształceń z wcześniej zapisanych ROS-bagów)
12 Kalibracja kamer easyHandEye (na ROS-bagach)

Laboratorium:
1. Skrypty systemowe w bashu/python, cron, bashrc, services
2. Programowanie układu Discovery z poziomu Linuxa
3. udev rules, stałe nazwy dla urządzeń USB, low latency dla komunikacji USB
4. węzeł w ROSie do komunikacji w USB i publikujący dane
5. Przetwarzanie współbieżne w C++ (wątki, procesy)
6. CUDA (wykonywanie operacji na karcie graficznej)
7. Tensorflow + ROS (uruchomienie sieci wykrywającej obiekty w ROS)
8. Remote Master (ROS na wielu komputerach)
9 Wykrywanie obiektów ArUco do kalibracji
10 ROS bags (zebranie danych z kamery do kalibracji)
11 TFy w ROSie (odczytywanie przekształceń z wcześniej zapisanych ROS-bagów)
12 Kalibracja kamer easyHandEye (na ROS-bagach)

Metody dydaktyczne

1. Wykład: prezentacja multimedialna, ilustrowana przykładami podawanymi na tablicy.
2. Ćwiczenia laboratoryjne: instrukcje realizowane na komputerach i robotach dostępnych w laboratorium

Literatura

Podstawowa
Mark Mitchell, Jeffrey Oldham, Alex Samuel, Advanced Linux Programming, New Riders Publishing
Robot Operating System (ROS), Springer 2016
Uzupełniająca
M. Galewski, STM32. Aplikacje i ćwiczenia w języku C, Wydawnictwo BTC, Legionowo 2011

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	100	4,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	40	1,50
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	60	2,50