KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Programowanie systemów sterowania w mechatronice [S2Mech1-KSUM>PSSM]

Przedmiot

Kierunek studiów
Mechatronika

Rok/Semestr
2/3

Studia w zakresie (specjalność)
Konstrukcje i sterowanie urządzeń mechatronicznych

Profil studiów
ogólnoakademicki

Poziom studiów
drugiego stopnia

Język oferowanego przedmiotu
polski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
0

Laboratorium
45

Inne
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

3,00

Koordynatorzy

dr inż. Arkadiusz Jakubowski
arkadiusz.jakubowski@put.poznan.pl

Wykładowcy

Wymagania wstępne

Podstawowa znajomość matematyki, informatyki, języków programowania. Znajomość obsługi komputera, systemu operacyjnego Windows i Linux oraz programowania w C++, C# i Python, budowanie algorytmów z wykorzystaniem elementów języka C++, C# i Python do sterowania urządzeniami mechatronicznymi. Świadomość potrzeby poszerzania wiedzy i umiejętności. Umiejętność przestrzegania zasad obowiązujących na zajęciach laboratoryjnych.

Cel przedmiotu

Poznanie podstaw programowania obiektowego, nabycie umiejętności posługiwania się klasami i strukturami. Poznanie podstaw sterowania robotami mobilnymi i ramionami robotów. Możliwość sterowania robotami w środowisku symulacyjnym. Poznanie i projektowanie oprogramowania do sterowania robotami. Autonomiczne sterowanie robotami.

Przedmiotowe efekty uczenia się

Wiedza:
Ma poszerzoną wiedzę z informatyki o znajomość systemów operacyjnych czasu rzeczywistego,

programowanie zadań współbieżnych, algorytmów przetwarzania sygnałów i sterowania, podstaw przetwarzania i analizy obrazu oraz o zasady opracowywania dokumentacji i zapewnienia jakości oprogramowania.

Umiejętności:
Potrafi wykorzystywać systemy komputerowe do projektowania i eksploatacji urządzeń mechatronicznych. Potrafi implementować układy sterowania w systemie operacyjnym czasu rzeczywistego. Umie wykorzystać podstawowe metody przetwarzania i analizy obrazu. Potrafi przygotować dokumentację oprogramowania.

Kompetencje społeczne:
Rozumie potrzebę uczenia się przez całe życie; potrafi inspirować i organizować proces uczenia się innych osób.
Potrafi ustalać priorytety służące realizacji określonego przez siebie lub innych zadania.
Potrafi współdziałać i pracować w grupie, przyjmując w niej różne role.

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Zaliczenie na podstawie sprawozdań z ćwiczeń laboratoryjnych oraz zaliczenia pisemnego składającego się z zadań programistycznych.
Skala ocen:
<51%-60%> punktów - 3.0,
(60%-70%> punktów - 3.5,
(70%-80%> punktów - 4.0,
(80%-90%> punktów - 4.5,
(90%-100%> punktów - 5.0.
Nagradzanie praktycznej wiedzy zdobytej podczas poprzednich zajęć laboratoryjnych.
Praktyczne sprawdzenie umiejętności programowania robotów.
Ocena wiedzy i umiejętności związanych z realizacją zadań indywidualnych i grupowych na laboratorium.
Zdobywanie dodatkowych punktów za aktywność na zajęciach, zwłaszcza za:
• umiejętność pracy w zespole, który praktycznie wykonuje określone zadanie,
• wykonywanie dodatkowych zadań,
• finezję rozwiązań technicznych.

Treści programowe

• Programowanie w języku C++, C# i Python.
• Instalacja i konfiguracja systemu operacyjnego Ubuntu.
• Instalacja i konfiguracja systemu ROS.
• Instalacja i konfiguracja środowiska symulacyjnego.
• Podstawy systemu operacyjnego Linux, Embedded Linux i ROS.
• Projektowanie, budowa i programowanie systemu sterowania robotami mobilnymi i ramionami robotów w środowisku ROS.
• Symulacja systemów sterowania w środowiskach symulacyjnych.
• Budowa oprogramowania do obsługi wybranych elementów systemu sterowania.
• Sterowanie autonomiczne robotami mobilnymi.

Tematyka zajęć

brak

Metody dydaktyczne

Indywidualne ćwiczenia praktyczne, wykonywanie eksperymentów, rozwiązywanie zadań, dyskusja, praca w zespole

Literatura

Podstawowa:
1. Dokumentacja ROS

2. ROS Robotics Projects, Lentin Joseph
3. Mastering ROS for Robotics Programming, Lentin Joseph, Jonathan Cacace

Uzupełniająca:
1. Python. Wprowadzenie, Mark Lutz
2. Opus magnum C++ 11. Programowanie w języku C++, Jerzy Grębosz

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	75	3,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	45	2,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	30	1,00