KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Inżynieria oprogramowania [S2Elmob1-SPE>IO1]

Przedmiot

Kierunek studiów
Elektromobilność

Rok/Semestr
1/2

Studia w zakresie (specjalność)
Systemy przetwarzania energii

Profil studiów
ogólnoakademicki

Poziom studiów
drugiego stopnia

Język oferowanego przedmiotu
polski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
30

Laboratorium
0

Inne
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

2,00

Koordynatorzy

mgr inż. Mariusz Świderski
mariusz.swiderski@put.poznan.pl

Wykładowcy

Wymagania wstępne

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

Cel przedmiotu

Przekazanie studentom podstawowej wiedzy oraz dobrych praktyk podczas tworzenia oprogramowania. Zapoznanie z popularnymi narzędziami do zarządzania projektami programistycznymi.

Przedmiotowe efekty uczenia się

Wiedza:
1. Ma poszerzoną wiedzę w zakresie technik programowania oraz stosowania nowoczesnych narzędzi informatycznych do analizy i syntezy układów elektrycznych pojazdów hybrydowych i elektrycznych w tym trakcyjnych.
2. Ma podbudowaną teoretycznie wiedzę na temat nowoczesnych metod gromadzenia, przetwarzania i analizy danych, także w zakresie stosowania uczenia maszynowego.
3. Ma rozszerzoną i usystematyzowaną wiedzę w zakresie projektowania algorytmów i programowania

mikrokontrolerów stosowanych w pojazdach oraz standardów i wykorzystania interfejsów komunikacyjnych do wymiany danych z podzespołami pojazdu.

Umiejętności:
1. Potrafi formułować i testować hipotezy związane ze złożonymi problemami inżynierskimi i prostymi problemami badawczymi z obszaru elektromobilności, a także interpretować uzyskane wyniki i wyciągać krytyczne wnioski.

Kompetencje społeczne:
1.Rozumie, że w obszarze techniki wiedza i umiejętności szybko się dewaluują co wymaga ciągłego ich uzupełniania.

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Wiedza nabyta w ramach wykładu jest weryfikowana przez jedno 45-minutowe kolokwium realizowane na ostatnim wykładzie. Kolokwium składa się z 15-20 pytań (testowych i otwartych), różnie punktowanych. Próg zaliczeniowy: 50% punktów. Zagadnienia zaliczeniowe, na podstawie których opracowywane są pytania zostaną przesłane studentom drogą mailową z wykorzystaniem systemu uczelnianej poczty elektronicznej.

Treści programowe

Paradygmaty oprogramowania, reguły projektowe, zasada komponentów, architektura oprogramowania, “Czysty kod, powstawanie projektu, detekcja i obsługa błędów,
Systemy kontroli wersji, bezpieczeństwo funkcjonalne.

Tematyka zajęć

Zagadnienia realizowane podczas laboratorium: omówienie zagadnień inżynierii oprogramowania, paradygmaty oprogramowania (programowanie: strukturalne, obiektowe i funkcyjne), reguły projektowe (SRP, OCP, Liskov substitution principle, ISP, DIP), zasada komponentów, architektura oprogramowania, “Czysty kod” (Tworzenie nazw: zmiennych, klas, metod, funkcji itp.; Komentarze; Formatowanie; Obiekty i struktury; Klasy; Systemy;), powstawanie projektu, detekcja i obsługa błędów, Unified Modeling Language (UML), systemy kontroli wersji (SVN, Git), bezpieczeństwo funkcjonalne (ISO 26262 i ASIL), automotive Software Process Improvement and Capability (ASPICE), standard MISRA-C. Tematy projektów nawiązują do treści prezentowanych w ramach wykładu oraz laboratorium. Ukierunkowane są na analizę, rozwiązanie i opracowanie wyników dla kompleksowego zagadnienia skomponowanego z kilku tematów laboratoryjnych.

Metody dydaktyczne

Wykład: prezentacja multimedialna, ilustrowana przykładami podawanymi na tablicy.

Literatura

Podstawowa:
1. K.Beck, A.Cynthia, Wydajne programowanie - Extreme Programming, Mikom, 2005.
2. A. Cockburn, Jak pisać efektywne przypadki użycia, WNT, Warszawa 2004.
3. E. Gamma, R. Helm, R. Johnson, J. Vlissides, Wzorce projektowe, Elementy oprogramowania obiektowego wielokrotnego użytku, WNT, Warszawa, 2005
4. Shalloway A.,Trott James R., Projektowanie zorientowane obiektowo. Wzorce projektowe. Gliwice, Helion, 2005
5. S.Covey, 7 nawyków skutecznego działania REBIS, 2002.
6. M.Fowler, K.Scott, UML w kropelce, LTP, 2002.
7. R. Pressman, Software Engineering, McGraw-Hill, New York 1997.

Uzupełniająca:
1. W. Humphrey, A Discipline for Software Engineering, Addison-Wesley, Reading 1995.
2. W. Humphery, Introduction to the Team Software Process, Addison-Wesley, Reading 2000.
3. Robert C. Martin, Czysty kod. Podręcznik dobrego programisty, Helion.

4. M. Świderski, A. Gąsiorek, Application of a control algorithm for the master unit of a distributed photovoltaic system, International Journal of Electronics and Telecommunications no. 4 vol. 68 2022.

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	55	2,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	30	1,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	25	1,00