

KARTA OPISU MODUŁU KSZTAŁCENIA		
Nazwa modułu/przedmiotu Systemy i aplikacje bez granic (ubiquitous)		Kod 1010511361010500089
Kierunek studiów Informatyka	Profil kształcenia (ogólnoakademicki, praktyczny) ogólnoakademicki	Rok / Semestr 3 / 6
Ścieżka obieralności/specjalność -	Przedmiot oferowany w języku: polski	Kurs (obligatoryjny/obieralny) obieralny
Stopień studiów: I stopień	Forma studiów (stacjonarna/niestacjonarna) stacjonarna	
Godziny Wykłady: 30 Ćwiczenia: - Laboratoria: 30 Projekty/seminaria: -		Liczba punktów 4
Status przedmiotu w programie studiów (podstawowy, kierunkowy, inny) kierunkowy		(ogólnouczelniany, z innego kierunku) z danego kierunku
Obszar(y) kształcenia i dziedzina(y) nauki i sztuki		Podział ECTS (liczba i %)
Odpowiedzialny za przedmiot / wykładowca:		
dr inż. Bartłomiej Prędko email: Bartlomiej.Predki@cs.put.poznan.pl tel. (61) 665-2932 Instytut Informatyki ul. Piotrowo 2, 60-965 Poznań		
Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:		
1	Wiedza:	Student rozpoczynający ten przedmiot powinien posiadać wiedzę z zakresu funkcjonowania komputera i programowania imperatywnego (zdobytą na zajęciach z przedmiotów Wprowadzenie do informatyki i Podstawy programowania) oraz wybranych elementów sieci komputerowych.
2	Umiejętności:	Powinien posiadać umiejętność rozwiązywania podstawowych problemów z zakresu implementacji i oceny kosztu działania prostych algorytmów oraz umiejętność pozyskiwania informacji ze wskazanych źródeł.
3	Kompetencje społeczne	Powinien również rozumieć konieczność poszerzania swoich kompetencji i wykazywać gotowość do podjęcia współpracy w ramach zespołu. Ponadto w zakresie kompetencji społecznych student musi prezentować takie postawy jak uczciwość, odpowiedzialność, wytrwałość, ciekawość poznawcza, kreatywność, kultura osobista, szacunek dla innych ludzi.
Cel przedmiotu:		
1. Przekazanie studentom podstawowej wiedzy na temat historii i funkcjonowania systemów powszechnych i mobilnych. 2. Pozyskanie przez studentów umiejętności projektowania i programowania systemów powszechnych, programowania i przetwarzania danych w chmurze. 3. Przekazanie wiedzy dotyczącej programowania komunikacji bezprzewodowej pomiędzy różnymi kategoriami urządzeń. 4. Kształtowanie u studentów umiejętności pracy zespołowej w trakcie realizacji projektu na zajęciach laboratoryjnych.		
Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia		
Wiedza:		
1. ma uporządkowaną, podbudowaną teoretycznie wiedzę ogólną w zakresie architektury systemów komputerowych, systemów operacyjnych - [K1st_W3] 2. ma uporządkowaną i podbudowaną teoretycznie wiedzę ogólną w zakresie kluczowych zagadnień informatyki, oraz wiedzę szczegółową w zakresie wybranych zagadnień tej dyscypliny nauki - [K1st_W4] 3. ma wiedzę o istotnych kierunkach rozwoju i najważniejszych osiągnięciach informatyki oraz innych pokrewnych dyscyplin naukowych, w szczególności elektroniki, telekomunikacji oraz automatyki i robotyki - [K1st_W5] 4. zna podstawowe techniki, metody oraz narzędzia wykorzystywane w procesie rozwiązywania zadań informatycznych, głównie o charakterze inżynierskim, z zakresu kluczowych zagadnień informatyki - [K1st_W7]		
Umiejętności:		

<p>1. pozyskiwać informacje z literatury, baz danych oraz innych źródeł (w języku ojczystym i angielskim), integrować je, dokonywać ich interpretacji i krytycznej oceny, wyciągać wnioski oraz formułować i wyczerpująco uzasadniać opinie - [K1st_U1]</p> <p>2. potrafi odpowiednio posługiwać się technikami informacyjno-komunikacyjnymi, znajdującymi zastosowanie na różnych etapach realizacji przedsięwzięć informatycznych - [K1st_U2]</p> <p>3. potrafi, formułując i rozwiązując zadania informatyczne, zastosować odpowiednio dobrane metody, w tym metody analityczne, symulacyjne lub eksperymentalne - [K1st_U4]</p> <p>4. potrafi - zgodnie z zadaną specyfikacją - zaprojektować, sformułować specyfikację funkcjonalną w formie przypadków użycia, sformułować wymagania pozafunkcyjne dla wybranych charakterystyk jakościowych) oraz zrealizować szeroko rozumiany system informatyczny, dobierając język programowania do danego zadania programistycznego oraz używając właściwych metod, technik i narzędzi - [K1st_U10]</p> <p>5. ma umiejętność formułowania algorytmów i ich implementacji z użyciem przynajmniej jednego z popularnych narzędzi - [K1st_U11]</p> <p>6. potrafi planować i realizować proces własnego permanentnego uczenia się oraz zna możliwości dalszego dokształcania się (studia II i III stopnia, studia podyplomowe, kursy i egzaminy przeprowadzane przez uczelnie, firmy i organizacje zawodowe) - [K1st_U19]</p>
Kompetencje społeczne:
<p>1. rozumie, że w informatyce wiedza i umiejętności bardzo szybko stają się przestarzałe - [K1st_K1]</p> <p>2. ma świadomość znaczenia wiedzy w rozwiązywaniu problemów inżynierskich oraz zna przykłady i rozumie przyczyny wadliwie działających systemów informatycznych, które doprowadziły do poważnych strat finansowych, społecznych lub też do poważnej utraty zdrowia, a nawet życia - [K1st_K2]</p>

<p style="text-align: center;">Sposoby sprawdzenia efektów kształcenia</p>
<p>Efekty kształcenia przedstawione wyżej weryfikowane są w następujący sposób:</p> <p>Ocena formująca:</p> <p>a) w zakresie wykładów:</p> <ul style="list-style-type: none">- na podstawie odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach; <p>b) w zakresie laboratoriów:</p> <ul style="list-style-type: none">- na podstawie oceny bieżącego postępu realizacji zadań, <p>Ocena podsumowująca:</p> <p>Sprawdzanie założonych efektów kształcenia realizowane jest przez:</p> <ul style="list-style-type: none">- ocenę umiejętności związanych z realizacją ćwiczeń laboratoryjnych,- ocenianie ciągle, na każdym zajęciach (odpowiedzi ustne) - premiowanie przyrostu umiejętności posługiwania się poznanymi zasadami i metodami,- ocenę wiedzy i umiejętności wykazanych na teście pisemnym w formie kilkunastu pytań, do zaliczenia testu potrzebne jest minimum 50% poprawnych odpowiedzi. <p>Uzyskiwanie punktów dodatkowych za aktywność podczas zajęć, a szczególnie za:</p> <ul style="list-style-type: none">- wykazanie się ciekawymi umiejętnościami ponadprogramowymi,- omówienia dodatkowych aspektów zagadnienia,- przygotowanie opracowania na określony,- efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,- umiejętność współpracy w ramach zespołu praktycznie realizującego zadanie szczegółowe w laboratorium,- uwagi związane z udoskonaleniem materiałów dydaktycznych,- wskazywanie trudności percepcyjnych studentów, umożliwiające bieżące doskonalenia procesu dydaktycznego.
<p style="text-align: center;">Treści programowe</p>

<p>W ramach wykładu przedstawiane są następujące zagadnienia:</p> <ul style="list-style-type: none"> - programowanie na platformie iOS, a w szczególności: - podstawy języków programowania Swift i Objective-C, - programowanie z wykorzystaniem różnorodnych API dla poszczególnych platform sprzętowych - iPhone, iPad, Apple Watch, - programowanie z wykorzystaniem platform uniwersalnych na przykładzie Xamarin, - korzystanie z usług chmurowych, np. Dropbox, Google Drive, iCloud, OneDrive z poziomu API aplikacji, - metodologia komunikacji urządzeń - sieci bezprzewodowe, Bluetooth, - protokoły wymiany danych, np. JSON, REST, - wstęp do chmur obliczeniowych, np. Microsoft Azure, - wykorzystanie nietypowych urządzeń, np. Microsoft Kinect, beaconów, itp. <p>W ramach ćwiczeń laboratoryjnych studenci będą mogli w praktyce przećwiczyć materiały prezentowane na wykładach w formie mini-projektów obejmujących od jednych do kilku zajęć laboratoryjnych.</p> <p>Metody dydaktyczne:</p> <ol style="list-style-type: none"> 1. Wykład: prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy, dyskusja i analiza problemów. 2. Ćwiczenia laboratoryjne: wykonywanie eksperymentów, rozwiązywanie zadań, dyskusja, praca w zespole. 		
<p>Literatura podstawowa:</p> <ol style="list-style-type: none"> 1. iOS 5 : programowanie: receptury / Vandad Nahavandipoor ; [tł.: Robert Górczyński], Helion 2013. 2. Tworzenie aplikacji na platformę iOS 5 : z wykorzystaniem Xcode, Interface Builder, Instruments, GDB oraz innych kluczowych narzędzi, Brandon Alexander, J. Bradford Dillon, Kevin Y. Kim, Helion, 2012 3. Objective-C : praktyczny podręcznik tworzenia aplikacji na systemy iOS i Mac OS X!, Stephen G. Kochan, Helion 2012 4. The Swift Programming Language 3.1, Apple Inc., 2017 5. Using Swift with Cocoa and Objective-C, Apple Inc., 2014 6. Podstawy języka Swift : programowanie aplikacji dla platformy iOS / Mark A. Lassoﬀ &#38; Tom Stachowitz, Helion 2016 7. Service design patterns : fundamental design solutions for SOAP/WSDL and RESTful Web services, Robert Daigneau, Addison-Wesley, 2012 8. Inteligentny dom : automatyzacja mieszkania za pomocą platformy Arduino, systemu Android i zwykłego komputera / Mike Riley, Helion 2013 9. Android : programowanie aplikacji / Dawn Griffiths, David Griffiths, Helion 2016 		
<p>Literatura uzupełniająca:</p>		
<p>Bilans nakładu pracy przeciętnego studenta</p>		
<p>Czynność</p>		<p>Czas (godz.)</p>
1. udział w zajęciach laboratoryjnych		30
2. przygotowanie do ćwiczeń laboratoryjnych:		5
3. dokończenie (w ramach pracy własnej) sprawozdań z ćwiczeń laboratoryjnych:		5
4. udział w konsultacjach związanych z realizacją procesu kształcenia, w szczególności ćwiczeń laboratoryjnych / projektu		3 8
5. napisanie programu / programów, uruchomienie i weryfikacja (czas poza zajęciami laboratoryjnymi)		4
6. przygotowanie do sprawdzianów / kolokwium		30
7. udział w wykładach		15
8. zapoznanie się ze wskazaną literaturą / materiałami dydaktycznymi (10 stron tekstu naukowego = 1 godz.), 150 stron		
<p>Obciążenie pracą studenta</p>		
<p>forma aktywności</p>	<p>godzin</p>	<p>ECTS</p>
Łączny nakład pracy	100	4
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	63	2
Zajęcia o charakterze praktycznym	58	2