(*faculty stamp*)

COURSE DESCRIPTION

1. Course title: GRAPHICAL PROGRAMMING	**2. Course code**
3. Validity of course description: 2016/2017	
4. Level of studies: MSc programme	
5. Mode of studies: intramural studies	
6. Field of study: MACROCOURSE	(FACULTY SYMBOL)
7. Profile of studies: general	
8. Programme: AUTOMATIC CONTROL	
9. Semester: 2	
10. Faculty teaching the course: Institute of Automatic Control, Rau1	
11. Course instructor: Witold Nocoń, P.hD., D.Sc.	
12. Course classification: programme courses	
13. Course status: elective	
14. Language of instruction: English	
15. Pre-requisite qualifications: Good understanding of computer programming in any text based programming language. It is assumed, that prior to commencing this course, students are able to implement complicated algorithms in any text based programming language using all control statements (loops and conditionals), subprograms, arrays, pointers etc. It is also assumed that students are familiar with object oriented programming concepts like encapsulation, inheritance, polymorphism.	
16. Course objectives: The goal of this course is to gain practical knowledge of and ability to design, implement and test advanced and complex software systems for automation, control, monitoring, simulation and modeling. Emphasis is put on using and modifying existing design patterns for designing scalable, reusable, easy to interpret and easy to debug software systems. The programming environment used in this course is LabVIEW	
17. Description of learning outcomes:	

Nr	Learning outcomes description	Method of assessment	Teaching methods	Learning outcomes reference code
1.	The student knows basic design patterns (state, machine, queued state machine) and synchronization mechanisms (queues, events etc.)	CL	Lecture/Laboratory	K_W03
2.	The student knows the multithreaded design patterns (master/slave, producer/consumer) and object basic oriented design patterns	CL	Lecture/Laboratory	K_W15
3.	The student can implement basic design patterns in PC and PAC environments.	CL, PS	Laboratory	K_U09
4.	The student can implement a class hierarchy for a given problem and use objects of those classes in an application.	CL, PS	Laboratory	K_U09
5.	The student is capable of selecting optimal design patterns for given problems and implement those in PC and PAC environments (including real-time environments) and SCADA systems	CL, PS	Laboratory	K_W07 K_W19 K_U23
6.	The student is capable of stating the benefits of using object-oriented solutions to programming for the given problems in PC and PAC environments.	PS, OS	Lecture	K_U13

18. Teaching modes and hours

Lecture / BA /MA Seminar / Class / Project / Laboratory

Sem 2 - 30 h., Sem 15 - 30 h

19. Syllabus description:

Semester 5 :

Lecture:

1. Fundamentals. Programming environment. Front panel and diagram. Controls and indicators. Controls and indicators terminals. Dataflow and connections on the diagram. Displaying data on a Chart. Property node.

2. Controlling execution 1. While loop. Case structure. Shift register. Iterative computations. For loop. Sequence structure. Additional methods for controlling order of execution.

3. Arrays and clusters. Creating, using and displaying arrays. Creating, using and displaying clusters. Displaying data on a Graph and XYGraph. Type definition.

4. SubVIs. Creating and using SubVIs. VI properties. Reentrant execution. Dynamically loaded VIs.

5. Local and global variables. Creating, using global/local variables. Mechanical action of buttons vs. local/global variables.

6. Object oriented programming. Encapsulation, Inheritance, Polymorphism.

7. Event based programming. Using the event structure. Signalizing and filter events. Practical rules of using the event structure. Methods for programmatic firing of events (two methods)

8. Strings, File I/O, Error handling, projects. String variables. Path variables. Binary and text files. Advanced file I/O functions. Spreadsheet functions. Measurement files. Error cluster. Error propagation. Handling errors. Creating, using projects. Building executable applications. LabVIEW run-time engine. Creating application setups.

9.Simple design patterns. Immediate VI pattern. SubVI with error handling pattern. Functional global pattern. Using a standard state machine framework. Limitations of standard state machine. Synchronization functions: First call, Occurences, Notifier functions, Queue operations,.

10. Queued state machine, Dynamic framework. Using a queued state machine framework. Limitations of queued state machine. Dynamic framework. Variant data type in dynamic framework.

11. Multithreading design patterns. Master/slave, producent/consumer.

12. Basic object-oriented design patterns. Channeling pattern, Factory pattern. Hierarchy Composition Pattern.

13. Networking and communication protocols. Shared variables, TCP, UDP, POP, SMTP etc.

14. Programmable Automation Controllers, Real-Time Targets (compactRIO for example).

15. SCADA systems.

Laboratory:

1. Fundamentals 1. Using basic programming structures, controlling execution of the program. using numerical, Boolean, string values. Programming simple mathematical computations, iterative computations etc.

2. Fundamentals 2. Using arrays, clusters. Creating and using SubVIs. Writing event-based applications.

3. Advanced task 1. Writing a data acquisition application using a DAQ device (for example a NI-USB-6008). Displaying data to the user in a configurable way. Data archive on disk.

4. Advanced task 2. Writing a Communications application using TCP/IP. Sending and receiving data over the network. Designing simple communication protocols.

5. Advanced task 3. Writing an application for control and monitoring of an industrial plant. Using for example ComactRIO controllers to read measurement data, write, control values, implement closed loop control systems, automatic and manual control, programming alarming functions etc.

20. Examination: No

21. Primary sources:

Jeffrey Travis, Jim Kring, Labview for Everyone: Graphical Programming Made Easy and Fun, Prentice Hall, 2006.
Peter Blume A., The Labview Style Book (National Instruments Virtual Instrumentation), Prentice Hall, 2007

22. Secondary sources:

Bitter R., Mohiuddin T., nawrocki M.: LabVIEW – advanced programming techniques. CRC Press, Taylor & Francis Group, 2007.

23. Total workload required to achieve learning outcomes

Lp.	Teaching mode :	Contact hours / Student workload hours
1	Lecture	30/10
2	Classes	/
3	Laboratory	15/35
4	Project	/
5	BA/ MA Seminar	/
6	Other	5/5
	Total number of hours	50/50

24. Total hours:100

25. Number of ECTS credits: 4

26. Number of ECTS credits allocated for contact hours: 2

27. Number of ECTS credits allocated for in-practice hours (laboratory classes, projects):2

26. Comments:

Approved:

…… ……

(date, Instructor's signature) (*date , the Director of the Faculty Unit signature)*