Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Department of Applied Radio Electronics

[RE-3] EMBEDDED SYSTEMS: ARM MICROCONTROLLERS ARCHITECTURE

Curriculum of the academic discipline (Syllabus)

Course details

Level of higher education	First (bachelor's)
Field of knowledge	17 - Electronics, Automation, and Electronic Communications
Specialization	172 - Electronic Communications and Radio Engineering
Educational program	All educational programs
Discipline status	Elective (F-catalog)
Form of higher education	Full-time
Year of study, semester of the discipline	Available for selection starting from the 2nd year, spring semester Scope 4 credits (Lectures 18 hours, Practical 0 hours, Lab 36 hours, Independent work 66 hours)
Semester control/control measures	Credit
Class schedule	https://schedule.kpi.ua
Language of instruction	Ukrainian / English
Information about the course coordinator / lecturers	Lecturer: V. S. Mosiychuk, Lab: S. O. Sokolsky, Independent work:V. S. Mosiychuk
Course location	https://sylabus.online/course/3-vbudovani_systemy_mikrokontrolery_arm_arkhitektury

Curriculum

1. Description of the course, its purpose, subject matter, and learning outcomes

The course is based on the **ARM Accredited MCU Engineer** (AAME) certification program for embedded systems developers. The course focuses on understanding processor architecture, particularly ARM architecture; implementing control and monitoring algorithms in embedded systems; the creation of embedded system software in assembly language and C language; and basic skills in creating embedded system software using operating systems.

Separate consideration is given to microcontroller operating modes for reducing power consumption; the concepts of interrupts and the construction of software for real-time operation.

Laboratory work covers the specifics of
configuring microcontrollers, creating projects in an automated software development environment for microcontrollers; implementing control and monitoring algorithms using peripheral devices microcontroller peripheral devices; and optimization of software according to various criteria.

The aim of the credit module is to develop students' abilities to:

- *understand processor architecture;*
- *design embedded systems based on ARM architecture microcontrollers;*
- *implement control and monitoring algorithms in embedded systems;*
- *create embedded system software in assembly language;*
- *create embedded system software in C;*
- *create embedded system software using operating systems.*

***Knowledge*:**

- the generalized structure and architecture of microcontrollers using the ARM Cortex-M0 processor as an example
- a typical set of assembly language instructions and their use in software development;
- microcontroller operating modes to reduce power consumption;
- concepts of interrupts and building software for real-time operation.

Skills:

- configure a microcontroller;
- create projects in an automated software development environment for microcontrollers;
- implement control and monitoring algorithms using microcontroller peripheral devices;
- perform software optimization according to various criteria;
- ability to perform certain actions properly, based on the appropriate use of acquired knowledge.

***Experience*:**

- design and implementation of embedded systems;
- creation of software for embedded systems.

2. Prerequisites and post-requisites of the discipline (place in the structural-logical scheme of training under the relevant educational program)

Before starting the course, it is desirable to complete the discipline "Digital Devices" or "Digital Circuitry."

3. Course content

Section 1. *Architecture of 32-bit microcontrollers*

Topic 1.1: "Features of 32-bit processor architectures and the advantages of their use as embedded systems"

Definition of embedded systems and systems on a chip; advantages of RISC architecture; features of 32-bit ARM architecture processors and microcontrollers based on them; ARM architecture processor technologies: JTAG Debug, Enhanced DSP, Vector Floating Point, EmbeddedICE, Jazelle, Long Multiply, Thrumb.

Topic 1.2: "ARM processor architecture"

Registers, stack, program and data memory; auxiliary processors and interaction with them; arithmetic logic unit; instruction pipeline; memory organization and management; processor exceptions (interrupts) and their handling; processor operating modes.

Topic 1.3: "Processor memory organization, exceptions, and interrupts in Cortex-M0"

Memory allocation; Internal Private Peripheral Bus, address ranges; byte sequence formats in memory; exceptions and interrupts, vector table; interrupt priority determination.

Topic 1.4: "Features of creating software for embedded systems"

Microcontroller initialization; polling peripheral devices in a cycle; interrupt handling; software development process; comparison of programming in assembly language and C; data types and access to peripheral devices in C; CMSIS standard;

Topic 1.5: "ARM and Thumb assembly instruction set"

Data movement commands; memory access commands; stack commands; arithmetic operation commands; logical operation commands; shift and rotation commands;

Topic 1.6: "ARM and Thumb assembly instruction set"

Unconditional and conditional jump commands; comparison commands; group data transfer commands; other instructions.

Topic 1.7: "Hardware support for operating systems"

Building software that supports the execution of a large number of tasks; SysTick timer, configuration registers; features of stack usage by the operating system; SVC and PendSV exception handling; OC Keil RTX Kernel.

Topic 1.8: "Power-saving modes of the Cortex-M0 microcontroller"

Energy-saving modes; WFE and WFI instructions; register configuration; features of developing low-power devices;

Topic 1.9: "Features of creating optimized software in C"

Rules to follow when writing optimal code; options for using profiling (automatic determination of software performance and efficiency parameters); frequency of data updates in memory, their location, and access speed; features of determining the parameters of functions, pointers, counters in cycles, and the resource intensity of various arithmetic operations.

Section 2. *Creating embedded system software for 32-bit microcontrollers*

Topic 2.1 Creating software in assembly language

Project creation, project structure, startup.s, assembler and linker directives; declaration of constants and variables, calling subroutines, initialization of the clock generator and system timer.

Topic 2.2 Debugging

Debugging programs in simulation mode, debugging programs in debug mode, viewing the contents of processor cache registers, viewing the contents of main memory, step-by-step program execution, breakpoints, searching for causes of hard_fault_exception exceptions, using technical documentation for the processor.

Topic 2.3 Creating software in C

Creating a project, project structure, startup.s, connecting libraries, CMSIS standard, using ready-made peripheral device drivers, creating your own driver.

Topic 2.4 Peripheral devices and effective work with them

Creating your own peripheral device initialization program (drivers) and working with them in the main program, initializing and processing interrupts from peripheral devices.

4. Training materials and resources

Recommended basic literature

1. Yiu J. The Definitive Guide to the ARM Cortex-M0 / J. Yiu . – Newnes, 2011. – 518 p. – ISBN: 978-0-12-385477-3.
2. Cortex-M0 r0p0 Technical Reference Manual [Electronic document]. – Access mode: http://www.arm.com/downloads/Technical Reference Manual_cortex_m0_r0p0_trm.pdf – Title from screen.

3. Cortex-M0 Devices Generic User Guide [Electronic document]. – Access mode: http://www.arm.com/downloads/Devices Generic User Guide_cortex_m0_r0p0_generic_ug.pdf – Title from screen.
4. ARMv6-M Architecture Reference Manual [Electronic document]. – Access mode: http://www.arm.com/downloads/DDI0419C_arm_architecture_v6m_reference_manual.pdf – Title from screen.
5. NUC140 Datasheet EN V3.02 [Electronic document]. – Access mode: http://www.nuvoton.com/resource-files/DA00-NUC140ENF1.pdf– Title from screen.
6. Technical Reference Manual for NUC140 [Electronic document]. – Access mode: http://www.nuvoton.com/resource-files/TRM_NUC130_NUC140(CN)_Series_EN_Rev2.05.pdf. – Title from screen.

Supporting

1. James A. Professional Embedded ARM Development / A. James
2. Wilmershurst T. Development of embedded systems using PIC microcontrollers / T. Wilmershurst; translated from English by V.N. Statsenko et al. – K.: MK-Press, 2008. – 544 p.
3. Ball S.R. Analog Interfaces for Microcontrollers / S.R. Ball. – Moscow: Dodeka, 2007. – 362 p.
4. Baker B. What a Digital Engineer Needs to Know About Analog Electronics / B. Baker; translated from English by Yu. S. Magda. – Moscow: Dodeka XXI, 2010. – 360 p. – ISBN: 978-5-94120-170-9.
5. Magda Yu. S. Programming and Debugging C/C++ Applications for ARM Microcontrollers / Yu. S. Magda. – Moscow: DMK Press, 2012. – 168 p. – ISBN: 978-5-94074-745-1.

Information resources

1. Nuvoton Technology Corporation [Electronic resource]. – Access mode: http://nuvoton.com. – Title from the screen.
2. ARM Corporation [Electronic resource]. – Access mode: http://www.arm.com. – Title from the screen.

Educational content

5. Methodology for mastering the academic discipline (educational component)

Lectures

No	Lecture topic and list of main questions (list of teaching aids, references to literature and assignments for independent study)
1	*Topic: "Embedded systems"* • *definition of embedded systems and systems on a chip;* • *advantages of RISC architecture;* • *Features of 32-bit ARM architecture processors and microcontrollers based on them;* • *ARM architecture processor technologies: JTAG Debug, Enhanced DSP, Vector Floating Point, EmbeddedICE, Jazelle, Long Multiply, Thrumb.* *Literature:* • *Cortex-M0 Technical Reference Manual, Chapter 1* *Assignment for independent study:* • *Compare ARM7TDMI and Cortex-M0 microcontrollers* • *Evaluate the performance and power consumption of the Cortex-M0 microcontroller with 8-bit microcontrollers*

2	*Topic: "ARM processor architecture"* • *registers, stack, program and data memory;* • *auxiliary processors and interaction with them;* • *arithmetic logic unit;* • *command pipeline;* • *memory organization and management;* • *processor exceptions (interrupts) and their handling* • *processor operating modes.* *References:* • *James A*. Professional Embedded ARM Development, pp. 29–51 • *Yiu J.* The Definitive Guide to the ARM Cortex-M0, pp. 34 - 51 • *Cortex-M0 Technical Reference Manual,* Chapters: 2, 3, 4, 5, 6 • *Cortex-M0 Devices Generic User Guide*, Chapter 2 *Assignment for independent study:* • *Performing arithmetic and logical operations on 32-bit signed and unsigned numbers, and determining the activation of of the N, Z, C, V status register.*
3	*Topic: "Processor memory organization, exceptions, and interrupts in Cortex-M0"* • *Memory allocation;* • *Internal Private Peripheral Bus, address ranges;* • *byte sequence formats in memory;* • *exceptions and interrupts, vector table;* • *Interrupt priority determination.* *References:* • *Cortex-M0 Technical Reference Manual,* Chapter 3 • *Cortex-M0 Devices Generic User Guide*, pp. 2-12 – 2-18 • *Yiu J.* The Definitive Guide to the ARM Cortex-M0, pp. 130 – 144, 145 – 182 *Assignment for independent study:* • *Defining and configuring the stack address and microcontroller initialization program;*
4	*Topic: "Features of creating software for embedded systems"* • *microcontroller initialization;* • *polling peripheral devices in a cycle;* • *Interrupt-driven operation;* • *Software development process;* • *comparison of programming in assembly language and C;* • *data types and access to peripheral devices in C;* • *CMSIS standard;* *References:* • *James A*. Professional Embedded ARM Development, pp. 52 - 71 • *Yiu J.* The Definitive Guide to the ARM Cortex-M0, pp. 51 - 79 *Assignment for independent study:* • *Summarize the provisions of the CMSIS standard.*
5	*Topic: "ARM and Thumb assembly instruction sets"* • *data transfer commands;* • *memory access commands;* • *stack operation commands;* • *arithmetic operation commands;* • *logic operation commands;* • *shift and rotation commands;* *References:* • *Cortex-M0 Devices Generic User Guide*, Chapter 3 • *James A*. Professional Embedded ARM Development*, pp. 121 - 132* • *Yiu J.* The Definitive Guide to the ARM Cortex-M0, pp. 80 - 129 *Assignment for independent study:* • *Provide examples of programs for each instruction.*

6	*Topic: "ARM and Thumb assembly instruction set"* • *unconditional and conditional jump commands;* • *comparison commands;* • *group data transfer commands;* • *other instructions.* *References:* • *Cortex-M0 Devices Generic User Guide*, Chapter 3 • *James A*. Professional Embedded ARM Development, pp. 132 - 143 • *Yiu J.* The Definitive Guide to the ARM Cortex-M0, pp. 80 - 129 *Assignment for independent study:* • *Provide examples of programs for each instruction.*
7	*Topic: "Hardware support for operating systems"* • *Building software that supports the execution of a large number of tasks* • *SysTick timer, configuration registers;* • *features of stack usage by the operating system;* • *processing of SVC and PendSV exceptions;* • *OC Keil RTX Kerner.* *References:* • *Yiu J.* The Definitive Guide to the ARM Cortex-M0, pp. 183–197, 331–357 *Assignment for SRC:* • *Provide examples of SVC and PendSV exception handling programs.*
8	*Topic: "Energy-saving modes of the Cortex-M0 microcontroller"* • *power saving modes;* • *WFE and WFI instructions;* • *register configuration;* • *Features of developing devices with low power consumption; References:* • *Yiu J.* The Definitive Guide to the ARM Cortex-M0, pp. 198–211, 310–329 *Assignment for independent study:* • *Provide examples of programs for configuring energy-saving modes.*
9	*Topic: "Features of creating optimized software in C"* • *Rules to follow when writing optimal code* • *options for using profiling (automatic determination of software performance and efficiency parameters);* • *frequency of data updates in memory, their placement, and access speed;* • *features of determining the parameters of functions, pointers, counters in cycles, and the resource intensity of various arithmetic operations* *References:* • *James A*. Professional Embedded ARM Development, pp. 175 - 190.

Laboratory work

The discipline "Embedded Systems" belongs to **the disciplines** in which considerable attention is paid to the practical component of training. Therefore, a larger number of classroom hours are allocated to computer practice. The main purpose of laboratory classes is to experimentally verify theoretical knowledge, acquire skills design, implementation algorithms using assembly language and C programming, debugging and verification of radio engineering device designs by simulating them on mock-ups.

No.	Name of laboratory work	Number of aud. hours
1	Creating the first project in the KEIL MDK system. Generating signals of a specified duration and period at the microcontroller outputs. Connecting to the PVV LEDs. PVV status indication.	4
2	Inputting information from the keyboard. Polling the status of buttons. Software implementation of "anti-jitter". Processing events for short and long button presses.	4
3	Dynamic indication. Methods of multifunctional use of PVV. Time separation of their use. Recoding of data from binary code to binary-decimal.	4

4	Initiation of events with a specified periodicity using interrupts due to timer overflow. Interrupt processing.	4
5	Using the UART asynchronous data transfer module. Data transfer to a PC. Displaying data in a PC terminal. Recoding data from binary code to ASCII table codes. Streaming data output to a terminal. The printf function. Implementing a dialog menu with the user in a PC terminal and on a graphic indicator.	4
6	Using the ADC module. Processing interrupts upon completion of the ADC. Forming the sampling frequency. Analog-to-digital conversion using low power modes.	4
7	PWM generation and decoding. Controlling the speed of DC motors . Controlling stepper motors.	4
8	Serial synchronous data interfaces: SPI, I2C, CAN. Recording and reading data to an SD memory card.	4
9	Installation of real-time OC (RTOS). Implementation of software in the OS.	4

All laboratory work is performed in the KEIL automated design environment on Nu-LB-NUC140 training models from NUVOTON. Each student receives an individual assignment, which they complete independently at their workstation, equipped with a personal computer and a model of an embedded system based on a microcontroller. Students receive their lab assignments in advance. Before the start of the class, a survey is conducted to assess the student's readiness to perform the work. After the work is completed, the results are defended and discussed.

Laboratory work is planned after studying the main material, as laboratory work is complex.

6. Independent work by students

No. No	Title of the topic for independent study	Number hours of ind.work
1	Topic 5. ARM and Thumb assembler instruction sets ARM assembly language instructions and examples of their use Literature: • James A. Professional Embedded ARM Development	6
2	Topic 9. Circuitry of embedded systems MCU clocking. Power supply and grounding for MCU microcircuits. Signal levels. Formation of control signals for various purposes, including for powerful drives. Coupling and interaction of embedded systems with analog blocks. Protection of radio-technical devices from impulse interference from the MCU core. Literature: • Baker B. What digital engineers need to know about analog electronics	6

Individual assignments

The curriculum for the course "Embedded Systems" includes the completion of a design project.

The main objectives of the course are to develop skills in independently managing one's own project, in particular, the development of an embedded system for a specific radio engineering device according to an individual assignment, which determines the functionality to be implemented by the microcontroller.

Organize the interaction and operation of one of the external peripheral devices specified in the individual assignment with a computer using a microcontroller (MC): ADC; DAC; RAM; flash memory; LCD display – character and graphic; LCD display – graphic; temperature sensor; acceleration sensor (accelerometer); frequency synthesizer (DDS); real-time device (RTC).

Communication with the computer is configured identically for all options via the MC's serial port using a UART module. The purpose of data exchange with the computer is to control the correct operation of the microcontroller system, or to store or display the received data, such as from sensors or ADCs.

All information necessary for performing the work regarding the organization of the interface is available in the documentation for the devices used as peripherals for the MC. The documentation (datasheet) is available for download from the manufacturers' websites on the Internet.

The following sections must be included in the calculation work:

1. Program algorithm according to the task.

- Initialization of the necessary MC modules;
- Initialization of the necessary MC interrupts according to the algorithm;
- Initialization of the external peripheral device according to the task option;
- Interrupt processing;

2. Calculation of the necessary delay time and transmission speed for the specified MC frequency.

3. Program listings with mandatory comments on commands.

4. Schematic diagram of the MCU connection with clock generator elements and auxiliary external peripheral devices.

Policy and control

7. Academic discipline policy (educational component)

Rules for attending classes (both lectures and practical/laboratory classes)

Laboratory work is mandatory. If these classes are missed, they must be made up during consultations or with other groups. If lectures are missed, tests on the material covered in the missed class must be taken and passed. Lecture materials and videos are posted on the LMS.

Defense of laboratory work

Laboratory work is defended on the day the laboratory work is completed. The student receives two grades. The first is for activity and initiative during the laboratory work and individual classes. The second is for the defense and answers to control questions.

Defense of individual assignments

As part of their independent work, students complete assignments based on lecture materials. Based on the results of the review, course participants receive comments from the instructor and a grade. Individual assignments cannot be retaken.

Incentive and penalty points and academic integrity policy

The most active students and students who complete individual assignments in an exemplary manner can receive up to 10 points towards their semester grade.

Penalty points are applied in cases where someone else's work is presented as their own, with mandatory subsequent reworking.

Deadline and resubmission policy

If the deadlines for submitting assignments are missed, the maximum score for the assignments is reduced by 10%.

8. Types of control and rating system for assessing learning outcomes

Control works

The purpose of the test is to check the quality of knowledge acquired in lectures and to monitor students' independent work, which is mainly of a reference nature, but knowledge of which is extremely important for achieving the objectives of the discipline. The test is divided into separate parts and is conducted in the form of independent work.

The main tasks included in the tests are:

1. Initialization of input/output ports (I/O) of the MC. Algorithms and programs for working with them.

2. Initialization and processing of interrupts from the main MC modules: timers, I/O, etc. Interrupt processing subroutines.

3. Methods and algorithms for forming accurate specified time intervals, their software implementation.

4. Initialization of PWM sequence formation. Algorithms and programs for forming and decoding PWM sequences.

5. Initialization of the analog-to-digital conversion (ADC) module. Algorithms and programs for implementing signal sampling. Interrupt processing. Analog-to-digital conversion in energy-saving mode. Multi-channel ADC.

6. Initialization of the synchronous serial port (SPI, I^2C) module. Receiving and transmitting data by interrupts.

7. Initialization of the universal synchronous/asynchronous receiver/transmitter (USART) module. Receiving and transmitting data by interrupts.

Grading system for assessing learning outcomes

- Lectures/webinars - 18 hours; (2 *MCR x 15 points*)
- Homework (6 assignments x 5 points)
- Laboratory work - 36 hours; (*8 labs x 5 points*)
- Calculation and design work (2 *stages x 15 points*)

Table of correspondence between rating points and university scale grades

Number of points	Grade
100-95	Excellent
94	Very good
84	Good
74-65	Satisfactory
64-60	Sufficient
Less than 60	Unsatisfactory
Admission requirements not met	Not admitted

9. Additional information on the discipline (educational component)

List of questions for the test

1. ARM-v6M processor memory organization
2. Purpose of registers LR, SP, PC.
3. Explain the purpose of the SPSR register.
4. Logic operation commands.
5. Arithmetic and logical operation commands.
6. BX, BLX commands. Their purpose.
7. LSLS, LSRS, and RORS commands. Examples of use.
8. Commands for calling and returning from subroutines.
9. Commands for working with the processor cache memory.
10. Conditional jump commands. Types of command suffixes.
11. Shift commands.
12. Examples of using direct memory access commands LDR, STR
13. Working with bits in assembly language
14. Features of working with constants.
15. Explain how the CMP command is executed. What commands are similar to the one given?
16. What is the difference between the data transfer commands ADDS R0, R1, #1 and ADDS R0, #7?
17. What do the directives #include, EQU, IMPORT, and EXPORT mean?
18. Methods of forming time delays. Examples with calculations.
19. Purpose of the stack.
20. PUSH and POP commands, examples of their use.

21. Examples of using the DTD command. Lookup tables.
22. The procedure for unlocking and locking writes to system control registers.
23. Provide analogues of while and for loops in assembly language.
24. The structure of input/output ports. Operating modes.
25. Initialization of input/output ports.
26. Clock generator settings.
27. Initialize the clock generator using the PLL frequency multiplier.
28. USART module structure.
29. Initialize the UART module.
30. Initialize the interrupt in case of TMR0 timer overflow.
31. Initialize the system timer to obtain an interrupt frequency of 1 kHz.
32. Assign a watchdog timer (WDT) and work with it.
33. Initialization of power-saving modes.
34. ARM-v6M exception and interrupt system.
35. Context saving of working registers in case of interrupt handling.
36. Provide an example of interrupt initialization and handling.
37. Interrupts from the system timer.
38. Perform interrupt initialization on input/output ports.

Description of material, technical, and information support for the discipline

A specialized microcontroller laboratory has been created for this course at the Department of Applied Radio Electronics. According to the NUVOTON academic program, the laboratory is equipped with 20 workstations with NUC140 Learning Board models with advanced peripherals and ARM Cortex-M0 NUC140VE3CN microcontroller. KEIL uVision 5 IDE software is used for laboratory work. Additional equipment, such as digital oscilloscopes and generators, is used for certain laboratory tasks.

The course syllabus (syllabus):
Compiled by V. S. Mosiychuk; S. O. Sokolsky;
Approved by the PRE Department (Minutes No. 06/2024 dated 06/27/2024)
Approved by the methodological commission of the faculty/research institute (protocol No. 06/2024 dated 28.06.2024)