



Internet of Things for Industry and Human Applications Simulation of Internet of Things Based Systems



Internet of Things for Industry and Human Applications

Simulation of Internet of Things Based Systems

PRACTICUM



**Ministry of Education and Science of Ukraine
Odessa National Polytechnic University
National Aerospace University “KhAI”
Zaporizhzhia National Technical University**

**O. V. Drozd, D. A. Maevsky, O. J. Maevskaya, O. M. Martynyuk,
G. V. Tabunshchyk, M. O. Kolisnyk, H. S. Stepova, V. S. Kharchenko,
Y. O. Chopyk, N. O. Nagachevsky, A. A. Savelev, V. V. Goroshko**

**Internet of Things for Industry and Human
Applications**

Simulation of Internet of Things based Systems

Practicum

Edited by D. A. Maevsky

Project

***ERASMUS+ ALIOT 73818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP
Internet of Thing: Emerging Curriculum for Industry and Human
Applications***

2019

UDC 004.415/.416:004.94](076.5)=111

M74

Reviewers:

Dr. Mario Fusani, ISTI-CNR, Pisa, Italy

Prof., DrS. Volodymyr Opanasenko, State Prize Winner of Ukraine, V.M. Glushkov Institute of Cybernetics of NAS, Ukraine

M74 Drozd O.V., Maevsky D.A., Maevskaya O.J., Martynyuk O.M., Tabunshchik G.V., Kolisnyk M.O., Stepova H.S., Kharchenko V.S., Chopyk Y.O., Nagachevsky N.O., Savelev A.A., Goroshko V.V. **Simulation of Internet of Things based Systems**. Practicum / Maevsky D.A. (Eds.) – Ministry of Education and Science of Ukraine, Odessa National Polytechnic University, National Aerospace University “KhAI”, Zaporizhzhia National Technical University, 2019. – 130p.

ISBN 978-617-7361-95-3

The materials of the practical part of the study course PC1 “Simulation of IoT-based systems”, developed in the framework of the ERASMUS+ ALIOT project “Modernization Internet of Things: Emerging Curriculum for Industry and Human Applications Domains” (573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP).

The structure of work on verification of residual knowledge in the discipline, the corresponding practical material, examples of tasks and criteria of evaluation are given. In the learning process, the theoretical aspects of modeling and simulation of IoT-based systems are presented. The course focuses on the assessment of the IoT based systems by use of Petri nets, Markov models and so on.

It is intended for engineers, developers and scientists engaged in the development and implementation of of IoT-based systems, for postgraduate students of universities studying in areas of IoT, cybersecurity, computer science, computer and software engineering, as well as for teachers of relevant courses.

Ref. – 63 items, figures – 40, tables – 9.

Approved by Academic Council of National Aerospace University “Kharkiv Aviation Institute” (record No 6, February 22, 2017).

ISBN 978-617-7361-95-3

© Drozd O.V., Maevsky D.A., Maevskaya O.J., Martynyuk O.M., Tabunshchik G.V., Kolisnyk M.O., Stepova H.S., Kharchenko V.S., Chopyk Y.O., Nagachevsky N.O., Savelev A.A., Goroshko V.V.

This work is subject to copyright. All rights are reserved by the authors, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms, or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar.

Міністерство освіти і науки України
Одеський національний політехнічний університет
Національний аерокосмічний університет
ім. М. Є. Жуковського «Харківський Авіаційний Інститут»
Запорізький національний технічний університет

Дрозд О.В., Маєвський Д.А., Маєвська О.Ю., Мартинюк О.М.,
Табунщик Г.В., Колісник М.О., Степова Г.С., Харченко В.С.,
Чопик Ю.О., Нагачевський Н.О., Савельєв А.А., Горошко В.В.

Інтернет речей
для
індустріальних і гуманітарних застосунків

МОДЕЛЮВАННЯ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ

Практикум

Редактор Маєвський Д. А.

Проект ERASMUS+ ALIOT
“Інтернет речей: нова освітня програма для потреб
промисловості та суспільства”
(573818-EPP-1-2016-1-UK-EPPKA2-SBHE-JP)

2019

УДК 004.415/.416:004.94](076.5)=111

М74

Рецензенти: Д-р Маріо Фузані, ISTI-CNR, Піза, Італія

Д.т.н., проф. Володимир Опанасенко, Лауреат Державної премії України, Інститут кібернетики ім. В.М.Глушкова НАН України

М74 Дрозд О.В., Маєвський Д.А., Маєвська О.Ю., Мартинюк О.М., Табунщик Г.В., Колісник М.О., Степова Г.С., Харченко В.С., Чопик Ю.О., Нагачевський Н.О., Савельєв А. А., Горошко В. В. **Моделювання систем Інтернету речей.** Практикум / За ред. Д. А. Маєвського. – Міністерство освіти та науки України, Одеський національний політехнічний університет, Національний аерокосмічний університет ім. М.Є. Жуковського «ХАІ», Запорізький національний технічний університет. – 130 с.

ISBN 978-617-7361-95-3

Практичні матеріали навчального PhD курсу “Моделювання систем Інтернету речей”, надані в цій книзі, розроблено в рамках проекту ERASMUS+ ALIOT 73818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP Internet of Thing: Emerging Curriculum for Industry and Human Applications.

Книга присвячена опису методів для міждисциплінарних досліджень з моделювання систем Інтернету речей. Проаналізовано використання методів математичного і натурального моделювання IoT систем. Надано навчальний план, опис лабораторних робіт та рекомендації щодо самостійного вивчення курсу.

Книга підготовлена для аспірантів університетів, які проводять дослідження з комп'ютерної безпеки, комп'ютерної та програмної інженерії, систем Інтернету речей. Вона може бути корисною для лекторів та викладачів, які проводять заняття на відповідних курсах.

Бібл. – 63, рисунків – 40, таблиць –

Затверджено Вченою радою Національного аерокосмічного університету «Харківський авіаційний інститут» (запис № 4, грудень 19, 2018).

ISBN 978-617-7361-95-3

© Дрозд О.В., Маєвський Д.А., Маєвська О.Ю., Мартинюк О.М., Табунщик Г.В., Колісник М.О., Степова Г.С., Харченко В.С., Чопик Ю.О., Нагачевський Н.О., Савельєв А.А., Горошко В.В.

Ця робота захищена авторським правом. Всі права зарезервовані авторами, незалежно від того, чи стосується це всього матеріалу або його частини, зокрема права на переклади на інші мови, перевидання, повторне використання ілюстрацій, декламацію, трансляцію, відтворення на мікрофільмах або будь-яким іншим фізичним способом, а також передачу, зберігання та електронну адаптацію за допомогою комп'ютерного програмного забезпечення в будь-якому вигляді, або ж аналогічним або іншим відомим способом, або ж таким, який буде розроблений в майбутньому.

ACRONIMS AND ABBREVIATIONS

AC – Alternating Current
CPN – Colored Petri Nets
CTL – Concurrent Temporal Logic
DC – Direct Current
DoS – Denial-of-Service
DDoS – Distributed Denial-of-Service
DFD – Data-flow Diagram
EGS – Evolutionary-Genetic System
ERD – Entity-Relationship Diagram
GPSS – General Purpose Simulation System
GUI – Graphical User Interface
HMM – Hidden Markov Model
HSMM - Hidden Semi-Markov Model
IDE - Integrated development environment
IoE – Internet of Everything
IoT – Internet of Things
LTL – Linear Temporal Logic
MAS – Multi-Agent Systems
MQTT – Message Queuing Telemetry Transport
MTBF - Mean Time Between Failures
MTTF - Mean Time to Failure
MTTR - Mean Time to Recover
NLP - Natural Language Processing
OPC – Open Platform Communications
OPC UA – Open Platform Communications Unified Architecture
QS – Queuing System
REST – REpresentational State Transfer

RUL - Remaining Useful Lifetime

SA - Service Available

SMR - Service May Recover

SMNR - Service May Not Recover

SNA - Service Not Available

TINA – Time Petri Net Analyzer

TTF - Time to Failure

TTR - Time to Recover

UML – Universal Modeling Language

UML-CSAS – UML-diagrams of Communications, Sequences,
Activity, States

XML – eXtensible Markup Language

INTRODUCTION

The manual presents the materials of the practical part of the training course "Simulation of IoT-based Systems" (PC1) prepared by the project ERASMUS+ ALIOT 73818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP "Internet of Thing: Emerging Curriculum for Industry and Human Applications"¹.

The manual provides a description of practical work, the general purpose of which is to get acquainted with the modeling and simulation of Internet of things devices, in particular, smart home systems.

The first part of the manual contains laboratory work on the course "Interaction Simulation for IoT Systems". The purpose of these guidelines is to consolidate and supplement the lecture material, as well as develop students' skills and abilities to understand existing IoT technologies and apply them to specific scenarios, as well as the ability to design complete IoT systems.

The second part describes the laboratory work on the course "Program Tools for the Smart Systems Simulation". The common goal of this course is to give students the skills to simulate and simulate Internet of Things devices based on the Arduino microprocessors. In the first paper, we consider the general principles of modeling and simulation in the Proteus program and the design of the laboratory stand "Smart Home". The second work is devoted to the simulation of the light sensor. The third work discusses the principles of operation and simulation of a smart home system with a motion sensor. For convenience, figures, tables and formulas are numbered within each work.

The labs of third part are dedicated to three-level simulation of IoT/IoE based systems. The laboratory work 1 allowed to study the capabilities of the structural-functional design of the IoT system based on the use of the universal modeling language UML in the StarUML Tool environment. Laboratory work 2 aimed at studying

¹ *The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.*

the possibilities of the structural-functional IoT system design with the use of Petri nets in the CPN Tools environment. Laboratory work 3 sets knowledge and develops skills in structural-functional, behavioral and event-time design of the IoT system using the LTL temporal logic in the SPIN / XSPIN environment.

Forth part of book describes labs on application of techniques and tools for Markov's modeling and simulation of IoT systems.

The course is intended for engineers engaged in the development and implementation of the Internet of Things technologies, for masters and graduate students of universities studying in the IoT systems, as well as for teachers of relevant courses.

The authors are grateful to the reviewers, project colleagues, staff of the departments of academic universities, industrial partners for valuable information, methodological assistance and constructive suggestions that were expressed during the discussion of the course program and the materials of the manual.

1 INTERACTION SIMULATION FOR IOT SYSTEMS

Laboratory work 1

Interaction simulation for web-oriented systems

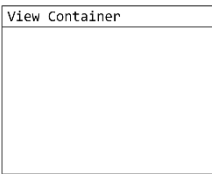

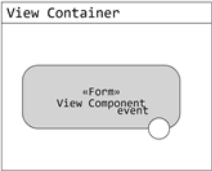
Objective: obtain practical skills in implementation of the IFML for the interaction simulation

1 Theoretical information

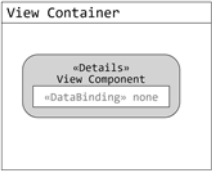
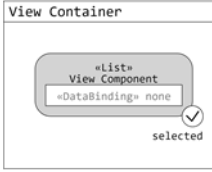
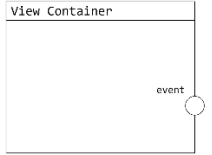
The standard Interaction Flow Modeling Language (IFML) is designed for expressing the content, user interaction and control behaviour of the front-end of software applications [1-7].

The main components of the IFML are in the Table 1.

Table 1. Main IFML components

1		View container
2		Action - a piece of business logic triggered by an event; it can be server side (the default) or client-side, denoted as [Client].
3		View component - an element of the interface that displays content or accepts input. Examples: an HTML list, a JavaScript image gallery, an input form. Examples: a database update, the sending of an email, the spell checking of a text.

Interaction Simulation for IoT Systems

4		<p>Parameters - a typed and named value.</p> <p>DetailsView Component used to display details of a specific DataBinding instance.</p>
5		<p>List View Component - ViewComponent used to display a list of DataBinding instances.</p>
		<p>Event</p>

Internet application (RIA) can be modeled as having one top-level container, the main window; instead, a Web application can be modeled as having multiple top-containers, one for every dynamic page template. Each view container can be internally structured in a hierarchy of sub-containers.

A view container can contain view components, which denote the publication of content or interface elements for data entry (e.g., input forms). A view component can have input and output parameters.

A view container and a view component can be associated with events, to denote that they support the user's interaction. Events in concrete are rendered as interactors, which depend on the specific platform and therefore are not modeled in IFML but produced by the PIM to Platform-Specific Model (PSM) transformation rules.

The effect of an event is represented by an interaction flowconnection, which connects the event to the view container or component affected by the event.

An event can also cause the triggering of an action, which is executed prior to updating the state of the user interface; for example,

in a web content management application the user can select from a list the elements to delete; the selection event triggers a delete action, after which the page with the list is redisplayed.

An input-output dependency between view elements (view containers and view components) or between view elements and actions is denoted by parameter bindings associated with navigation flows (interaction flows for navigating between view elements).

2 The task workflow

1. Select one of the application field: connected cars, e-health, smart campus.
2. Create the prototype of the web-application.
3. Develop the model of the user interaction with the web-application.
4. Create the minimum viable application in Django, Python and Nginx.
5. Upload to the github.

3 The content of the report

The content of the report is as follows:

1. Title.
2. The requirements to the system.
3. Prototype of the application.
4. Model of the designed system.
5. Link to the git-hub with the web application
6. Answers to the self-assessment questions according to the individual task.

4 Self-assessment questions

1. What are the main components of the IFML?
2. Give examples of the interaction between IFML, BPMN, UML.
3. Is IFML integrated into BPMN?
4. Which syntax is used in the IFML?

Laboratory work 2

Interaction simulation with mobile applications

Objective: obtain the skills of simulation of the interaction with mobile applications for connected cars.

1 Theoretical information

What the internet of things introduces -- or better said, unites -- is data, inter-actions and the physical world. Interaction coupled with data transcends from laptop and mobile devices, and it becomes literally embedded into any object, infrastructure or interaction.

There are several technologies which are developing in the field of connected cars: Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Device (V2D), Vehicle-to-Pedestrian (V2P), Vehicle-to-Home (V2H), Vehicle-to-Device (V2D), Vehicle-to-Grid (V2G) solutions **Error! Reference source not found.****Error! Reference source not found.**

Vehicle to Home (V2H) capability describes a scenario in which a plug-in electric vehicle (PEV) not only receives charge from the grid to power the vehicle, but also provides backup power to an islanded load such as a home during an outage, similar to a stand-alone emergency generator **Error! Reference source not found.**

Vehicle-to-grid (V2G) describes a system in which plug-in electric vehicles, such as electric cars (BEV) and plug-in hybrids (PHEV), communicate with the power grid to sell demand response services by either returning electricity to the grid or by throttling their charging rate.

V2I Applications could be grouped in the following classes:

1. Safety applications - warning for hazardous situations (such as congestions, accidents, obstacles etc.), merging assistance, intersection safety, speed management, rail crossing operations, priority assignment for emergency vehicles.
2. The efficiency applications - traffic jam notification, prior recognition of potential traffic jams, dynamic traffic light control, dynamic traffic control, connected navigation

3. The efficiency applications - traffic jam notification, prior recognition of potential traffic jams, dynamic traffic light control, dynamic traffic control, connected navigation.

Mobile App Development – What are the common use cases for Automotive APIs being used in Mobile App development?

Many APIs match those in the retail industry – product catalog, product detail, etc. In addition, APIs for available accessories, incentives, owner’s manual information, and service locations could be provided. Looking at auto dealerships, APIs for location, available inventory, and prices are sample APIs.

Custom APIs that map your specific vehicle to required service, recalls, or trade in value are also good candidates for APIs.

Mobile advantages – APIs that unlock the car door or start the car are typical examples where mobile Apps might access APIs. However, these need to be secured so locked out that only the owner of the car can access the APIs to perform these functions on the car. Direct calls from the mobile App to car APIs might be hacked leading to theft or potential danger. So, authenticating the user is critical.

What APIs are available today in the Automotive Industry? Here are just a few samples of public APIs:

1. Dash Chassis API – is a connected car platform providing access to fuel consumption, expenses, and efficiency data. It also offers driving statistics and data, route information, and provides real time alerts.

2. IN Decoder – will return manufacturer data including: Manufacturer, manufacturer address, product type, make, check digit, model year, plant code, and sequential number, as well as data from police data bases.

3. GM Developer APIs – supports unlocking doors or activating the alarm, and accessing subscriber or vehicle data.

4. Ford API – offers a developer program for developers to create voice-activated applications to be integrated into vehicle technology.

5. Kelly Blue Book InfoDriver API – syndicates data about car models, features, and selling prices from the provider’s extensive database covering both new and used cars across the full range of makes and models and across North American markets. Methods provide projected car values at levels for suggested dealer price, used car trade-in transactions, and private party sales. The API includes

access to a VIN decoder to determine a specific car's equipment level.

2 The task workflow

The task of the work is to create application with Google Maps. The Google Maps API which could be used in your application:

- Google Maps Android API;
- Google Maps SDK for iOS;
- Google Maps JavaScript API;
- Google Static Maps API;
- Google Maps Embed API;
- Google Places API for Android;
- Google Places API for iOS;
- Google Places API Web Service;
- Google Maps Roads API;
- Google Street View Image API;
- Google Maps Geocoding API;
- Google Maps Directions API;
- Google Maps Distance Matrix API;
- Google Maps Geolocation API;
- Google Maps Elevation API;
- Google Maps Time Zone API.

Development of the Application with Google Maps should start with following steps:

- Start Android Studio.
- Create a new project as follows:
 - If you see the **Welcome to Android Studio** dialog, choose **Start a new Android Studio project**, available under 'Quick Start' on the right of the dialog.
 - Otherwise, click **File** in the Android Studio menu bar, then **New, New Project**.
 - Enter your app name, company domain, and project location, as prompted. Then click **Next**.
 - Select the form factors you need for your app. If you're not sure what you need, just select **Phone and Tablet**. Then click **Next**.
 - Select **Google Maps Activity** in the 'Add an activity to Mobile' dialog. Then click **Next**.

- Enter the activity name, layout name and title as prompted. The default values are fine. Then click **Finish**.

Next you should get an API key and enable the necessary APIs [7]:

1. Go to the Google API Console.
2. Create or select a project.
3. Click Continue to enable the Google Maps Android API.
4. On the Credentials page, get an API key.
5. Note: If you have an existing API key with Android restrictions, you may use that key.
6. From the dialog displaying the API key, select Restrict key to set an Android restriction on the API key. In the Restrictions section, select Android apps, then enter your app's SHA-1 fingerprint and package name. For example:

```
BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:91:AF:A1:66:6E:44:5D:75  
com.example.android.mapexample
```

7. Click Save.

Create prototype of the mobile application.

Create IFML model for the application.

Create minimum viable product and upload it to the github.

3 Report content

The content of the report is as follows:

1. Title, goal, tasks.
2. The requirements to the system.
3. Prototype of the mobile application.
4. IFML model of the designed system.
5. Link to the git-hub with the mobile application
6. Answers to the self-assessment questions according to the individual task.

4 Self-assessment questions

1. Which technologies exist for implementation of the connected cars approach?
2. For 3 industry leaders in the automotive industry describe existing solutions for V2V applications.

3. Describe 5 examples for the V2I and V2P applications.
4. Compare existing open API for the automotive industry.
5. Compare proprietary API for the automotive industry.
6. Compare open software for the mobile applications prototyping.

5 Recommended literature

1. Guth J. et al. (2018) A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences. In: Di Martino B., Li KC., Yang L., Esposito A. (eds) Internet of Everything. Internet of Things (Technology, Communications and Computing). Springer, Singapore

2. "Design Patterns for the Internet of Things", Community.arm.com, 2019. [Online]. Available: <https://community.arm.com/iot/b/blog/posts/design-patterns-for-an-internet-of-things>. [Accessed: 28- Jul- 2019].

3. M. Vega-Barbas, I. Pau, J. C. Augusto and F. Seoane, "Interaction Patterns for Smart Spaces: A Confident Interaction Design Solution for Pervasive Sensitive IoT Services," in IEEE Access, vol. 6, pp. 1126-1136, 2018.

4. "IFML: The Interaction Flow Modeling Language | The OMG standard for front-end design", Ifml.org, 2019. [Online]. Available: <https://www.ifml.org/>. [Accessed: 28- Jul- 2019].

5. "IFML online tool" [Online]. Available: <http://www.ifmledit.org/>

6. M. Bambilo, "Interaction Flow Modeling Language in the IIoT context". [Online] Available: <https://www.omg.org/news/meetings/tc/ma-15/special-events/iioT-pdf/Brambilla.pdf>

7. "Google developers' documentation". [Online]. Available: <https://developers.google.com/maps/documentation/android-api/signup>

2 PROGRAM TOOLS FOR THE SMART SYSTEMS SIMULATION

Laboratory work 1 Simulation of the Arduino devices in Proteus 8 Professional. Getting started

The purpose of laboratory work:

Get acquainted with the location and main functions of the Arduino Mega board contacts; connect the board to a power source and run a test sketch.

Teaching tasks:

Know the location of the main inputs/outputs of the board contacts, their parameters and functions.

Practical experience:

Connect the ARDUINO Mega card to your computer and download the sketch program

1 Brief theoretical information

To perform laboratory work you need to get acquainted with the stand "Smart Home". This stand consists of two parts. The first non-portable part (the panel with sensors) is the image of the house with sensors installed on it and their corresponding images with the outputs of sensors (Fig. 1).



Fig. 1 – General view of the stand panel with sensors

The second part is a multifunctional unit within which board of Arduino Mega 2560 is located (Fig. 2). On the front of this block are the images of the board, all its outputs and additional fields that allow to work with more than one sensor at the same time.

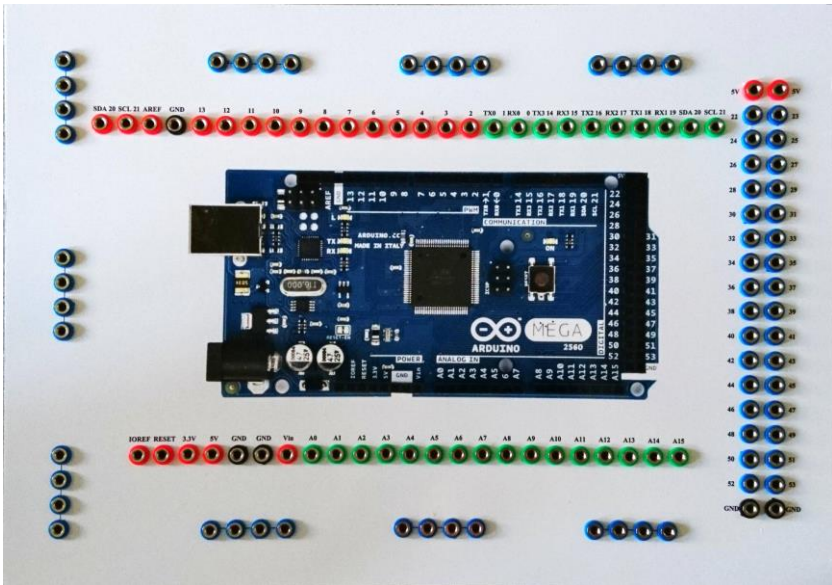


Fig. 2 – General view of the multifunctional unit

1.1 Definition of ARDUINO set

Arduino is a set of electronic unit and software storage. An electronic unit here is a printed circuit board with a set microcontroller and a minimum of elements, which are necessary for its functioning. The software is needed to create control programs. It consists of a simple development environment the Arduino IDE and programming language, namely the C/C++ version for microcontrollers, supplemented by certain functions for controlling inputs/outputs on the contacts of board. In fact, the electronic unit Arduino is analogous to the motherboard of the modern computer. Arduino Mega 2560 (Fig. 1.3) – this is a device, which based at microcontroller ATmega2560

(datasheet). The board itself consists of:

- 54 digital inputs/outputs (15 can be used as PWM-outputs);
- 16 analog inputs;
- 4 UART (hardware receivers for the implementation of serial interfaces);
- quartz resonator at 16 MHz;
- USB connector;
- power connector;
- ICSP connector for internal circuit programming;
- reset button.

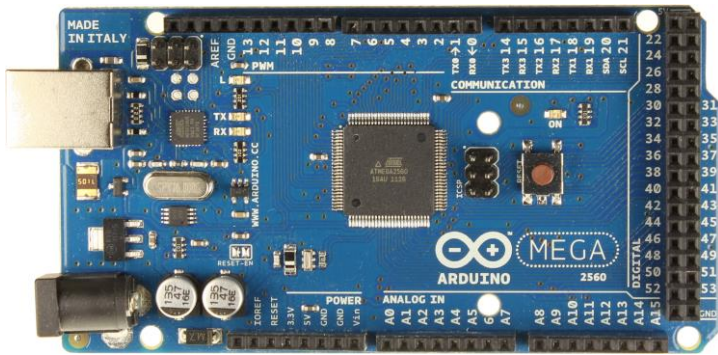


Fig. 3 – Board Arduino Mega 2560

1.2 Technical characteristics Arduino Mega 2560

The main characteristics of the Arduino Mega 2560 are shown in the table 1.

Table 1 – Characteristics of the Arduino Mega 2560

Operating voltage	5 V
Power supply (Recommended)	7-12 V
Power supply (Maximum)	6-20 V
Digital inputs / outputs	54
Analogue inputs	16
Maximum current of 5 V output	40 mA
Maximum current of 3.3 V output	50 mA
Clock frequency	16 MHz

1.3 Power supply Arduino Mega 2560

The board Arduino Mega 2560 can be powered by two methods, namely:

- from a computer by using a USB cable;
- from an AC / DC adapter or from battery.

When using an external power supply, it is necessary to select its value in the range of 6 to 20 V. But when using a power source with a voltage below 7 V, the voltage at the output 5 V decreases, which leads to unstable function of the board. Using a power supply with a voltage higher than 12V leads to overheating of the voltage regulator and to breaking down of board. Considering this, it is recommended to use a power source with a voltage value in the range of 7 to 12 V.

1.4 Inputs and outputs of Arduino Mega 2560

The main inputs and outputs that are located on the Arduino Mega 2560 (Fig. 4) are designed to connect of contacts board with sensors or actuators. Pins on the board Arduino Mega 2560:

- **VIN** – the input voltage to the board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V** – this pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V).
- **3.3V** – this pin comes in 3.3 V from the voltage regulator on the board. Maximum current is 50 mA;
- **GND** – 2 ground pins;
- **IOREF** – this pin on the board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

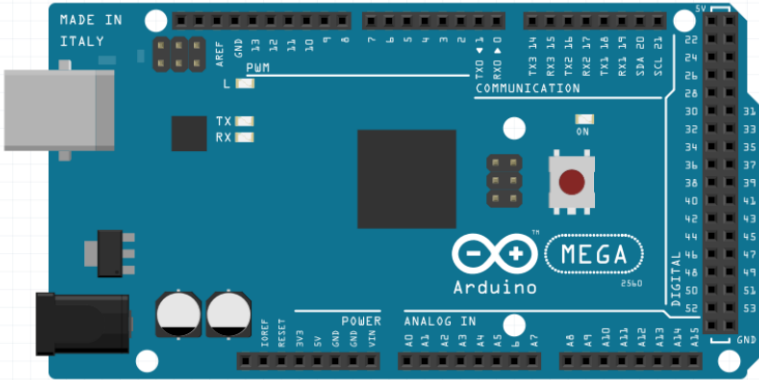


Fig. 4 – The pins location of the board Arduino Mega 2560

The Arduino Mega 2560 is located 54 digital pins. Each digital contact can be used as an input and output, using functions *pinMode()*, *digitalWrite()* and *digitalRead()*. They work at 5 V voltage. Each pin can provide or receive 20 mA in the recommended operating mode and has an internal load resistor (default disabled) with a nominal impedance of 20-50 k Ω . The maximum permissible current value is 40 mA - this value should not be exceeded to avoid damage to the microcontroller. In addition, some foams have specialized functions:

Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.

External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low level, a rising or falling edge, or a change in level. See the *attachInterrupt()* function for details.

PWM: from 2 to 13 and from 44 to 46. Provides 8-bit PWM output using the *analogWrite()* function. PWM is a pulse-width modulation, an operation for obtaining a variable analog value using digital devices. By outputting a signal that consists of high and low levels, the voltage is modeled between the maximum (5 V) and the minimum (0 V) values.

The duration of switching on the maximum value is called the pulse width. It changes to get different analog values.

SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These foams support the SPI connection using the SPI libraryI. SPI is a serial peripheral interface. SPI is a synchronous interface in which each transmission is synchronized with a clock signal generated by the master device (microcontroller).

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the wire library

The Mega 2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 V, though is it possible to change the upper end of their range using the AREF pin and *analogReference()* function.

There are a couple of other pins on the board:

AREF: Reference voltage for the analog inputs. Used with *analogReference()*.

Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

1.5 Arduino IDE

The development of programs to perform certain tasks with the help of the Arduino boards is carried out in the official programming environment of the Arduino IDE (Fig. 5). This environment is intended for writing, compiling and downloading created programs in the memory of the microcontroller. It consists of the following main elements:

- Program code editor;
- Notification area;
- Text output window;
- Toolbar with command buttons;
- Several menus.

A program written in the Arduino IDE programming environment is called sketch. It is written in a program code editor (Fig. 6).

When saving and exporting a project in the notification area appear explanations and error information.

The text output window shows Arduino messages, which show full error reports and other important information.

The toolbar buttons allow you to check and write the program, create, open and save the sketch; open the monitor for the serial bus.

Developed sketches can receive additional functions through libraries, which are a specially designed code. It helps to realize some of the opportunities that can be added to the project. There are many specialized libraries. Usually, libraries are written to simplify the decision of a particular task.

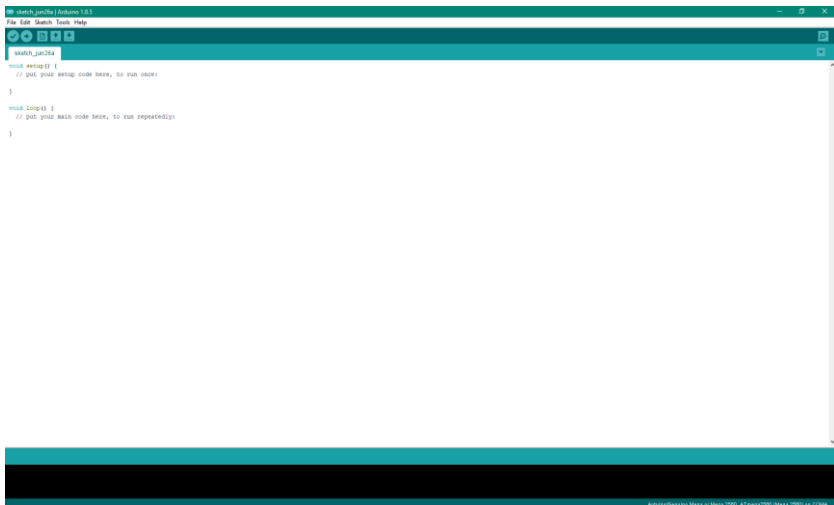


Fig. 5 – General view of Arduino IDE programming environment

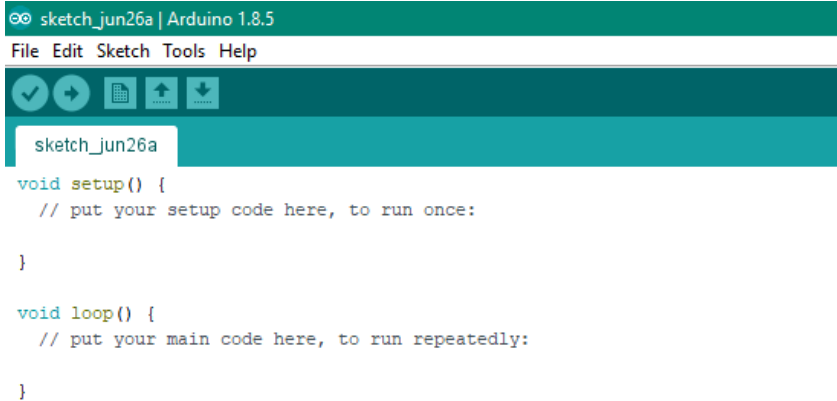


Fig. 6 – Basic elements of the Arduino IDE programming environment

2 Program of developments and researches

- 2.1 Learning the "Smart House" stand;
- 2.2 Connecting the multifunction unit to the power supply, to the computer (laptop);
- 2.3 Run the test sketch.

3 Guidelines for a work

Connect with the USB cable the appropriate outputs of the multifunction unit and computer to work with the board Arduino Mega 2560, which is located inside the multifunction unit.

Check for the Arduino IDE environment development required on your computer to work with the board Arduino Mega 2560. In the absence of this program - download it from the official site: <https://www.arduino.cc/en/Main/Software>

Run on the computer the program of development environment Arduino IDE.

Select the board Arduino Mega 2560 by: «Tools/Board/Arduino Mega 2560» (Fig. 7).

Program Tools for the smart systems simulation

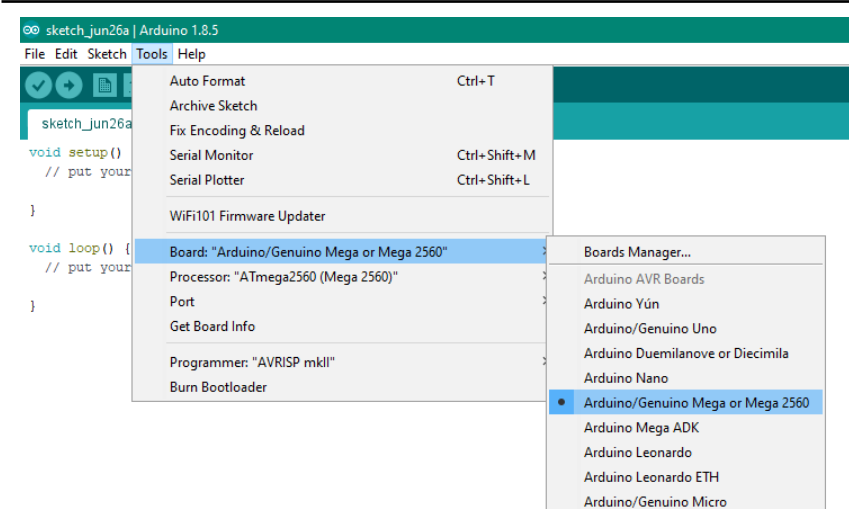


Fig. 7 – Choice of board Arduino Mega 2560

The next step is to select the appropriate port by Tools/ Port/ COM1 (Arduino Mega 2560) (Fig. 8).

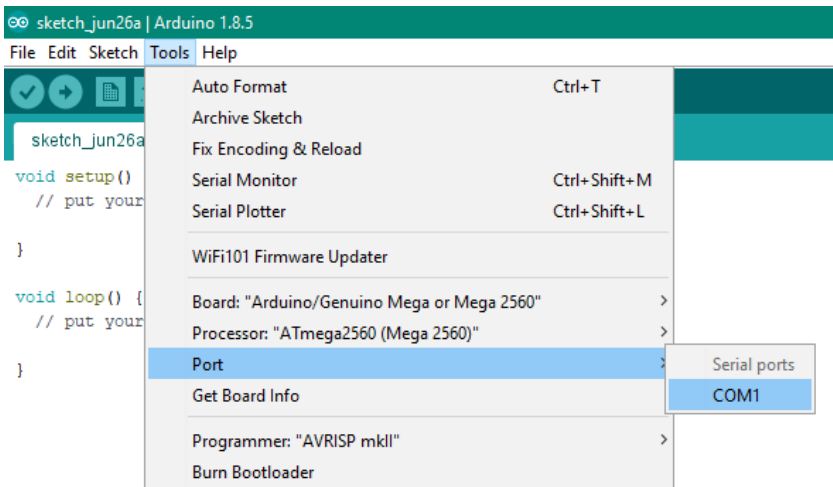


Fig. 8 – Choice of the port

To verify the correct connection of the board, need to create a text sketch. To do this, first needs to connect the LED on the stand by the wires to the multifunctional unit.

The LED connection is made as follows:

1) Cathode connects to any GND in foam panel of the multifunction unit;

2) Anode connected to pin 22 in foam panel of the multifunction unit (this pin is digital).

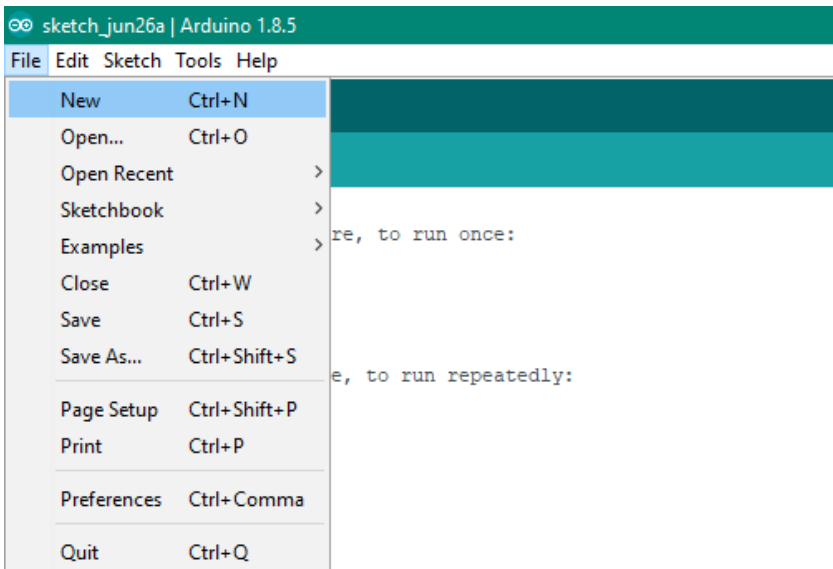


Fig. 9 – Create a new sketch

Next you need to open the program Arduino IDE, create a new sketch (in the toolbar, click “File / New” (Fig. 1.9)), load the next sketch there:

```
int led = D22;
void setup()
{
  pinMode(led, OUTPUT);
}
```

```
void loop()
{
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
```

Next, need to check the sketch text for errors: in the toolbar, click “Sketch / Verify” (Fig. 10). After that, at the status window will display information about the structure of the sketch, as well as the presence or absence of pillocks.

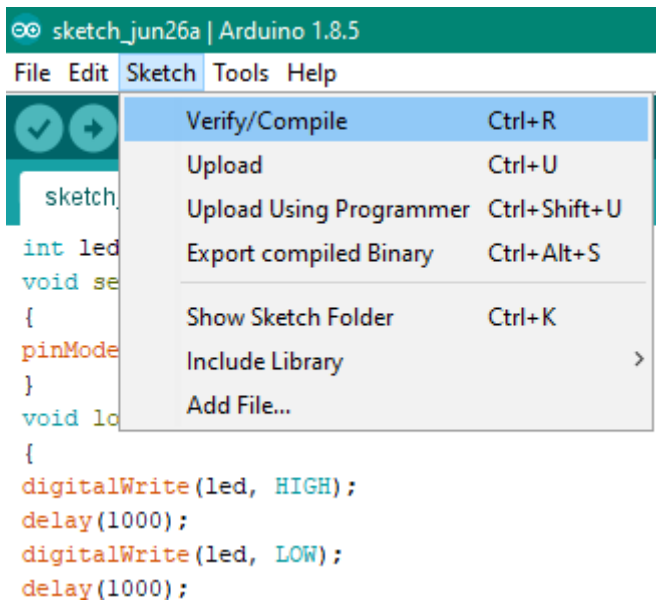


Fig. 10 – Function for verify a sketch

If the program does not detect any inequalities in the code, man can download sketch to the microprocessor. To do this, click : “Sketch / Upload” (Fig. 11).

After this, the LED should be measured once a second.

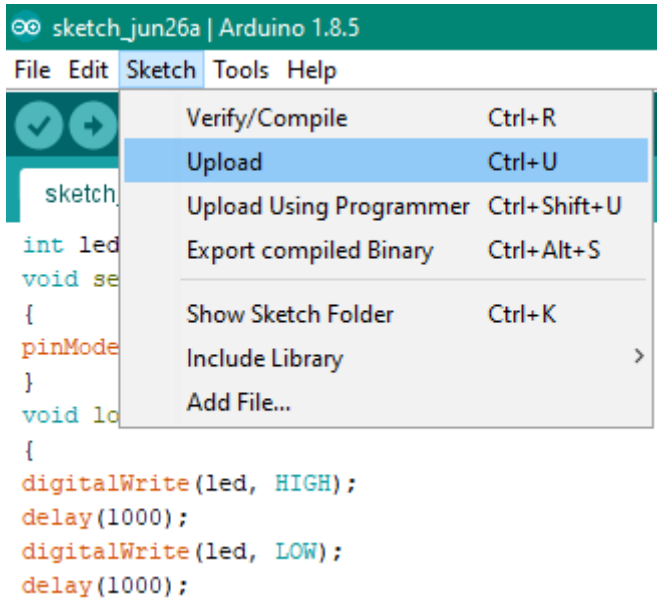


Fig. 11 – Function of upload a sketch

4 Individual tasks

According to variant, the sketch should be remade in accordance with next tasks:

1. Connect 1st LED so that it lights;
2. Connect 2nd LED so that it lights;
3. Connect 3rd LED so that it lights;
4. Connect 4th LED so that it lights;
5. Connect 5th LED so that it lights;
6. Connect 6th LED so that it lights;
7. Connect 7th LED so that it lights;
8. Connect 8th LED so that it lights;
9. Connect 9th LED so that it lights;
10. Connect 1st LED so it flashes twice every 1 second;
11. Connect 2nd LED so it flashes twice every 1 second;
12. Connect 3rd LED so it flashes twice every 1 second;
13. Connect 4th LED so it flashes twice every 1 second;

14. Connect 5th LED so it flashes twice every 1 second;
15. Connect 6th LED so it flashes twice every 1 second;
16. Connect 7th LED so it flashes twice every 1 second;
17. Connect 8th LED so it flashes twice every 1 second;
18. Connect 9th LED so it flashes twice every 1 second;
19. Connect 2 LEDs so that 1st lights and 9th LED so it flashes twice every 1 second;
20. Connect 2 LEDs so that 2st lights and 9th LED so it flashes twice every 1 second;
21. Connect 2 LEDs so that 3rd lights and 9th LED so it flashes twice every 1 second;
22. Connect 2 LEDs so that 4th lights and 9th LED so it flashes twice every 1 second;
23. Connect 2 LEDs so that 5th lights and 9th LED so it flashes twice every 1 second
24. Connect 2 LEDs so that 6th lights and 9th LED so it flashes twice every 1 second
25. Connect 2 LEDs so that 7th lights and 9th LED so it flashes twice every 1 second.

5 Requirements to the content of the report

- 5 Bring brief description of the structure of the stand, the Arduino Mega card, its input and output data.
- 6 Bring the text of the test sketch.
- 7 Bring a screenshot to the sketch text.
- 8 Bring the text to the sketch according to an individual task with explanations.
- 9 Bring a screenshot with sketch text according to an individual task from Serial Monitor.

6 Control questions

1. From what parts does the stand "Smart House" consist?
2. What board is inside a multifunctional unit?
3. What does Arduino mean? Of which parts does it consist?
4. What programming language is used in the Arduino IDE development environment?

5. What is a board Arduino?
6. What elements does the Arduino Mega 2560 card have?
7. What is the operating voltage on the outputs of the Arduino Mega 2560?
8. What is the maximum and recommended power supply voltage for power supply of the Arduino Mega 2560?
9. How is the Arduino Mega 2560 powered?
10. What are the consequences of power supply that is not included in the recommended values?
11. Which pins does Arduino Mega 2560 have?
12. How many digital pins are on the Arduino Mega 2560 and what are their characteristics?
13. What is the principle of PWM outputs and where are they located on the board?
14. How many analog pins are on the Arduino Mega 2560 and what are their characteristics?
15. For what is pin Reset used?
16. What is the development environment Arduino IDE and what elements does it consist of?
17. What is a sketch?
18. How to create a new sketch?
19. How to choose the necessary board, port when you connect a computer to multifunctional block?
20. How to download and check errors in generated sketch?

7 Recommended literature

1. V. Petin, Arduino и Raspberry Pi в проектах Internet of Things. Petersburg: BHV, 2014, p. 432.
2. Brian W. Evans, Arduino Programming Notebook. 2007.
3. T. Kurt, "Bionic Arduino – Introduction to Microcontrollers with Arduino – todbot blog", Todbot.com, 2019. [Online]. Available: <https://todbot.com/blog/bionicarduino/>. [Accessed: 30- Jul- 2019].
4. "Arduino - Software", Arduino.cc, 2019. [Online]. Available: <https://www.arduino.cc/en/Main/Software>. [Accessed: 30- Jul- 2019].
5. "Arduino - Windows", Arduino.cc, 2019. [Online]. Available: <https://www.arduino.cc/en/Guide/Windows>. [Accessed: 30- Jul- 2019].

Laboratory work 2

Simulation of the Arduino devices in Proteus 8 Professional. Using of light sensors in smart electrical systems

The purpose of laboratory work:

Get to know about and learn how to use the light sensor LM393.

Teaching tasks:

Know the operating principle of the light sensor LM393, its technical characteristics and usage features.

Practical experience:

Connecting the light sensor LM393 to the Arduino Mega 2560

1 Brief theoretical information

Introduction to lighting sensors in intelligent electrical systems begins with the use of the light sensor LM393 (Fig. 1), the main part of which is a photoresistor.



Fig. 1 – Light sensor LM393

Module light sensor LM393, used to measure light intensity in certain cases, such as automation of light control (light on at night).

1.1 Principle of operation of the sensor

Measurement is carried out using a photo-sensitive element (photoresistor), which changes the resistance, depending on the illumination (Fig. 2).

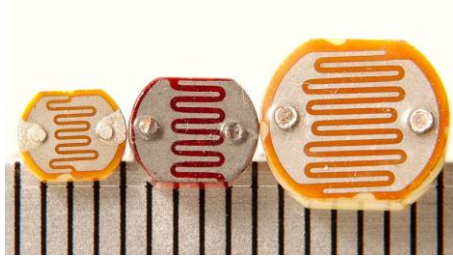


Fig. 2 – Photoresistor

Photoresistor – photovoltaic semiconductor radiation detector whose principle is based on the photoconductivity effect - the phenomenon of resistances reduction of a semiconductor in the case of the excitation of charge carriers by light. It is characterized by the same conductivity irrespective of the direction of flow of current.

Photoconductivity - the phenomenon of increasing the electrical conductivity of matter in the illumination.

Semiconductors are materials whose electrical conductivity has an intermediate value between the conductance of conductor and dielectric. They differ from the conductors by a strong dependence of specific conductivity on the concentration of impurities, temperature and various types of radiation. The main property of these materials is an increase in electrical conductivity with increasing temperature.

Thus, when changing light from bright light to darkness, its resistance varies from hundreds of Ohms to several mega Ohms, or vice versa.

1.2 Construction of module light sensor LM393

Module light sensor LM393 consists of the following elements (Fig. 3):

- photoresistor - the main measuring element;
- 4 outputs - analog and digital inputs, ground output, power output;
- potentiometer - to control the level of lighting for operation;
- power indicator;
- analogue mode indicator.

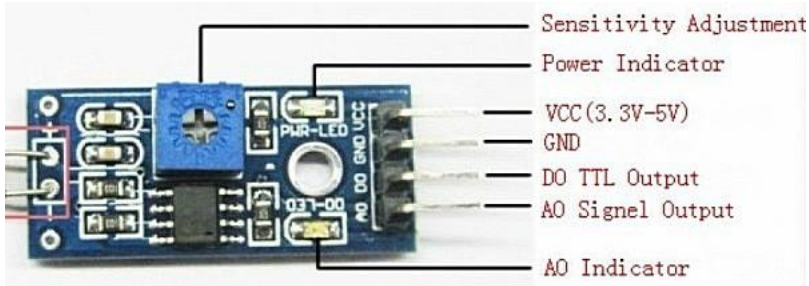


Fig. 3 – Construction of the module light sensor LM393

1.3 Connecting the sensor to the Arduino boards

To connect the Light Sensor LM393 to the Arduino boards, the following foams are used:

- VCC – output for power supply from the Arduino board; for the sensor, a voltage of 5 V is required;
- GND – output for connecting ground from the Arduino board, from the corresponding pin (GND).
- D0 – digital output (if the level of illumination of the photoresistor is higher than the set threshold, then the log LOW is formed);
- A0 – analog output (voltage at contact A0 is proportional to the luminance of the photoresistor).

1.4 Technical characteristics of the Light Sensor LM393

The main technical characteristics of the Light Sensor LM393 are given in Table 1.

Table 2.1– Technical characteristics of the light sensor LM393

Operating voltage	5 V
Nominal current	15 mA
Consumption current	10 mA
Analogue output	0 B ... VCC
Digital output:	TTL (log 1 or lof 0)
Operating temperature	from 0 to + 80 C
Dimensions	42x15x8 mm

2 Program of developments and researches

2.1 Creating a sketch for working with a sensor

2.2 Loading sketch on the Arduino board, which is installed in the multifunction unit

2.3 Connecting the sensor

2.4 Checking the created sketch

3 Guidelines for a work

3.1 Upload sketch

To perform any operations with the LM393 lighting sensor, need to create and download sketch into the microprocessor of Arduino Mega 2560, taking into account sensor functions and the task.

To do this, need to open the program, to create a new sketch (on the toolbar, click File / New), to download the following sketch (Fig. 4):

```
int pinA0 = A0
int pinD0 = 33
void setup()
{
  pinMode (pinA0, INPUT);
  pinMode (pinD0, INPUT)
  Serial.begin (9600);
}
void loop()
{
  int xA0, xD0;
  xA0 = analogRead (pinA0);
  xD0 = digitalRead (pinD0);
  Serial.print("Sensor: ");
  if (xD0 == HIGH)
  {
    Serial.println ("Dark");
  }
  else
  {
    Serial.println ("Light");
  }
  delay (2000);
}
```



```

Ir2 | Arduino 1.8.5
File Edit Sketch Tools Help

Ir2 $

int pinA0 = A0
int pinD0 = 33
void setup()
{
  pinMode (pinA0, INPUT);
  pinMode (pinD0, INPUT)
  Serial.begin (9600);
}
void loop()
{
  int xA0, xD0;
  xA0 = analogRead (pinA0);
  xD0 = digitalRead (pinD0);
  Serial.print("Sensor: ");
  if (xD0 == HIGH)
  {
    Serial.println ("Dark");
  }
  else
  {
    Serial.println ("Light");
  }
  delay (2000);

```

Fig. 4 – Sketch code

Next, need to check the sketch text for errors, for this in the toolbar, click “Sketch / Check”. After that, the status window will display information about the structure of the sketch, as well as the presence or absence of errors. If the checking did not detect any problems, then after connecting to the board, the light sensor LM393 will begin to work and perform operations according to the written sketch.

3.2 Connection of the sensor

At the stand "Smart House" is located the light sensor LM393. Its contacts in the appropriate sequence are at the bottom of this stand.

By wires need to be connected contacts of sensor to contacts of board Arduino Mega 2560, which are inside the multifunction unit.

The connection of the sensor is performed as follows:

- 1) Pin GND connects to any pin GND at pins panel of multifunction unit;
- 2) Pin VCC - to 5V at the pins panel of the multifunction unit;
- 3) Pin D0- to any digital input / output at the pins panel of the multifunction unit;
- 4) Pin A0- to any analog input / output at the pins panel of the multifunction unit.

3.3 Testing of the sensor

When testing the sensor for the already established sketch, namely, to check its efficiency, it is necessary to start the monitoring function of the port "Tools / Serial Monitor" (Fig. 5):

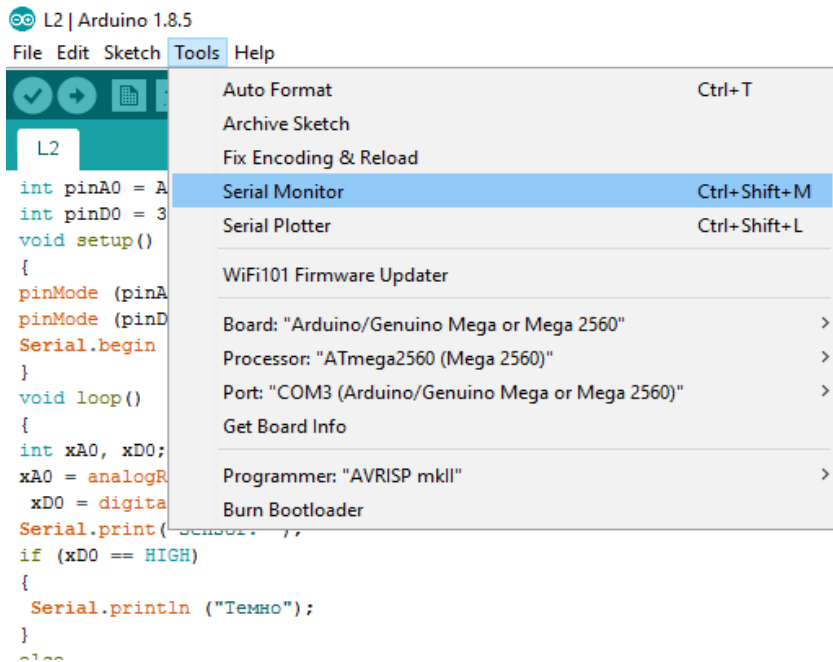


Fig. 5 – Serial Monitor

After open the port monitor and download the sketch to the microcontroller board, when connecting the sensor, according to the sketch code, the word "Light" will appear every 2 seconds, and when the photoresist is covered - word "Dark" (Fig. 6).

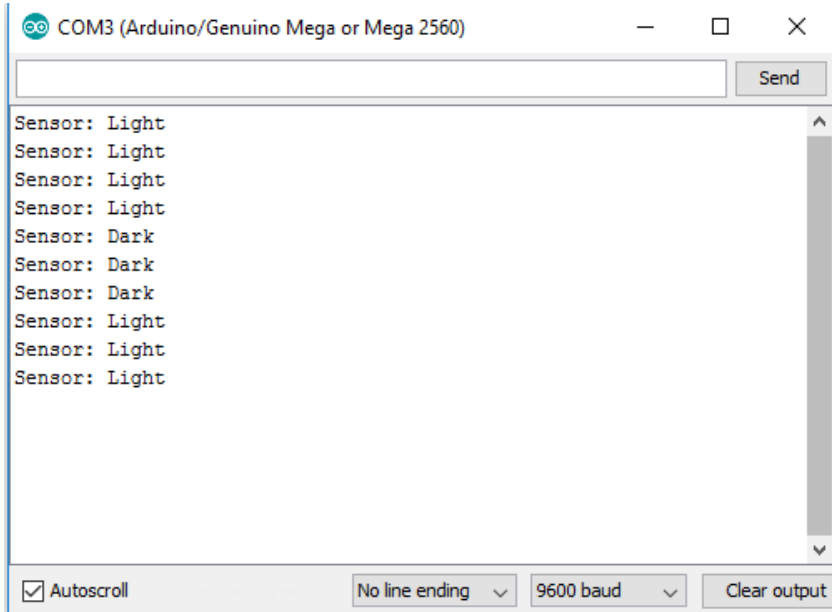


Fig. 6 – Testing results of the motion sensor

4 Individual tasks

According to variant, the sketch should be remade in accordance with next tasks:

1. LED flashes once every 1 second, when the sensor is triggered, it stops flashing;
2. LED flashes once every 1 second, when the sensor is triggered, it lights up;
3. LED flashes twice every 1 second, when the sensor is triggered, it stops flashing;

4. LED flashes twice every 1 second, when the sensor is triggered, it lights up;
5. LED flashes once every 2 seconds, when the sensor is triggered, it stops flashing;
6. LED flashes once every 2 seconds, when the sensor is triggered, it lights up;
7. LED flashes once every 3 seconds, when the sensor is triggered, it stops flashing;
8. LED flashes once every 3 seconds, when the sensor is triggered, it lights up;
9. LED lights up, and when the sensor is triggered, it starts flashing once every 1 second;
10. LED does not light, and when the sensor is triggered, it starts flashing once every 1 second;
11. LED lights up, and when the sensor is triggered, it starts flashing twice every 1 second;
12. LED does not light, and when the sensor is triggered, it starts flashing twice every 1 second;
13. LED lights up, and when the sensor is triggered, it starts flashing once every 2 seconds;
14. LED does not light, and when the sensor is triggered, it starts flashing once every 2 seconds;
15. LED does not light, and when the sensor is triggered, it starts flashing once every 3 seconds;
16. LED does not light, and when the sensor is triggered, it starts flashing once every 3 seconds;
17. LED lights up, and when the sensor is triggered, it does not light;
18. LED does not light, and when the sensor is triggered, it lights up;
19. LEDs are connected, the first flashes, and when the sensor is triggered, it does not burn; the second one lights up only when the sensor is triggered;
20. LEDs are connected, the first lights up, and when the sensor is triggered, it does not burn; the second one lights up only when the sensor is triggered;

21. 2 LEDs are connected, the first does not light, and when the sensor is triggered, it lights up; the second one flashes only when the sensor is triggered;
22. 2 LEDs are connected, the first lights up, and when the sensor is triggered, it does not burn; the second one flashes only when the sensor is triggered;
23. 2 LEDs are connected, the first does not light, and when the sensor is triggered, it lights up; the second one lights up only when the sensor is triggered;
24. 2 LEDs are connected, the first lights up, and when the sensor is triggered, it does not burn; the second one lights up only when the sensor is triggered;
25. 2 LEDs are connected, the first flashes, and when the sensor is triggered, it does not burn; the second one flashes only when the sensor is triggered.

5 Contents of the work report

5.1 Brief description of the construction, principle of operation, its inputs and outputs, and technical characteristics of the light sensor LM393

5.2 Bring the text of the testing sketch.

5.3 Bring screenshot of port monitoring during testing of sensor.

5.4 Bring the text of the sketch from individual tasks with explanations.

5.5 Bring screenshot of port monitoring during testing of sensor.

6 Control questions

- 1 For what is the lighting sensor LM393 used?
- 2 What is a Photoresistor?
- 3 What parameter of the photoresistor does change under the influence of light?
- 4 What is photoconductivity?
- 5 What are semiconductors?
- 6 What are different semiconductors between conductors?
- 7 How do properties of semiconductor change?

- 8 What elements does the module lighting sensor LM393 have?
- 9 For what is the potentiometer used in the light sensor?
- 10 How to connect light sensor LM393 to boards Arduino?
- 11 What pins are used to connect the lighting sensor LM393 to the stand "Smart Home"?
- 12 For what is pin VCC used?
- 13 What is the operating voltage and current of the lighting sensor LM393?
- 14 For what is pin GND used?
- 15 What function does the digital output D0 do?
- 16 What function does the analog output A0 do?
- 17 How to create a new sketch?
- 18 How to download test sketch for lighting sensor LM393?
- 19 How to start the monitoring function of the port, to check the sketch?
- 20 How to connect LED to control it?

7 Recommended literature

1. V. Petin, Arduino и Raspberry Pi в проектах Internet of Things. Petersburg: BHV, 2014, p. 432.
2. Brian W. Evans, Arduino Programming Notebook. 2007.
3. T. Kurt, "Bionic Arduino – Introduction to Microcontrollers with Arduino – todbot blog", Todbot.com, 2019. [Online]. Available: <https://todbot.com/blog/bionicarduino/>. [Accessed: 30- Jul- 2019].

Laboratory work 3

Simulation of the Arduino devices in Proteus 8 Professional.

Using of PIR Motion Sensors in smart electrical systems

The purpose of laboratory work:

Get to know about and learn how to use the motion sensor HC-SR501.

Teaching tasks:

Know the operating principle of the motion sensor HC-SR501, its technical characteristics and usage features.

Practical experience:

Connecting the motion sensor HC-SR501 to the Arduino Mega 2560.

1 Brief theoretical information

Introduction to motion sensors in smart electrical systems is carried out on the example of the infrared motion sensor HC-SR501 (Fig. 1), which refers to passive infrared sensors (PIR).



Fig. 1 – Motion sensor HC-SR501

PIR - pyroelectric (passive) infrared sensor.

Pyroelectricity is the property to generate a certain electric field when irradiated with infrared (thermal) beams.

Motion sensor is a contactless sensor that captures the movement of objects and is used to control the surrounding environment or automatically trigger the necessary actions in response to the movement of objects.

The principle of operation of the motion sensor is based on the analysis of thermal (infrared) radiation and received thermal rays from the outside world.

Infrared radiation is electromagnetic radiation occupying a spectral region between the red boundary of visible light with a wavelength $\lambda = 700$ nm and microwave radiation with a wavelength $\lambda \sim 1$ mm.

The infrared motion sensor HC-SR501 consists of a pyroelectric sensing element (Fig. 2 and Fig. 3), which is attached to the sensor module.

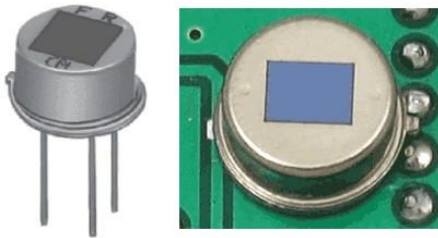


Fig. 2 – PIR element

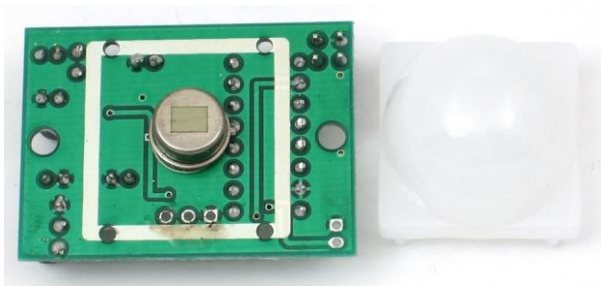


Fig. 3 – Module of sensor

1.1 Construction of the PIR sensor

Sensitive elements of the PIR-sensor are installed in a metal hermetic body (Fig. 3.3), which protects against external noise, temperature and humidity fluctuations. The centered rectangle is made

of a material that transmits infrared radiation (usually a silicone material). At this plate two sensitive elements are installed.

1.2 Principle of operation of the sensor

Pyroelectric motion sensor consists of two main parts. Each of its parts, is shown in Fig. 4, includes a special material which is sensitive to infrared radiation.

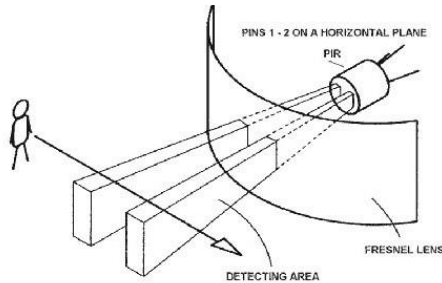


Fig. 4 – Principle of operation of the sensor

In this case, the lenses do not particularly affect the operation of the sensor, so we see two areas of sensitivity of the entire module. When the sensor is in a state of rest, both sensors determine the same amount of radiation. For example, this may be the radiation of the room or the environment on the street. When a warm-blooded object (human or animal) passes by, it crosses the sensitivity zone of the first sensor, resulting in two different values of radiation will be on the module of the PIR sensor. When a person leaves the zone of sensitivity of the first sensor, the values are aligned. The changes in the readings of the two sensors are logged and generate HIGH or LOW pulses at the output.

1.3 Connecting the sensor to the Arduino boards

To connect an infrared motion sensor HC-SR501 to the Arduino board, the following pins are used (Fig. 5):

- VCC – output for power supply from the Arduino board; for the sensor, a voltage of 5 V is required;
- OUT – digital output, (for example: the OUT pin gives a high logical level when the object is detected in the sensor zone);

- GND – output for connecting ground from the Arduino board, from the corresponding pin (GND).

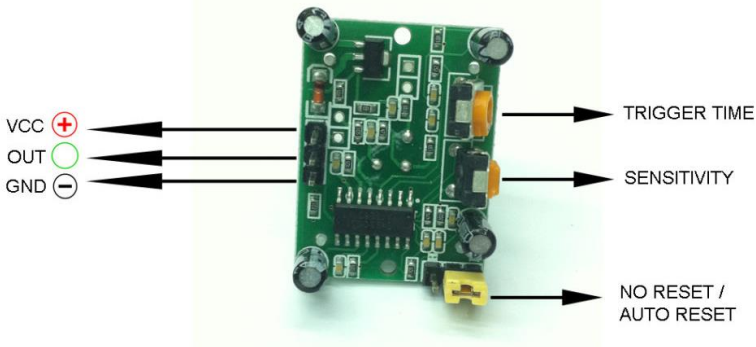


Fig. 5 – Contacts of sensor

1.4 Methods for setting sensor parameters

In the module of the infrared motion sensor HC-SR501 (Fig. 6) as a controller of settings, are used two potentiometers and jumper, which allow you to control the following parameters:

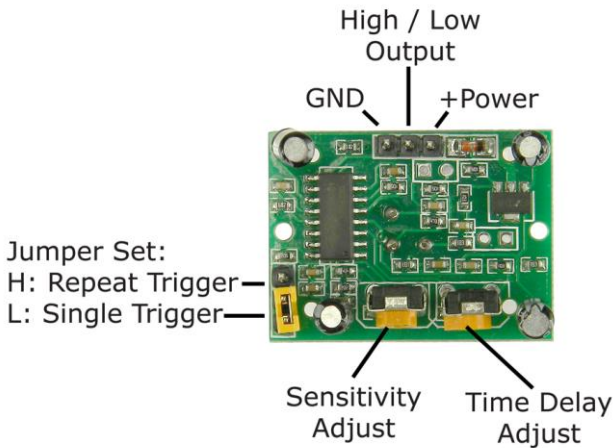


Fig. 6 – Module of PIR-sensor

- parameter "Delay Time Adjust", which is controlled by one of the potentiometers, determines the time during which a high logic level of the signal (5-300 seconds) will be maintained on the OUT contact when detecting the movement;
- parameter "Distance Adjust" determines the sensitivity of the sensor - the center of the controlled area can vary from 3 to 7 m.

1.5 Work modes

The work mode of the infrared motion sensor HC-SR501 is given by jumper, which provides the choice of one of two modes: mode H or mode L. If the mode H is set in the module (Fig. 3.6) when the sensor is triggered several times in succession, at its output (OUT) remains high logical level. In mode L, each time a sensor is triggered, a separate pulse appears on the output of the module.

1.6 Technical characteristics

The main technical characteristics of infrared motion sensor HC-SR501 are given in Table 1.

Table 1– Technical characteristics of the sensor HC-SR501

Operating voltage		5 V
Nominal current		50 mA
Voltage at output OUT:	HIGH	3,3 B
	LOW	0 V
Detection interval		3-7 m
Length of delay after operation		5 - 300 s
Observation angle		until 120
Lock time until next measurement		
Operating temperature		from -20 to + 80 C
Dimensions		32x24x18 mm
Operating modes:	L	solitary operation
	H	operation at each time

1.7 Features of using

When using the sensor, it is necessary to take into account certain its features, namely:

To operate the sensor from the beginning, the bridge is set to L mode and the power is fed. But need to wait about 20-40 seconds - this time is used for calibration. Now, as soon as the sensor records movement, to the corresponding digital pin of the Arduino board, a high logic signal will be received the certain time, which is set by the Delay Time Adjust potentiometer.

2 Program of developments and researches

2.1 Creating a sketch for working with a sensor

2.2 Loading sketch on the Arduino board, which is installed in the multifunction unit

2.3 Connecting the sensor

2.4 Checking the created sketch

3 Guidelines for a work

3.1 Upload sketch

To perform any operations with the motion sensor HC-SR501, need to create and download sketch into the microprocessor of Arduino Mega 2560, taking into account sensor functions and the task.

To do this, need to open the program, to create a new sketch (on the toolbar, click File / New), to download the following sketch (Fig. 7):

```
#define pirPin 2
#define LedPin 13
void setup() {
  Serial.begin(9600);
  pinMode(pirPin, INPUT);
  pinMode(LedPin, OUTPUT);
}
void loop() {
  int pirVal = digitalRead(pirPin);
```

```

if (pirVal == HIGH) {
    digitalWrite(LedPin, HIGH);
    Serial.println("Motion detected");
    delay(300);
} else {
    Serial.println("No motion");
    digitalWrite(LedPin, LOW);
    delay(300);
}

```

Next, it is need to check the sketch text for errors, for this in the toolbar, click “Sketch / Check”. After that, the status window will display information about the structure of the sketch, as well as the presence or absence of errors. If the checking did not detect any problems, then after connecting to the board, the motion sensor HC-SR501 will begin to work and perform operations according to the written sketch.

```

#define pirPin 2
#define LedPin 13
void setup() {
    Serial.begin(9600);
    pinMode(pirPin, INPUT);
    pinMode(LedPin, OUTPUT);
}
void loop() {
    int pirVal = digitalRead(pirPin);
    if (pirVal == HIGH) {
        digitalWrite(LedPin, HIGH);
        Serial.println("Motion detected");
        delay(300);
    } else {
        Serial.println("No motion");
        digitalWrite(LedPin, LOW);
        delay(300);
    }
}

```

Fig. 7 – Sketch code

3.2 Connecting of the sensor

At the stand "Smart House" is located the motion sensor HC-SR501. Its contacts in the appropriate sequence are at the bottom of this stand. By wires need to be connected contacts of sensor to contacts of board Arduino Mega.

The connection of the sensor is performed as follows:

- 1) Pin GND connects to any pin GND at pins panel of multifunction unit;
- 2) Pin VCC – to 5V at the pins panel of the multifunction unit;
- 3) Pin OUT – to any digital input / output at the pins panel of the multifunction unit;

3.3 Testing the sensor

When testing the sensor for the already established sketch, namely, to check its efficiency, it is necessary to start the monitoring function of the port "Tools / Serial Monitor" (Fig. 8):

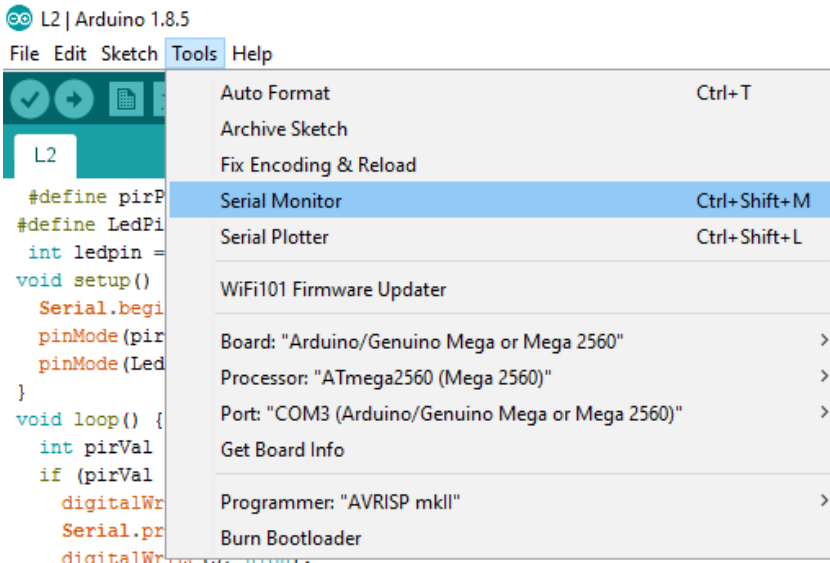


Fig. 8 – Serial Monitor

After open the port monitor and download the sketch to the microcontroller of board, when moving near the sensor - the text "Motion detected" appears in the monitoring window, and in the absence of the sketch code, once in a second, the message "No motion" appears (Fig. 9).

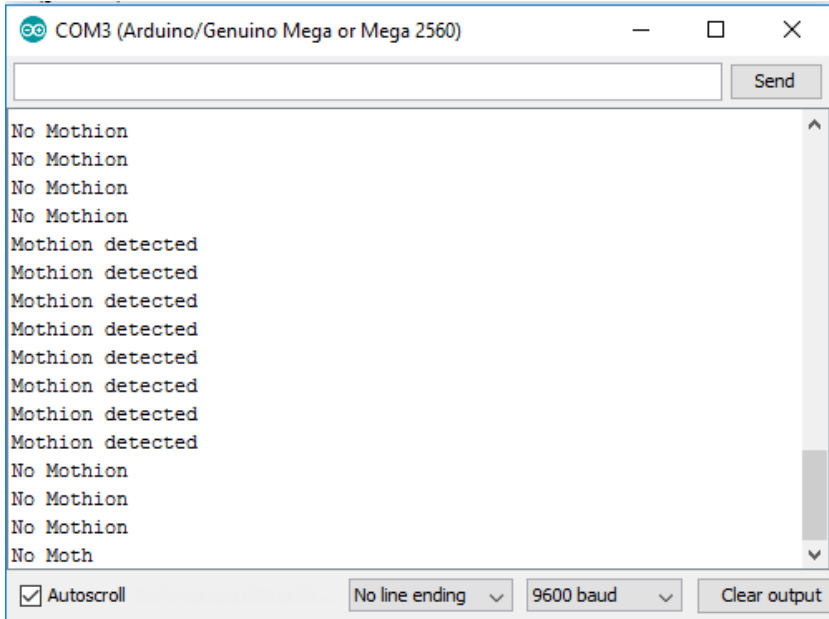


Fig. 9 – Testing results of the motion sensor

4 Individual tasks

According to variant, the sketch should be remade in accordance with next tasks:

1. LED lights up, and when the sensor is triggered, it starts flashing once every 1 second;
2. LED does not light, and when the sensor is triggered, it starts flashing once every 1 second;
3. LED lights up, and when the sensor is triggered, it starts flashing twice every 1 second;

4. LED does not light, and when the sensor is triggered, it starts flashing twice every 1 second;
5. LED lights up, and when the sensor is triggered, it starts flashing once every 2 seconds;
6. LED does not light, and when the sensor is triggered, it starts flashing once every 2 seconds;
7. LED does not light, and when the sensor is triggered, it starts flashing once every 3 seconds;
8. LED does not light, and when the sensor is triggered, it starts flashing once every 3 seconds;
9. LED lights up, and when the sensor is triggered, it does not light;
10. LED does not light, and when the sensor is triggered, it lights up;
11. LED flashes once every 1 second, when the sensor is triggered, it stops flashing;
12. LED flashes once every 1 second, when the sensor is triggered, it lights up;
13. LED flashes twice every 1 second, when the sensor is triggered, it stops flashing;
14. LED flashes twice every 1 second, when the sensor is triggered, it lights up;
15. LED flashes once every 2 seconds, when the sensor is triggered, it stops flashing;
16. LED flashes once every 2 seconds, when the sensor is triggered, it lights up;
17. LED flashes once every 3 seconds, when the sensor is triggered, it stops flashing;
18. LED flashes once every 3 seconds, when the sensor is triggered, it lights up;
19. 2 LEDs are connected, the first flashes, and when the sensor is triggered, it does not burn; the second one lights up only when the sensor is triggered;
20. 2 LEDs are connected, the first lights up, and when the sensor is triggered, it does not burn; the second one lights up only when the sensor is triggered;

21. 2 LEDs are connected, the first does not light, and when the sensor is triggered, it lights up; the second one flashes only when the sensor is triggered;

22. 2 LEDs are connected, the first lights up, and when the sensor is triggered, it does not burn; the second one flashes only when the sensor is triggered;

23. 2 LEDs are connected, the first does not light, and when the sensor is triggered, it lights up; the second one lights up only when the sensor is triggered;

24. 2 LEDs are connected, the first lights up, and when the sensor is triggered, it does not burn; the second one lights up only when the sensor is triggered;

25. 2 LEDs are connected, the first flashes, and when the sensor is triggered, it does not burn; the second one flashes only when the sensor is triggered.

5 Contents of the work report

5.1 Brief description of the construction, principle of operation, its inputs and outputs, and technical characteristics of the infrared motion sensor HC-SR501

5.2 Bring the text of the testing sketch.

5.3 Bring screenshot of port monitoring during testing of sensor.

5.4 Bring the text of the sketch from individual tasks with explanations.

5.5 Bring screenshot of port monitoring during testing of sensor.

6 Control questions

1. For what is the motion sensor HC-SR501 used?
2. How is the abbreviation PIR decoded??
3. What is pyroelectric?
4. What is infrared radiation?
5. What elements does the module motion sensor HC-SR501 have?
6. What is the principle of action of motion sensor HC-SR501?
7. What pins are used to connect the motion sensor HC-SR501 to the stand "Smart Home"?

8. For what is pin VCC used?
9. What is the operating voltage and current of motion sensor HC-SR501?
10. For what is pin GND used?
11. What impulses are generated on the digital output of the sensor?
12. How can be the sensor settings configured?
13. What function of the parameter Delay Time Adjust is?
14. What function of the parameter Distance Adjust is?
15. What are the work modes of the sensors, how to switch them?
16. What are the features of the use of sensors?
17. How to create a new sketch?
18. How to download test sketch for motion sensor HC-SR501?
19. How to start the monitoring function of the port, to check the sketch?
20. How to connect LED to control it?

7 Recommended literature

1. V. Petin, Arduino и Raspberry Pi в проектах Internet of Things. Petersburg: BHV, 2014, p. 432.
2. Brian W. Evans, Arduino Programming Notebook. 2007.
3. T. Kurt, "Bionic Arduino – Introduction to Microcontrollers with Arduino – todbot blog", Todbot.com, 2019. [Online]. Available: <https://todbot.com/blog/bionicarduino/>. [Accessed: 30- Jul- 2019].

3 THREE-LEVEL SIMULATION OF IOT-BASED SYSTEMS WITH THE USE OF UML MODELS, PETRI NETS AND TEMPORAL LOGIC

Laboratory work 1 Modelling Internet of Things Using UML Diagrams at the Stages of their Architectural Design

1 The purpose and objectives

The purpose of these guidelines is to consolidate and supplement the lecture material, as well as develop students' skills and abilities to understand existing IoT technologies and apply them to specific scenarios, as well as the ability to design complete IoT systems.

For the tasks of laboratory work, the basic theoretical and reference provisions are given, as well as general and individual tasks for independent work of students.

In the course of independent work preceding laboratory work, the study of lecture and additional material is carried out, preparation and analysis of solutions to problems of laboratory work.

At the beginning of each laboratory work an individual control of theoretical readiness is made, the result of which is admission to work. In the course of work, each student performs the task on time and draws up a protocol with the results of work. Control of knowledge, protection of a working program and protection of the protocol are made for each student individually.

The aim of the work is to study the capabilities of the structural-functional design of the IoT system when applying the basic methods and means of verifying models in the environment of the universal modeling language UML and the StarUML tool environment.

1.1 Teaching tasks:

- The study of the principles of structural-functional, object-component, event-time design of the iot system;
- Familiarization with the methodology of structural-functional, object-component, event-time iot system design and redesign when debugging iot system projects;

- Familiarization with the fundamentals of model verification, in particular, based on the use of intuitive- “manual”, full-selection, and automaton testing in the “promotion” method;
- Familiarization with the capabilities of the universal modeling language UML, the principles of operation and the basics of using the tool environment CPN Tools, which provides a formal specification and performs the basic structural and functional, object-component, event-time modeling and verification of IoT system projects.

1.2 Practical tasks:

- building models of the IoT system implemented at the application level;
- performing structural (static) intra-level verification of IoT system models based on the built-in tools of the CPN Tools environment.

1.3 Research tasks:

- analysis and subject optimization of the computational complexity of the design, verification and redesign of the IoT system;
- assessment and subject optimization of the length, multiplicity and completeness of verification tests when designing an IoT system.

2 Preparation for laboratory work

Preparation for laboratory work includes:

- clarification of the purpose and tasks of the laboratory work;
- the study of theoretical material given in the description and additional sources;
- selection of the IoT system design variant for the performance of laboratory work (in consultation with the teacher);
- definition of the program (scenario) of the laboratory work;
- check the installation of the necessary software (StarUML).

3 Execution of laboratory work

3.1 Procedure for laboratory work performing

Stage 1. In the first stage, the following steps are performed:

1. Construction of formal models of the architecture selected according to the variant of the IoT subsystem task - structural and functional UML diagrams (at least five) of the following levels:

a) the level of technical specifications (external adjacent environment of actors, their precedents) using UML-diagrams of use (precedents);

b) structural and functional level (structure, structure blocks, general IoT system functions, private block functions) using UML diagrams of components, internal structure, packages, interactions (communications), deployment (deployment);

c) the object-component level (informational entities, classes of objects and interfaces, data, parameters and methods sent in the indicated links of the IoT system structure) using UML diagrams of packages, components, classes, objects;

d) event-time level (conditions, events, actions and functions / handlers of the IoT system as a whole, its blocks / components, objects, interfaces, precedence-quasi-order relation for conditions, events, actions and functions / handlers based on the analysis of the interaction of block functions, object-interface methods and event handlers in the exchange of data, parameters and methods) using UML diagrams of interactions, components, internal structure, sequences;

e) automaton-algorithmic level (implementing component abstract automata, procedures and algorithms) using UML diagrams of activity, states, sequences.

2. The upper estimate of the computational complexity of the analysis and the construction of UML diagrams is performed using a simplified formula for basic entities and relations.

$$C_{UML} = a*(n_e+n_e^2+3*n_r), \quad (1)$$

where n_e is the number of classifiers of the diagram entities;

n_e^2 is the number of cells in a square matrix of possible relationships between entities;

n_r is the number of real assigned ratios of the diagram, the multiplier "3" means the need to consider both the relation r itself and the two entities e_1 and e_2 incident to it;

a - conditional coefficient of abstraction level (for technical specifications - 3, structural-functional - 2, object-component - 2, event-time - 1, automaton-algorithmic - 1).

3. Automatic syntactic verification in the StarUML environment in the "Model - Verify Model - Verify" menu panel with the analysis of the received verification messages and the execution of the UML diagrams if necessary.

4. Selective manual verification of specified instructions, out of thirty-eight specific for the analysis of the correctness of the model (adaptation of the UML specification instructions, see Appendix 1.10).

5. Building a script for automated local verification of specifications (properties) based on an intuitive, “manual”, full-choice testing method for constructing a path covering all entities for each of the selected diagrams (at least three different out of five) and determining the path length.

6. The upper estimate of the computational complexity of the covering path (including its construction) using the simplified formula:

$$C_{\text{Path}} = a \cdot (n_e^2 / 2 + n_r). \quad (2)$$

7. The upper estimate of the reduced computational complexity of the design due to the use of an automated local verification scenario in comparison with the case of re-design in the event of an error, when the computational complexity of analyzing and building the UML diagram is doubled, is determined by the simplified formula:

$$C_{\Delta C} = 2 \cdot C_{\text{UML}} - (C_{\text{UML}} + C_{\text{Path}}) = a \cdot (n_e + n_e^2 / 2 + 2 \cdot n_r). \quad (3)$$

3.2 Examples of the tasks

ormal models of the IoT subsystem architecture and local verification scenarios”.

Perform Phase 1

Through step-by-step construction of formal UML specifications and verification of the IoT system architecture are performed.

To represent the architecture of the IoT subsystem using UML diagrams, 5 types of diagrams are selected that provide a fairly complete representation of the IoT architecture. When designing a lighting system, these types of diagrams allow you to see it from the user's side. A method based on the construction of individual scenarios of the system (one of the variants of the program execution) is proposed. This method is implemented on the example of scenarios with a positive outcome of user interaction with the lighting system.

It is not necessary to use all chart types to represent the architecture of the IoT subsystem. So, when designing a lighting system and temperature control, these types of diagrams allow you to see it from the users - visitors to the premises. For a better understanding, you can imagine a hierarchy of users of the lighting system (see. Fig. 1).

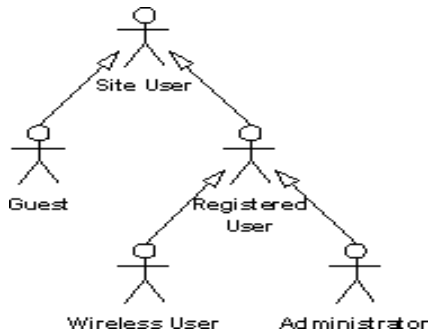


Fig. 1 – Performing-role hierarchy

The figure shows several user groups called in the UML language “actors” (actors) who will work with the lighting system.

3.2.1 Usage chart. At the initial stage, when it is necessary to clarify the idea of the IoT subsystem and understand the user’s vision, the use diagram (of precedents) is best suited (see Fig. 2).

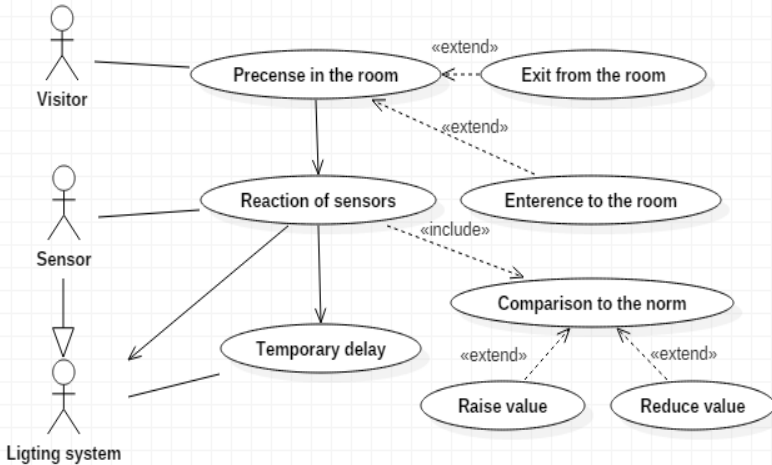


Fig. 2 – Usage Diagram (Use Case)

The usage diagram may be accompanied by a text description “Subsystem Use Options”.

Validation Entities:

- Design / analysis actors: teacher; student; lighting system (IoT subsystem).
- Precedents using actors for design / analysis: presence checking; sensor response; temporary delay; value options.
- Association relation for design / analysis: lighting system - presence check; lighting system - time delay
- Expansion ratio for design / analysis: presence check □ value options, sensor response - value options.

Intuitive heuristic validation of all entities of the subsystem's web service usage diagram can be performed by the customer and the contractor based on the complete (from beginning to the end of the design / analysis) presentation route of this use, showing the application and showing the correctness of each diagram in their end-to-end connected set:

- Entity entities - actors, entity-object - design / analysis.
- Entity-roles - precedents for the use of actors for designing // analysis.
- Entity-relationships are associations and extensions.
- The complete route of using entities for design / analysis includes the steps: 1) setting the initial limits; 2) determine the range of values; 3) tracking system accounts.

Consistent participation in the validation of all entities of the usage diagram, confirmed by the subsystem and the visitor, indicates its correctness.

Representation of the route of use is possible and natural to do with the help of a general diagram of the activity of the IoT subsystem.

The upper complexity scores are:

$$C_{UML} = a*(n_e + n_e^2 + 3*n_r) = 3*(10 + 10^2 + 3*9) = 411; \quad (4)$$

$$C_{Path} = a*(n_e^2/2 + n_r) = 3*(10^2/2 + 9) = 177; \quad (5)$$

$$L_{Path} = 3*9 = 27; \quad (6)$$

$$\Delta C = a*(n_e + n_e^2/2 + 2*n_r) = 3*(10 + 10^2/2 + 2*9) = 234. \quad (7)$$

Reducing the computational complexity of analysis and design through the use of an automated local verification script amounted to 234 conventional UML analysis units.

3.2.2 Package diagram. At the next stage, it is possible to set the structure for the IoT subsystem; the packet diagram (see Fig. 3) allows this to be done effectively.

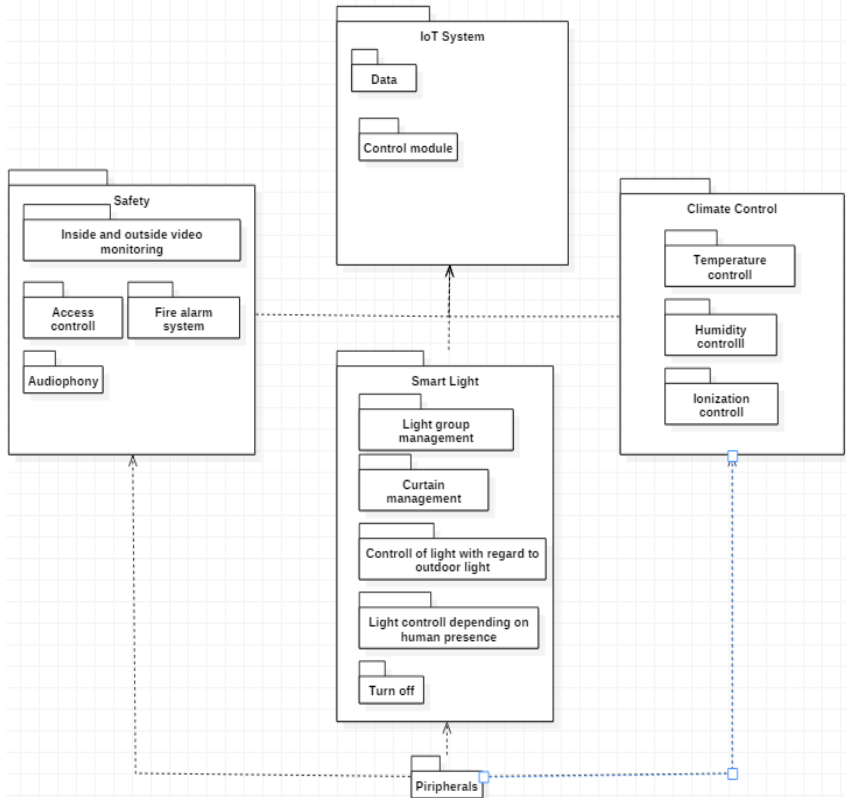


Fig. 3 – Package Diagram

In the packet diagram, all the functions of the IoT subsystem are distributed into several packet-areas, which are shown in the figure.

Formal automated verification of the subsystem's packet diagram entities is performed based on a complete test of coverage.

The set of complete simple paths can be different, it is important that the condition for covering all the entities of the packet diagram is met, as minimal as possible.

The upper complexity scores are:

$$C_{UML} = a*(n_e + n_e^2 + 3*n_r) = 3*(14 + 14^2 + 3*21) = 819; \quad (8)$$

$$C_{Path} = a*(n_e^2/2 + n_r) = 3*(14^2/2 + 21) = 357; \quad (9)$$

$$L_{Path} = 4+4+5+6+6+6+7 = 38; \quad (10)$$

$$\Delta_C = a*(n_e + n_e^2/2 + 2*n_r) = 3*(14 + 14^2/2 + 2*21) = 336. \quad (11)$$

Reducing the computational complexity of analysis and design through the use of an automated local verification script totaled 336 conventional UML analysis units.

3.2.3 Class diagram. At the next stage, it is advisable to create tables in the database (DB) and fill them with initial data. Before proceeding to this stage, it is necessary to determine which entities appear in the system. When determining the design of the IoT subsystem based on entities (objects), Data Driven Design (DDD) is usually used – a design based on the data model, the data model itself has an object nature displayed in the Document Object Model (DOM). Any system consists of objects and relationships between them. Object templates in the system are represented by classes. Some of the classes are implemented in the form of tables in the database and in the form of code, some – only in the form of code. The class diagram gives a description of the structure of the model entities (see Fig. 4 for a demonstration query class diagram with an explanation of the relationship-relationships).

Formal automated verification of class diagram entities is carried out on the basis of a full coverage test — interactive software selection-viewing of full navigation paths, with registration of results and their interactive testing for compliance with the standard (“Name - Properties - Methods” class templates, generalization relation, composition relation, relation) modified association).

Almost all class methods are used in each of the paths, however, to cover all methods, some repetition of paths will be required.

The upper complexity scores are:

$$C_{UML} = a*(n_e + n_e^2 + 3*n_r) = 3*(7 + 7^2 + 3*8) = 240; \quad (12)$$

$$C_{Path} = a*(n_e^2/2 + n_r) = 3*(7^2/2 + 8) = 99; \quad (13)$$

$$L_{Path} = 5 + 11 = 16; \quad (14)$$

$$\Delta_C = a*(n_e + n_e^2/2 + 2*n_r) = 3*(7 + 7^2/2 + 2*8) = 144. \quad (15)$$

Reducing the computational complexity of analysis and design amounted to 144 conventional UML analysis units.

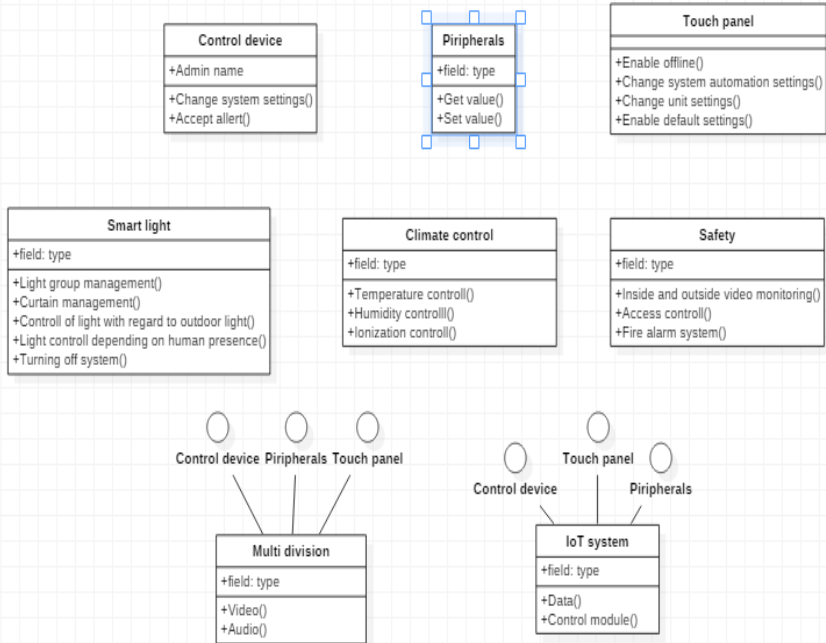


Fig. 4 – Class diagram of the subsystem lighting, temperature and air ionization

3.2.4 Sequence diagram. The UML sequence (scenario) diagram is used to describe the sequence of switching initiatives between process participants in a particular interaction session (scenarios) (see Figure 5). This refers to architectural objects - components, ordinary objects, subsystems, packages, etc.

Formal automated verification of entities of a certain sequence diagram is performed on the basis of the complete coverage test — an interactive programmatic selection-view of the only complete navigation path corresponding to the scenario, through association relations (calls to the corresponding methods) from the initial entity to the final one, covering all verification entities, and registering the results and their interactive testing, in particular, of property values against standards (architectural objects and relations of associations).

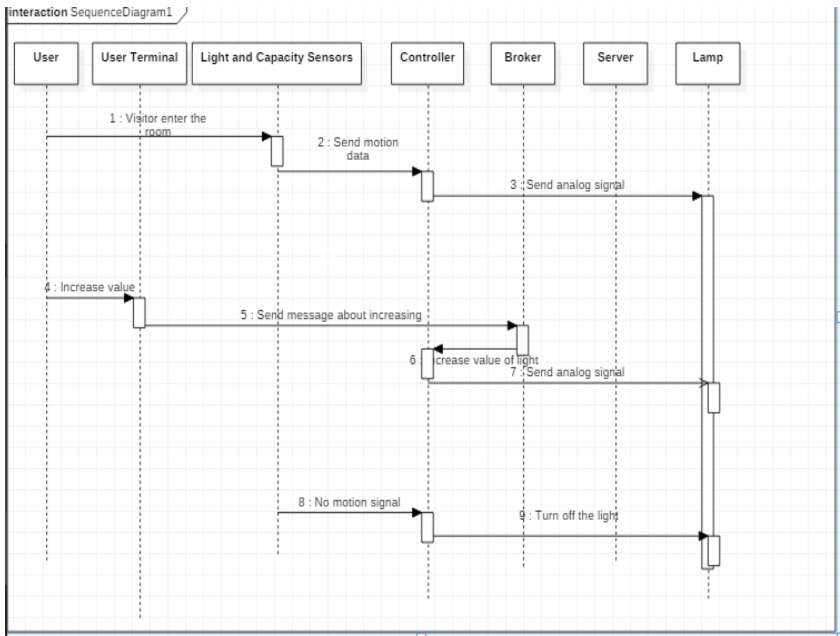


Fig. 5 – Sequence Diagram (Scenario) UML

There can be several sequence diagrams for the architecture of a certain system; each one makes sense in a complete representation — a passage from the initial object to the final one. However, their joint verification should only cover the set of all entities declared in them.

The upper complexity scores are:

$$C_{UML} = a*(n_e+n_e^2+3*n_r) = 3*(5+5^2+3*21) = 279; \quad (16)$$

$$C_{Path} = a*(n_e^2/2+n_r) = 3*(5^2/2 +21) = 102; \quad (17)$$

$$L_{Path} = 3+3+2+2 = 21; \quad (18)$$

$$\Delta_C = a*(n_e + n_e^2/2 + 2*n_r) = 3*(5+5^2/2 + 2*21) = 180. \quad (19)$$

Reducing the computational complexity of analysis and design amounted to 180 conventional UML analysis units.

4 Conclusion

UML diagrams are quite simple, and for the customer there is no need to master their full set. The developer's knowledge of all UML

diagrams is mandatory for reading, designing, verifying and drafting project documentation. For each task, use the appropriate UML diagram type. When the system is designed at the architectural-systemic, general structural and functional level, the stages of detailing and implementation begin, in this case other tools are relevant. In particular, these are systems that allow organizing work on a project.

5 Themes of individual tasks

1. Designing a smart system of operational individual lighting and air conditioning of a hotel room.
2. Designing a smart system of operational monitoring, suggestions for the selection and design by the buyer of goods in the store.
3. Designing a smart system of operational monitoring and suggestions for choosing a master of auto parts for repair.
4. Designing a smart operational lighting and air-conditioning library system.
5. Designing an interactive online theater box office.
6. Designing an online jewelry ordering system.
7. Designing an interactive smart clothing store.
8. Designing an interactive online poster and cinema help.

6 Recommended literature

1. Pallavi Sethi and Smruti R. Sarangi Internet of Things: Architectures, Protocols, and Applications. Journal of Electrical and Computer Engineering Volume 2017, Article ID 9324035, 25 p. <https://doi.org/10.1155/2017/9324035>
2. Dependable IoT for Human and Industry: Modeling, Architecting, Implementation, Vyacheslav Kharchenko, Ah Lian Kor, Andrzej Rucinski (Eds), River Publishers Series in Information Science and Technology, 2018, 450 p.
3. Tara Salman Networking Protocols and Standards for Internet of Things. – https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/index.html
4. Saber Talari, Miadreza Shafie-khah, Pierluigi Siano, Vincenzo Loia, Aurelio Tommasetti and João P. S. Catalão. A Review of Smart Cities Based on the Internet of Things Concept, Energies 2017, 10, 421. P. 23. <http://www.mdpi.com/1996-1073/10/4/421/pdf>

5. Rumbaugh James, “The unified modeling language reference manual,” 2-nd edition / James Rumbaugh, Ivar Jacobson, Grady Booch. Addison-Wesley on Web: <http://www.awprofessional.com> Available from: https://www.utdallas.edu/~chung/Fujitsu/UML_2.0/Rumbaugh--UML_2.0_Reference_CD.pdf.

6. Grady Booch James Rumbaugh Ivar Jacobson The Unified Modeling Language User Guide. Addison-Wesley Longman Inc., 1999.

7. For Programmer (UML diagrams in Visual Studio Feature Pack) <http://cc.ee.ntu.edu.tw/~farn/courses/BCC/NTUEE/2012.spring/vs.uml.instruction.pdf>

8. Y. Shoham, K. Leyton-Brown, “Multiagent Systems. Algorithmic, Game-Theoretic, and Logical Foundations,” Revision 1.1 / Shoham and Leyton-Brown, 2010. <http://www.masfoundations.org/mas.pdf>

9. A. Sugak, O. Martynyuk. Drozd. The Hybrid Agent Model of Behavioral Testing, International Journal of Computing, 2015, Volume 14, Issue 4, Ternopil, pp. 232-244.

10. O. Martynyuk, A. Sugak, D. Martynyuk, O. Drozd. Evolutionary Network of Testing of the Distributed Information Systems, Proceedings of the 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2017, Bucharest, Romania, pp. 888-893. <https://ieeexplore.ieee.org/document/8095215>

11. Anduo Wang Formal Analysis of Network Protocols / University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-10-16. 2010. https://repository.upenn.edu/cgi/viewcontent.cgi?article=1970&context=cis_reports

12. D. Câmara, “Formal Verification of Communication Protocols for Wireless Networks,” Belo Horizonte, 2009. http://www.eurecom.fr/~camara/files/ThesisCamara_FormalVerification.pdf

13. A. Boyarchuk, V. Kharchenko, O. Illiashenko, D. Maevsky, C. Phillips, A. Plakhteev, L. Vystorobska. Internet of Things for Human and Industry Applications: ALIOT Based Curriculum. In book: Dependable IoT for Human and Industry: Modeling, Architecting, Implementation, Vyacheslav Kharchenko, Ah Lian Kor, Andrzej Rucinski (Eds.), River Publishers Series in Information Science and Technology, 2018.

Laboratory work 2

Research of IoT systems in behavior by using Color Petri Net Tools

1 The purpose and objectives

The purpose of these guidelines is to consolidate and supplement the lecture material, as well as develop students' skills and abilities to understand existing IoT technologies and apply them to specific functions, procedures and scenarios, as well as the ability to design complete IoT systems. The basic theoretical and reference provisions are given, as well as general and individual tasks for independent work of students. In the course of independent work preceding laboratory work, the study of lecture and additional material is carried out, preparation and analysis of solutions to problems of laboratory work.

At the beginning of each laboratory work an individual control of theoretical readiness is made, the result of which is admission to work. In the course of work, each student performs the task on time and draws up a protocol with the results of work. Control of knowledge, protection of a working program and protection of the protocol are made for each student individually. The goal of the work is to study the possibilities of the structural-functional design of the IoT system when using Petri nets and the CPN Tools tool environment.

1.1 Teaching tasks:

- The study of the basic principles of structural-functional, behavioral analysis with the use of Petri nets;
- Familiarization with the methodology of structural-functional, behavioral, event-time design;
- Familiarization with the fundamentals of model verification and redesign when debugging system projects, in particular, based on the use of intuitive- “manual”, re-selection automatic testing, as well as verification of the graph of achievable markings, construction and analysis of position and transition invariants in the “spin” method;
- Familiarization with the capabilities of Petri nets, the principles of operation and the basics of using the CPN Tools tool environment, which provides a formal specification and performs automated structural-functional and event-time modeling and verification of IoT projects.

1.2 Practical tasks

Practical tasks for IoT systems are solved based on the use of Star UML, CPN Tools instrumental environments and include:

- visual construction of input UML diagrams of activities and states implemented at the IoT application layer in the Star UML environment editor;
- visual construction of Petri nets based on input UML diagrams of activities and states in the CPN Tools editor;
- automated static and dynamic verification of Petri nets and its evaluation based on the CPN Tools with step-by-step and end-to-end modeling, as well as determining the achievable markup;
- intuitive “manual”, repeated synthesis and evaluation of behavioral tests for logic (link structure) of UML diagrams and Petri nets covering their start-stop paths.

1.3 Research tasks

Research tasks for IoT systems that are solved using Petri nets for their construction, behavioral analysis and verification with the implementation of redesign, if necessary, include intuitive- “manual”, subject-behavioral:

- analysis of the correctness of Petri nets, in particular, the determination of basic and complex properties;
- optimization of the structural and computational complexity of Petri nets, the length of their paths and cycles;
- optimization of the length, multiplicity and completeness of tests for the logic of Petri nets.

2 Preparation for laboratory work

Preparation for laboratory work involves the need to:

- understand the purpose and objectives of the laboratory work;
- study of the theoretical material given in the description and additional sources;
- selection of the IoT system design option for laboratory work (in consultation with the teacher);
- determine the program (scenario) of the laboratory work;
- check the installation of the necessary software (CPN Tools).

3 Perform of laboratory work

3.1 The laboratory work order

Stage 1. In the first stage, the following steps are performed:

1. Construction of formal models of the architecture selected according to the variant of the IoT subsystem task - activity diagram and state diagram.

2. The upper estimate of the computational complexity of the analysis and the construction of UML diagrams is performed using a simplified formula for basic entities and relations.

$$C_{UML} = a*(n_e+n_e^2+3*n_r), \quad (1)$$

where n_e is the number of classifiers of the diagram entities; n_e^2 is the number of cells in a square matrix of possible relationships between entities; n_r is the number of real assigned ratios of the diagram, the multiplier “3” means the need to consider both the relation r itself and the two entities e_1 and e_2 incident to it; a - conditional coefficient of abstraction level (for technical specifications - 3, structural-functional - 2, object-component - 2, event-time - 1, automaton-algorithmic - 1).

1. Building a script for automated local verification of specifications (properties) based on an intuitive, “manual”, full-choice testing method for constructing a path covering all entities for each of the selected diagrams (at least three different out of five) and determining the path length.

2. The upper estimate of the computational complexity of the covering path (including its construction) using the simplified formula:

$$C_{Path} = a*(n_e^2/2+n_r). \quad (2)$$

1. The upper estimate of the reduced computational complexity of the design due to the use of automated local verification script in comparison with the case of re-design in case of an error, when the computational complexity of the analysis and construction of the UML diagram is doubled, using the simplified formula:

$$C_{\Delta C} = 2*C_{UML} - (C_{UML}+C_{Path}) = a*(n_e+ n_e^2/2 +2*n_r). \quad (3)$$

Stage 2. In the second stage, the following steps are performed:

1. Visual construction of Petri nets in the CPN Tools editor according to the variant of the IoT system job based on the input UML diagrams of activities and stage states1.

2. Determination of the upper estimate of the computational complexity of the analysis and the construction of Petri nets using a simplified formula for their entities – positions, transitions, chips and relations for them:

$$C_{\text{Petri}} = a \cdot (n_e + n_q + 4 \cdot n_r), \quad (4)$$

where n_{ep} is the number of position classifiers, n_{et} is the number of transition classifiers, n_{em} is the number of chip token classifiers, $n_e = n_{ep} + n_{et} + n_{em}$; $n_q = n_{ep} \cdot n_{et} \cdot n_{em}$ is the number of cells in the three-dimensional matrix of possible relationships between entities; n_r is the number of real assigned ratios of the diagram, the multiplier "4" means that it is necessary to consider both the relation r itself and the three incident entities e_{1p} , e_{2t} , e_{3m} ; a - conditional coefficient of abstraction level (for technical specifications - 3, structural-functional - 2, object-component - 2, event-time - 1, automaton-algorithmic - 1).

3. Intuitive-manual building of scenarios of static and dynamic verification of selected basic and complex properties of the Petri net based on CPN Tools tools for:

- a) step-by-step simulation;
- b) end-to-end modeling;
- c) constructing a graph of attainable markings;
- d) matrix definition of position and transition invariants (outside of CPN Tools).

4. Intuitive “manual” construction of the test, as a walk-around covering all the positions and transitions of the Petri net based on recurring behavioral testing, and determining the length of this path.

5. The upper estimate of the computational complexity of the covering path (including its construction) using the simplified formula:

$$C_{\text{PetriPath}} = a \cdot (n_q / 3 + n_r). \quad (5)$$

6. The upper estimate of the reduced computational complexity of the design due to the use of the automated local verification scenario in comparison with the case of re-design in case of an error when the computational complexity of analyzing and building the Petri net doubles, using the simplified formula:

$$C_{\Delta C} = 2 \cdot C_{\text{Petri}} - (C_{\text{Petri}} + C_{\text{PetriPath}}) = a \cdot (n_e + 2n_q / 3 + 3 \cdot n_r). \quad (6)$$

Stage 3. At the third stage, the following intuitive “manual”, subject-behavioral research steps are performed with the implementation of redesign, if necessary:

1. A detailed analysis of the correctness of Petri nets, in particular, the definition of selected properties:

a) basic: lack of static and dynamic locks; completeness; uniqueness of the correspondence of states; lack of redundancy; limitations; self-synchronization;

b) complex: partial / complete correctness; liveness; security.

2. Optimization of the structural and computational complexity of Petri nets, the length of their paths and cycles.

3. Optimization of the length, multiplicity and completeness of tests for the logic of Petri nets.

3.2 Examples of the laboratory work execution

Perform Phase 1. Step-by-step construction of the IoT system activity and state diagrams is performed.

To represent the architecture of the IoT subsystem using Petri nets, we need 2 types of diagrams. When designing a lighting system, these types of diagrams allow you to see it from the user's side.

A method based on the construction of separate scenarios (one of the program variants) of the system operation is proposed. We show how this method is implemented on the example of scenarios with a positive outcome of user interaction with the lighting system.

3.2.1 Activity Chart. The first step of the IoT subsystem description is the actions that can be performed by each user, a convenient type of diagrams for this is an activity diagram, and you can also use a simpler and more familiar language of flowcharts – graph diagrams – instead. We describe the main user actions when working with the system.

The first thing that is required of the user is to enter the premises, thereby causing the motion sensor tripped to work, then enter the flow time of the day (Input the time of day). Next, the system includes the corresponding mode (day / night). After that, the user sets a valid value for the light level for the specified mode. Further, the user is given a choice: to continue working with the system or not. In case of agreement, the system returns to the time setting.

Imagine the logic of work in the form of activity diagrams (see Fig. 1).

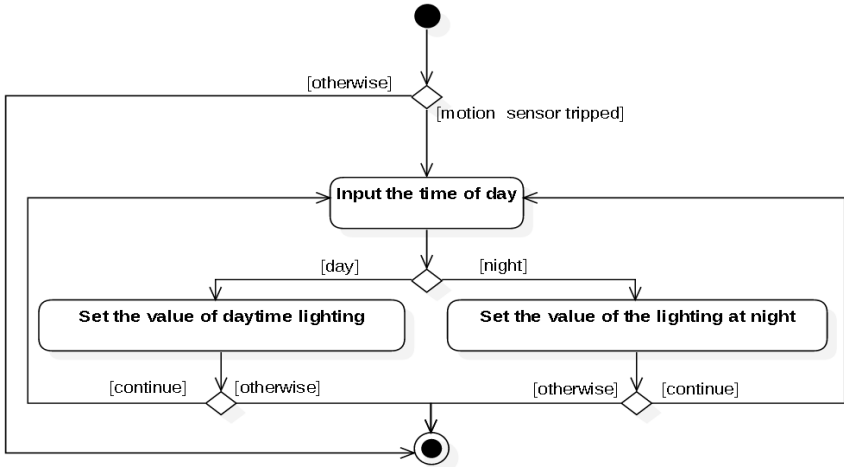


Fig. 1 – Activity diagram of the lighting system

Entity verification for the diagram:

- Actions: enter the stream time of day (Input the time of day), set the allowable value of the lighting level for day mode, and set the allowable value of the level of light for night mode.

- Relations-transitions: start → branching; branching → time; time → branching; branching → values by day; values by day → branching; branching → time; branching → end; branching → values at night; values at night → branching; branching → time; branching → end; branching → end; branching → end.

- Constructions for registration transitions: branching; compound.

The verification of entities of a simple diagram of the illumination system activity is performed on the basis of a complete test of coverage — an interactive program enumeration of complete simple paths from start to final vertex, covering all verification entities.

The set of complete simple paths includes six paths:

- (start, branch, time set, branch, set by day, branch, end) is a simple way;

- (start, branch, set time, branch, set values at night, branch, end) is a simple way;

- (start, branch, set time, branch, set values by day, branch, (set time, branch, set values by day, branch) end) - a simple loop is nested.
- (start, branch, set time, branch, set values by day, branch, (set time, branch, set values at night, branch) end) - a simple loop is nested.
- (start, branch, set time, branch, set values at night, branch, (set time, branch, set values at night, branch) end) - a simple loop is nested.
- (start, branch, set time, branch, set values at night, branch, (set time, branch, set values by day, branch) end) - a simple loop is nested.

Consequently, the combinatorial-enumerated number of possible paths that differ in the input data is six.

The upper complexity estimates are of the form:

$$C_{UML} = a*(n_e + n_e^2 + 3*n_r) = 3*(7 + 7^2 + 3*6) = 222; \quad (4)$$

$$C_{Path} = a*(n_e^2/2 + n_r) = 3*(7^2/2 + 6) = 91.5; \quad (5)$$

$$L_{Path} = 7 + 9 + 6 = 22; \quad (6)$$

$$\Delta_C = a*(n_e + n_e^2/2 + 2*n_r) = 3*(7 + 7^2/2 + 2*6) = 130.5. \quad (7)$$

Reducing the computational complexity of analysis and design through the use of an automated local verification script amounted to 130.5 conventional units of analysis.

3.2.2 State diagrams. State diagrams (statechart diagrams) show the possible states of individual components or the system as a whole, transitions between them in response to any events and actions performed during this.

States can be arranged hierarchically; they can be decomposed into parallel substates. A subsystem can have global (within its framework) variables storing some data. The values of these variables are common parts of all depicted states.

The peculiarity of the state diagram is the use of a modified association of state transitions, not transitions — message exchanges, as well as the possibility of an automatic representation of all scenarios as relations. And also, in the temporal, rather than spatial, character of the representation of transition relations.

Verification of the state diagram, checking all the scenarios or their essential part, is more complex (up to NP-complex for non-trivial diagrams with tens and hundreds of entities). Reducing the NP-complexity can be achieved through the use of behavioral testing and decomposition methods.

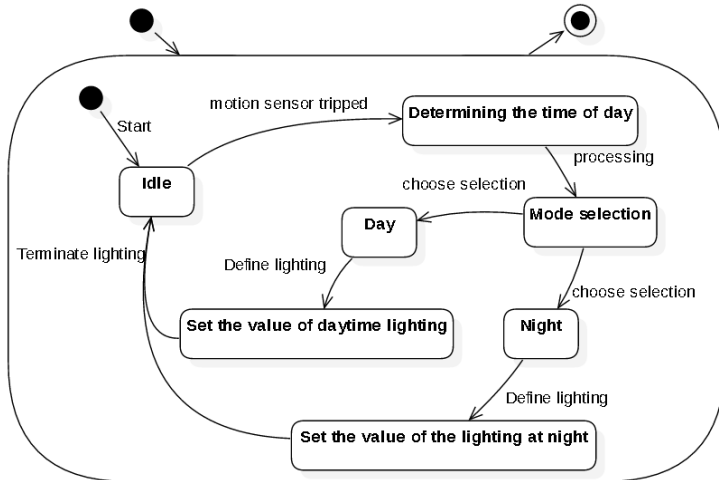


Fig. 2 – Lighting system state chart diagram

Entities verification of the state diagram:

– States: Idle, Determining the time of day, Mode selection, Day, Night, Set the value of daytime lighting, Set the value of the lighting at night, StartOut, End, StartIn.

– Relations of the modified association - state transitions: StartOut \rightarrow start() \rightarrow system; system \rightarrow start() \rightarrow StartIn; StartIn \rightarrow start() \rightarrow Idle; Idle \rightarrow motion sensor tripped \rightarrow Determining the time of day; Determining the time of day \rightarrow processing \rightarrow Mode selection; Mode selection \rightarrow choose selection \rightarrow Day; Day \rightarrow Define lighting \rightarrow Set the value of daytime lighting; Set the value of daytime lighting \rightarrow Terminate lighting \rightarrow Idle; Mode selection \rightarrow choose selection \rightarrow Night; Night \rightarrow Define lighting \rightarrow Set the value of the lighting at night; Set the value of the lighting at night \rightarrow Terminate lighting \rightarrow Idle; StartPage \rightarrow back() \rightarrow system; system \rightarrow back() \rightarrow End.

A formal automated verification of entities of a given, relatively simple state diagram is performed on the basis of a complete transition coverage test — one of the possible Eulerian paths through association relations (transitions — calls to the corresponding methods) from the initial Start entity to the final End, covering all verification entities, with registration of results and their interactive verification, in particular, of the property values against the standards (architectural objects and association relations).

The upper complexity estimates are of the form:

$$C_{UML} = a*(n_e+n_e^2+3*n_r) = 3*(10+10^2+3*11) = 429; \quad (8)$$

$$C_{Path} = a*(n_e^2/2+n_r) = 3*(10^2/2 + 11) = 183; \quad (9)$$

$$L_{Path} = 1+2+9 = 12; \quad (10)$$

$$\Delta_C = a*(n_e + n_e^2/2 + 2*n_r) = 3*(10+10^2/2 + 2*11) = 246. \quad (11)$$

Reducing the computational complexity of analysis and design through the use of an automated local verification script amounted to 246 conventional units of analysis.

3.2.3 Petri net construction. To check the correctness of the construction and identification of logical errors, we translate the diagrams into the advising Petri net (Figure 3). For the constructed Petri net, the transitions correspond to the actions on the activity diagram.

Let us analyze the resulting network by examining the state space, some of which are analyzed by the floating line method. The end states of the algorithm operation are marked in black: the chips hit the Finishing work place. Since it is proposed to analyze the state space by individual parts and draw a conclusion about the correctness of the network design using them.

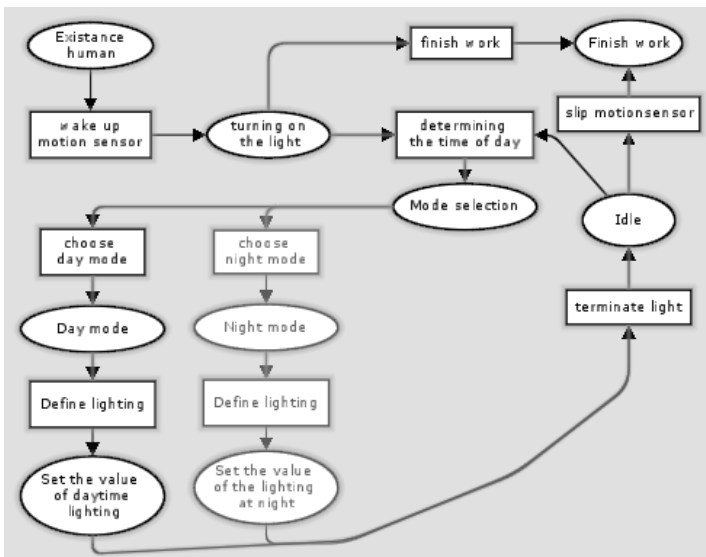


Fig. 3 – Petri net of turning user interaction with the lighting system

Comparing the detailed graph and the graph generated using the CPNTools software environment, one can draw a conclusion about their similarity and the possibility of “gluing” the graphs (separate scenarios of the same algorithm). Thus, it is possible to analyze the state graph by examining individual scenarios obtained from the general algorithm (activity diagram) and then combining them if necessary. Based on a common Petri net, a Petri Subject Net is built in the CPN Tools (TINA) environment (see. Fig. 4).

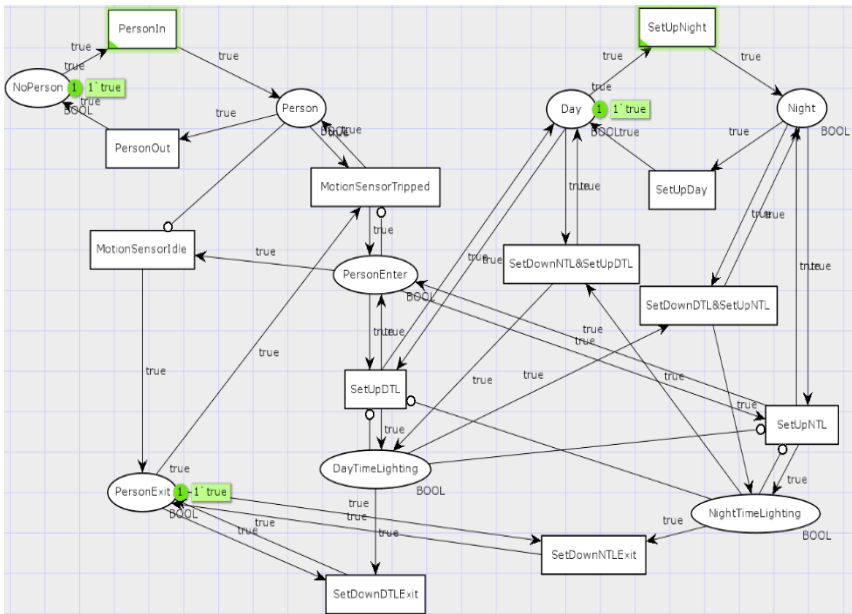


Fig. 4 – Subject network of Petri user interaction with the lighting system

This network can be divided into 3 parts. The first, top left (NoPerson, Peerson, PersonIn, PersonOut), is responsible for having a person as a physical object. The second one from the top right (Day, Night, SetUpDay, SetUpNight), is responsible for changing the time of day. The third part, all other blocks, are responsible for the operation of the system, or rather, its reaction to the movement of chips in the first two parts.

3.3 Findings. Thus, a method for constructing system operation scenarios obtained from an activity diagram is demonstrated using the example of user interaction with a lighting system. In this example, two scenarios of work, modeled on the basis of UML and Petri nets were studied, studied in CPN Tools (TINA) environments by constructing and analyzing step and through graphs of markings and a graph of achievable markings, as well as defining invariants of positions and transitions. The obtained results confirmed the presence of the basic required properties of the correctness of the object network of Petri and testify to the expediency of studying and practical mastering the proposed models of Petri nets and the method of their analysis.

4 Conclusion

Petri nets are a tool for mathematical modeling and research of complex systems. In one of the approaches to the design and analysis of systems, Petri nets are used as an auxiliary analysis tool. Here, to build the system, conventional design methods are used. Then the constructed system is modeled by the Petri net, and the model is analyzed. If in the course of the analysis a flaw is found in the project, then the project is modified to eliminate them. The modified project is then modeled and analyzed again. This cycle is repeated until the analysis is successful. When the system is designed, the implementation phase begins, when other tools are relevant. In particular, these are systems that allow organizing work on a project.

5 Report content requirements

The report should include:

- purpose, individual task and program of laboratory work
- source code of the program in the selected programming language
- report on the performance of each task.
- conclusions on the results of completed tasks.

6 Themes of individual tasks

1. Designing a smart system of operational individual lighting and air conditioning of a hotel room.

2. Designing a smart system of operational monitoring, suggestions for the selection and design by the buyer of goods in the

store.

3. Designing a smart system of operational monitoring and suggestions for choosing a master of auto parts for repair.

4. Designing a smart operational lighting and air-conditioning library system.

5. Designing an interactive online theater box office.

6. Designing an online online jewelry ordering system.

7. Designing an interactive smart clothing store.

7 Recommended literature

1. Pallavi Sethi and Smruti R. Sarangi Internet of Things: Architectures, Protocols, and Applications // Journal of Electrical and Computer Engineering Volume 2017, Article ID 9324035, 25 p. <https://doi.org/10.1155/2017/9324035>

2. Dependable IoT for Human and Industry: Modeling, Architecting, Implementation, Vyacheslav Kharchenko, Ah Lian Kor, Andrzej Rucinski (Eds), River Publishers Series in Information Science and Technology, 2018, 450 p.

3. Tara Salman Networking Protocols and Standards for Internet of Things. – https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/index.html

4. Saber Talari, Miadreza Shafie-khah, Pierluigi Siano, Vincenzo Loia, Aurelio Tommasetti and João P. S. Catalão. A Review of Smart Cities Based on the Internet of Things Concept, Energies 2017, 10, 421. – P. 23. <http://www.mdpi.com/1996-1073/10/4/421/pdf>

5. Rumbaugh James. The unified modeling language reference manual – 2-nd edition / James Rumbaugh, Ivar Jacobson, Grady Booch. Addison-Wesley on Web: <http://www.awprofessional.com> Available from: https://www.utdallas.edu/~chung/Fujitsu/UML_2.0/Rumbaugh--UML_2.0_Reference_CD.pdf.

6. Jorg Desel, Javier Esparza Free Choice Petri Nets // Cambridge University Press, Cambridge, 1995. – P. 256. Available from: <https://www7.in.tum.de/~esparza/fcbook-middle.pdf>

7. Dmitry A. Zaitsev, Tatiana R. Shmeleva. Modeling with Colored Petri Nets: Specification, Verification, and Performance Evaluation of Systems Automated Systems in the Aviation and Aerospace Industries Copyright. 2019. P. 27.

8. A. Sugak, O. Martynyuk, O. Drozd. The Hybrid Agent Model of Behavioral Testing, *International Journal of Computing*, 2015, Volume 14, Issue 4, Ternopil, P. 232-244.

9. O. Martynyuk, A. Sugak, D. Martynyuk, O. Drozd. Evolutionary Network of Testing of the Distributed Information Systems, *Proceedings of the 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 2017, Bucharest, Romania, P. 888-893. <https://ieeexplore.ieee.org/document/8095215>

10. Michael Westergaard CPN Tools / Eindhoven, 2010. – P. 46. <https://westergaard.eu/wp-content/uploads/2010/09/CPN-Tools.pdf>

11. Anduo Wang Formal Analysis of Network Protocols / University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-10-16. 2010. – P. 32. https://repository.upenn.edu/cgi/viewcontent.cgi?article=1970&context=cis_reports

12. A. Boyarchuk, V. Kharchenko, O. Illiashenko, D. Maevsky, C. Phillips, A. Plakhteev, L. Vystorobska. Internet of Things for Human and Industry Applications: ALIOT Based Curriculum. In book: *Dependable IoT for Human and Industry: Modeling, Architecting, Implementation*, Vyacheslav Kharchenko, Ah Lian Kor, Andrzej Rucinski (Eds.), River Publishers Series in Information Science and Technology, 2018.

Laboratory work 3

Modeling Internet of Things systems using LTL temporal logic at the stages of their architectural design

1 The purpose and objectives

The purpose of these guidelines is to consolidate and supplement the lecture material, as well as develop students' skills and abilities to understand existing IoT technologies and apply them to specific functions, procedures and scenarios, as well as the ability to design complete IoT systems. For the tasks of laboratory work, the basic theoretical and reference provisions are given, as well as general and individual tasks for independent work of students. In the course of independent work preceding laboratory work, the study of lecture and additional material is carried out, the preparation of tasks for laboratory work. At the beginning of each laboratory work an individual control of theoretical readiness is made, the result of which is admission to work. In the course of work, each student performs the task on time and draws up a protocol with the results of work. Control of knowledge, protection of a working program and protection of the protocol are made for each student individually.

The goal of the work is to study the possibilities of structural-functional, behavioral, event-time design of the IoT system using the LTL temporal logic and the SPIN / XSPIN tool environment.

1.1 Teaching tasks:

- studying the principles of causal, logical-temporal design, the basics of logical-temporal inference, proof and verification of causal, logical-temporal temporal models, redesign when debugging system projects, in particular, based on the use of LTL in derivation and proof in synchronization processes;
- familiarization with the possibilities, principles of operation and the basics of using the SPIN / XSPIN tool environment, which provides a formal specification, a logical-temporal derivation of LTL expressions and verification of system designs.

1.2 Practical tasks

Practical tasks for IoT systems are solved based on the use of SPIN/XSPIN tool environments and include:

- determination of entities and relations of systems at the IoT application level, selection of conditions, events, actions and functions implemented in the system, and determination of cause-effect, logical-temporal relations for them;
- building a system of LTL expressions for a logical-temporal representation of a cause-effect relationship for the conditions, events, actions and functions implemented in the system;
- building a system of LTL expressions for logical-temporal inference for atomic fragments of the processes implemented in the functioning of the system;
- constructing a system of LTL expressions for a logical-temporal output in synchronization of atomic fragments of processes among themselves in accordance with the main scenarios of the system operation;
- construction and verification of Kripke structures for the LTL expression system for synchronizing atomic fragments of processes in the system operation scenarios;
- building and verifying the specifications of Promela and Buchi automata for the LTL expression system and Kripke’s synchronization structures for atomic fragments of processes in the system operation scenarios;
- automated verification of Promela specifications based on SPIN (XSPIN, ISPIN, JSPIN) tools.

1.3 Research tasks

The research tasks for IoT systems that are solved using LTL temporal logic for building, behavioral, causal, event-time analysis and verification with redesign, if necessary, include analytic and intuitive construction and analysis of the correctness properties of the LTL expressions when synchronization of processes implemented in the system.

2. Preparation for laboratory work

Preparation for laboratory work involves the need to:

- understand the purpose and objectives of the laboratory work;
- study of the theoretical material given in the description and additional sources;

- selection of the IoT system design option for laboratory work (in consultation with the teacher);
- determine the program (scenario) of the laboratory work;
- check the installation of the necessary software.

3. Laboratory work execution

3.1 The order of the laboratory work performance

Stage 1. In the first stage, the following steps are performed:

1. Construction of models of the architecture selected according to the variant of the IoT system task, in particular, UML diagrams of interaction, sequences, activities and states.

2. Selection of basic conditions, events (condition structures), actions (function structures) and functions for basic entities and relations of UML diagrams of the IoT system.

3. Determination of cause-effect, logical-temporal relations between conditions, events, actions and functions, as well as their input-output signals and the construction of a matrix representation for these relations.

Stage 2. In the second stage, the following steps are performed:

1. Building a system of LTL temporal logic expressions based on a causal, logical-temporal relationship built on stage 1 between conditions, events, actions and functions, input-output signals of the IoT system.

2. Building Kripke structures for the LTL expression system for synchronizing atomic fragments of processes and their synchronization, which are implemented in the functioning of the system in accordance with its main work scenarios.

Stage 3. In the third stage, the following steps are performed:

1. Construction of Buchi automata and Promela specifications for the LTL synchronization expression system of processes implemented in the system.

2. Automated verification of Buchi and Promela automata specifications based on the SPIN (XSPIN, ISPIN, JSPIN) verifier.

Stage 4. In the fourth stage, the following steps are performed:

1. Construction and analysis of a system of LTL expressions of a logic-time proof of the properties of correctness and reachability of subject properties-services in the functioning of the system during the synchronization of processes implemented in its functioning.

3.2 Examples of the tasks

3.2.1 Temporary verification. Building LTL Expressions.

Sensor and controller interaction. A unidirectional channel between communicating processes in the network: a sensor (Sender - S) and a controller (Receiver - R). S is equipped with an output buffer ($S.out$), and R with an input buffer ($R.in$) of unlimited capacity.

If S sends a message m to R , then it puts the message in its output buffer $S.out$. The output buffer $S.out$ and the input buffer $R.in$ are connected by a unidirectional channel.

R accepts messages by deleting them from the $R.in$ input buffer. Messages are identically identified, the set of atomic sentences $AP = \{m \in S.out, m \in R.in\}$, where m is the message. It is implicitly assumed that all properties are formulated for any message m (universal quantification with respect to m does not affect the solvability of the satisfiability problem, assuming that the number of messages is finite). The $S.out$ and $R.in$ buffers do not distort or lose the message, the message is not in the buffer indefinitely.

Formalization of informal channel requirements using LTL:

“The message cannot be in both buffers at the same time”:

$$\mathbf{G} \neg(m \in S.out \wedge m \in R.in).$$

"The channel does not lose the message." If the channel does not lose the message, this means that messages placed in $S.out$ will ultimately be delivered to $R.in$:

$$\mathbf{G}(m \in S.out \rightarrow \mathbf{F}(m \in R.in)).$$

If the uniqueness of messages is not implied, then this property is not enough, since if, for example, two copies of m were transmitted and only the last one was received, then this situation would satisfy this property. The uniqueness of messages is a prerequisite for the specification of requirements in temporal logic for systems that transmit messages. If we assume the validity of the previous property, then we can equivalently write:

$$\mathbf{G}(m \in S.out \rightarrow \mathbf{XF}(m \in R.in)),$$

since m cannot be in $S.out$ and $R.in$ at the same time.

"The channel maintains order." This means that if m and then m' are transferred to S in its output buffer $S.out$, then m will be taken R before m' :

$$\mathbf{G}[m \in S.out \wedge \neg m' \in S.out \wedge \mathbf{F}(m' \in S.out) \rightarrow \rightarrow \mathbf{F}(m \in R.in \wedge \neg m' \in R.in \wedge \mathbf{F}(m' \in R.in))].$$

In the premise, the conjunction $\neg m' \in S.out$ is required in order to specify that m' is sent to $S.out$ after m . $\mathbf{F}(m' \in S.out)$ does not exclude that m' is in the sender's buffer when m is in $S.out$.

The channel does not generate messages spontaneously. This means that any m in $R.in$ must be sent in advance by the sender S .

Using the precedence operator F , this can be formalized as follows: $\mathbf{G}[(m \in R.in) \rightarrow \mathbf{F}(m \in S.out)]$.

If there are no precedence statements, you can use the U statement: $\mathbf{G}[\neg(m \in R.in) \mathbf{U}(m \in S.out)]$.

3.2.2 Check models for LTL. The task of checking models for linear temporal logic is formulated in terms of Kripke models.

The task of verifying the LTL formula for the Kripke model is to determine whether the formula is executed on all paths (or, equivalently, whether there are paths on which the formula is not performed). According to a given formula f and the Kripke model M , a labeled LTL table T is constructed for formula f . This table is a Kripke model with a fixed set of initial states consisting of all paths that satisfy the formula f .

Reduction of a formula to a divided normal form

For convenience, the formula is initially reduced to a form with close negations (which are only atomic sentences).

For LTL expressions:

It was: $\neg(f \wedge (\mathbf{F} g \vee \mathbf{X}(h \mathbf{W} r)))$.

It became: $\neg f \vee \mathbf{G} \neg g \wedge \mathbf{X}(\neg r \mathbf{U} (\neg h \wedge \neg r))$.

This process is required. The algorithm also works for general formulas. After this, the formula is decomposed:

- a new atomic variable p_i is introduced for each of its inner subformulas f_i , in which the last operation (with the lowest priority) is temporal;

- is added to the formula new conjunct $\mathbf{G}(p_i \rightarrow f_i)$.

A conjunct indicates that whenever p_i is true, the formula f_i is also true, that is, the variable p_i "encodes" f_i .

In the formula, the subformulas f_i are replaced by the atomic variables p_i corresponding to them. It was: $\neg f \vee \mathbf{G} \neg g \wedge \mathbf{X}(\neg r \mathbf{U} (\neg h \wedge \neg r))$.

It became: $\mathbf{G}(p_1 \rightarrow \mathbf{G} \neg g) \wedge \mathbf{G}(p_2 \rightarrow (\neg r \mathbf{U}(\neg h \wedge \neg r))) \wedge \mathbf{G}(p_3 \rightarrow \mathbf{X} p_2) \wedge (\neg f \vee p_1 \wedge p_3)$.

There are three atomic variables added: one for the subformula $\mathbf{G}\neg g$, one for the subformula $(\neg r \mathbf{U}(\neg h \wedge \neg r))$ and one for the subformula $\mathbf{X}(r \mathbf{U}(\neg h \wedge \neg r))$.

After that, all such subformulas were replaced by the corresponding atomic variables. The meaning of the formula has not changed. At this stage, the formula in question is a conjunction of one formula that does not contain temporal operators, with several formulas of the form $\mathbf{G}(p_i \rightarrow \mathbf{T})$, where \mathbf{T} is the application of a certain temporal operator to one or two operands, and these operands do not contain other temporal operators.

Now it is required to get rid of conjunctions.

1. Get rid of $\mathbf{G}(p_i \rightarrow \mathbf{G} f_i)$. For each such conjunct, a new atomic variable r_i is introduced and this conjunctive is replaced by the formula:

– $\mathbf{G}(p_i \rightarrow r_i) \wedge \mathbf{G}(r_i \rightarrow f_i) \wedge \mathbf{G}(r_i \rightarrow \mathbf{X} r_i)$.

– The meaning of the r_i variable is that it gives a “promise” that f_i will always be true.

2. Getting rid of $\mathbf{G}(p_i \rightarrow (f_i \mathbf{W} g_i))$. For each such conjunct, a new atomic variable r_i is introduced and this conjunctive is replaced by the formula:

– $\mathbf{G}(p_i \rightarrow g_i \vee f_i \wedge r_i) \wedge \mathbf{G}(r_i \rightarrow \mathbf{X}(g_i \vee f_i \wedge r_i))$.

– The variable r_i makes such a promise: if g_i “now” is not yet fulfilled, then, firstly, f_i is now fulfilled, and secondly, the same is guaranteed in the next step.

3. Getting rid of $\mathbf{G}(p_i \rightarrow (f_i \mathbf{U} g_i))$. This stage differs from the previous one only in the fact that the g_i property will sometime come to be guaranteed. For each such conjunct, a new atomic variable r_i is introduced and this conjunctive is replaced by the formula:

– $\mathbf{G}(p_i \rightarrow g_i \vee f_i \wedge r_i) \wedge \mathbf{G}(r_i \rightarrow \mathbf{X}(g_i \vee f_i \wedge r_i)) \wedge \mathbf{G}(p_i \rightarrow \mathbf{F} g_i)$.

– The formula differs from the previous one only by the addition of a conjunct. $\mathbf{G}(p_i \rightarrow \mathbf{F} g_i)$. In the next step, this conjunct will be liquidated.

4. Getting rid of $\mathbf{G}(p_i \rightarrow \mathbf{F} f_i)$. For each such conjunct, a new atomic variable r_i is introduced and this conjunctive is replaced by the formula:

- $\mathbf{G}(p_i \rightarrow (\neg f_i \rightarrow r_i)) \wedge \mathbf{G}(r_i \rightarrow \mathbf{X}(\neg f_i \rightarrow r_i)) \wedge \mathbf{GF} \neg r_i$.
- The meaning of the variable r_i is as follows: it is true if f_i has not yet come (after activating p_i), and the conjunction $\mathbf{GF} \neg r_i$ “promises” that after any activation of p_i it (r_i) will ever become incorrect and, accordingly, f_i will be true.

It was: $\mathbf{G}(p_1 \rightarrow \mathbf{G} \neg g)$.

It became: $\mathbf{G}(p_1 \rightarrow r_1) \wedge \mathbf{G}(r_1 \rightarrow \neg g) \wedge \mathbf{G}(r_1 \rightarrow \mathbf{X} r_1)$.

It was: $\mathbf{G}(p_2 \rightarrow (\neg r \mathbf{U} (\neg h \wedge \neg r)))$.

It became: $\mathbf{G}(p_2 \rightarrow (\neg h \wedge \neg r \vee \neg r \wedge r_2)) \wedge \mathbf{G}(r_2 \rightarrow \mathbf{X}(\neg h \wedge \neg r \vee \neg r \wedge r_2)) \wedge \mathbf{G}(p_2 \rightarrow \mathbf{F}(\neg h \wedge \neg r))$.

It was: $\mathbf{G}(p_2 \rightarrow \mathbf{F}(\neg h \wedge \neg r))$.

It became: $\mathbf{G}(p_2 \rightarrow (\neg(\neg h \wedge \neg r) \rightarrow r_3)) \wedge \mathbf{G}(r_3 \rightarrow \mathbf{X}(\neg(\neg h \wedge \neg r) \rightarrow r_3)) \wedge \mathbf{GF} \neg r_3$.

After performing these operations, all operators \mathbf{G} and propositional sub-formulas are put out of the bracket (they can also be simplified). The original formula will take the form:

$p \wedge \mathbf{G}(q \wedge (r_1 \rightarrow \mathbf{X} s_1) \wedge \dots \wedge (r_n \rightarrow \mathbf{X} s_n) \wedge \mathbf{F} t_1 \wedge \dots \wedge \mathbf{F} t_m)$,

where $p, q, r_1, \dots, r_n, s_1, \dots, s_n, t_1, \dots, t_m$ are propositional subformulas that do not contain temporal operators.

The meaning of the formulas:

- formula p "is valid only at the starting moment of time";
- formula q "always works";
- the formulas r_i and s_i (for i from 1 to n) determine the condition for the next step in any position of the Kripke model;
- formulas t_j (for j from 1 to m), called justice conditions, should be reachable from any state of the Kripke model.

The above formula $\neg(f \wedge (\mathbf{F} g \vee \mathbf{X}(h \mathbf{W} r)))$. After performing all the described operations, it turns into the following:

$(\neg f \vee p_1 \wedge p_3) \wedge \mathbf{G}((p_1 \rightarrow r_1) \wedge (r_1 \rightarrow \neg g) \wedge (p_2 \rightarrow \neg r \wedge (h \rightarrow r_2 \wedge r_3)) \wedge (r_1 \rightarrow \mathbf{X} r_1) \wedge (r_2 \rightarrow \mathbf{X}(\neg r \wedge (h \rightarrow r_2))) \wedge (r_3 \rightarrow \mathbf{X}(h \vee r \rightarrow r_3)) \wedge (p_3 \rightarrow \mathbf{X} p_2) \wedge \mathbf{F} \neg r_3)$.

Build a table. For the formulas $p, q, r_1, \dots, r_n, s_1, \dots, s_n$ and t_1, \dots, t_m the table is constructed as follows.

Let AP be the set of all atomic sentences of the obtained formula, for them there are a total of $2^{|AP|}$ value options (binary sets). One vertex is constructed for each possible binary set and this vertex is labeled with those atomic sentences that are true in this set.

Let, for $AP = \{a, b, c\}$, the atomic sentences a and c are active at the vertex corresponding to the set $\{a, b, c\} = \{1, 0, 1\}$, and the sentence b is inactive. Among these vertices are left only those whose corresponding binary set satisfies the formula q (makes its value true). The remaining vertices are deleted, since q must always be executed.

Further, for any binary sets x and y , an edge from the “vertex” x is added to the “vertex” y , if and only if for any i from 1 to n the implication is performed $r_i(x) \rightarrow s_i(y)$ condition. This edge corresponds to the condition $(r_1 \rightarrow \mathbf{X} s_1) \wedge \dots \wedge (r_n \rightarrow \mathbf{X} s_n)$ and means that for any i “if now r_i is satisfied, then in the next step s_i will be executed”.

Further, marked as starting states (there may be several) those in which the corresponding binary set satisfies the formula p . This corresponds to the condition that the formula p is true "at time 0".

The last stage is called “table cleaning” and is optional. It is designed to work more effectively with the model.

At this stage, the requirement is ensured that all paths in the graph must be infinite and that there must be paths to the vertices everywhere where the formulas t_j (j from 1 to m) are fulfilled. This is done by “recursive” removal from the model of all vertices that have no descendants, and all such vertices, from which for some j there is no path to one vertex in which the formula t_j is executed. That is, in the table you should leave only those states of which there is at least one infinite path that satisfies the conditions of justice (this means that all t_j are executed on this path infinitely often).

After these actions, you should leave in the table the vertices that are reachable from the initial ones, the rest should be deleted.

The resulting table reflects the meaning of the formula, and its endless paths satisfy two properties:

1. Let σ be some abstract path (a sequence of binary sets corresponding to atomic sentences). If the formula in question is true for the path σ , then the table contains an instance of this path, starting in the initial state.

2. Let σ be the path in the constructed table (starting in the initial state) such that each propositional formula t_j (for j from 1 to m) becomes true on the path σ infinitely often. Then the formula under consideration is true on the path σ .

In the first property, the presence of the path σ in the model is guaranteed. Indeed, in this way the formula is executed. Therefore, t_j for j from 1 to m become true infinitely often, and each vertex of this path satisfies two conditions:

- some endless way starts from it;
- for any j from it one can get to any vertex in which the formula t_j is true.

A vertex satisfying these two conditions is recursively removable only if one of the conditions is violated. This vertex is not deleted, and it is present in the model on the implementation of the path σ .

From the content of the formula, it follows that on the implementation of this path in the model, each propositional formula t_j (for j from 1 to m) becomes infinitely often true.

When building a table, the process of its “cleansing” can lead to the fact that all starting states will be excluded from it (and all the others, since only those states that are reachable from the initial states are left in the model) and the model becomes empty. That is, the formula is impracticable - there is no way in which it is true.

In the second property (which asserts almost the same, but in the opposite direction) the condition on t_j is essential.

Let the constructed model is not empty. Then, indeed, from each vertex of the model (by construction) you can reach all the vertices that correspond to the formulas t_j .

Therefore, from each vertex there is a path in which all formulas t_j are executed infinitely often. This path is constructed as follows.

First get from the vertex under consideration to the vertex for which t_1 is true, then from t_1 to t_2 , and so on, to t_m .

From t_m again to t_1 , and so on to infinity.

The foregoing can be done, since any t_j is reachable from any vertex. But, besides the way on which all t_j become true infinitely often, there may be other ways for which this condition is incorrect. Therefore, the condition on t_j is indicated explicitly. Each path in the model satisfies the formula: $p \wedge \mathbf{G}(q \wedge (r_1 \rightarrow \mathbf{X} s_1) \wedge \dots \wedge (r_n \rightarrow \mathbf{X} s_n))$, which differs from the original one in that it excludes requirements for t_j . The above two properties are performed for the paths and in the event that the “cleaning of the table” was not performed.

Thus, a method of constructing LTL expressions, LTL inference and their verification, including using an LTL verifier using the example of the specifications of two basic systems and their processes, is demonstrated. The obtained results confirmed the presence of the basic required properties of the subject LTL-specifications and indicate the feasibility of studying and practical development of the proposed models and the method of their analysis.

4 Conclusion

LTL Time Logic is a tool for mathematical modeling and research of complex systems. In one of the approaches to the design and analysis of systems, LTL analysis is used as an auxiliary tool. Here, to build the system, conventional design methods are used. Then the constructed system is modeled in LTL logic and the model is analyzed. If in the course of the analysis a flaw is found in the project, then the project is modified to eliminate them.

When the system is designed, the implementation phase begins, when other tools are relevant. In particular, these are systems that allow organizing work on a project.

5 Report content requirements

The report should include:

- purpose, individual task and program of laboratory work
- the source code of the program in the selected programming language
- report on the performance of each task.
- conclusions on the results of the tasks performed.

6 Themes of individual tasks

LTL-analysis of interactions in:

1. Smart system of operational individual lighting and air conditioning.
2. Smart system of operational monitoring, proposals for the selection and clearance of goods.
3. Smart system of operational monitoring and suggestions for

the selection of auto parts for repair.

4. System of interactive online theater box office.
5. Interactive online poster and help cinema.
6. Smart interactive online travel agency system.
7. Interactive Internet application filing system for the implementation of service services.

7 Recommended literature

1. Pallavi Sethi and Smruti R. Sarangi Internet of Things: Architectures, Protocols, and Applications. Journal of Electrical and Computer Engineering Volume 2017, Article ID 9324035, 25 p. <https://doi.org/10.1155/2017/9324035>

2. Dependable IoT for Human and Industry: Modeling, Architecting, Implementation, Vyacheslav Kharchenko, Ah Lian Kor, Andrzej Rucinski (Eds), River Publishers Series in Information Science and Technology, 2018, 450 p.

3. Tara Salman Networking Protocols and Standards for Internet of Things. – https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/index.html

4. Saber Talari, Miadreza Shafie-khah, Pierluigi Siano, Vincenzo Loia, Aurelio Tommasetti and João P. S. Catalão. A Review of Smart Cities Based on the Internet of Things Concept, Energies 2017, 10, 421. P. 23. <http://www.mdpi.com/1996-1073/10/4/421/pdf>

5. Y. Kondratenko, O. Kozlov, A. Topalov, O. Korobko, O. Gerasin. Automation of Control Processes in Specialized Pyrolysis Complexes Based on Industrial Internet of Things. In book: Dependable IoT for Human and Industry: Modeling, Architecting, Implementation, Vyacheslav Kharchenko, Ah Lian Kor, Andrzej Rucinski (Eds.), River Publishers Series in Information Science and Technology, 2018.

6. Daniel Shahaf Temporal Logics I: Theory. Tel-Aviv University November 2007. – P. 155. Available from: http://www.cs.tau.ac.il/~annaz/teaching/TAU_winter08/Seminar/daniel.pdf

7. Patricia Bouyer. Model-Checking Timed Temporal Logics. LSV – CNRS & ENS de Cachan, France. P. 142. Available from: <http://www.lsv.fr/~bouyer/files/tfit08.pdf>

8. Yoav Shoham, Kevin Leyton-Brown Multiagent Systems. Algorithmic, Game-Theoretic, and Logical Foundations. Revision 1.1 / Shoham and Leyton-Brown, 2010. P. 532. <http://www.masfoundations.org/mas.pdf>

9. A. Sugak, O. Martynyuk, A. Drozd. The Hybrid Agent Model of Behavioral Testing, International Journal of Computing, 2015, Volume 14, Issue 4, Ternopil, P. 232-244.

10.A. Sugak, O. Martynyuk, O. Drozd. Models of the Mutation and Immunity in Test Behavioral Evolution, Proceedings of the 2015 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2015, Warsaw, Poland, P. 790-795.

11. O. Martynyuk, A. Sugak, D. Martynyuk, O. Drozd. Evolutionary Network of Testing of the Distributed Information Systems, Proceedings of the 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2017, Bucharest, Romania, P. 888-893.

12. I. Skarga-Bandurova, M. Derkach, A. Velykzhanin, A Framework for Real-Time Public Transport Information Acquisition and Arrival Time Prediction Based on GPS Data. In book: Dependable IoT for Human and Industry: Modeling, Architecting, Implementation, Vyacheslav Kharchenko, Ah Lian Kor, Andrzej Rucinski (Eds.), River Publishers Series in Information Science and Technology, 2018.

13. Anduo Wang Formal Analysis of Network Protocols / University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-10-16. 2010. P. 32. https://repository.upenn.edu/cgi/viewcontent.cgi?article=1970&context=cis_reports

14.A. Boyarchuk, V. Kharchenko, O. Illiashenko, D. Maevsky, C. Phillips, A. Plakhtey, L. Vystorobska. Internet of Things for Human and Industry Applications: ALIOT Based Curriculum. In book: Dependable IoT for Human and Industry: Modeling, Architecting, Implementation, Vyacheslav Kharchenko, Ah Lian Kor, Andrzej Rucinski (Eds.), River Publishers Series in Information Science and Technology, 2018.

4 MARKOV'S AND SIMULATION MODELING OF IOT SYSTEMS

Laboratory work 1

Final probabilities assessment of IoT based systems with use of Markov models mathematical apparatus

1 Objectives and tasks

Objectives: to study and apply the mathematical method of Markov models for the final probabilities of model states assessment of IoT systems.

Learning tasks: to study the features of using Markov models mathematical apparatus for assessment the final probabilities;

Practical tasks:

– to learn how to work with Markov models mathematical method;

– to find values of final probabilities.

Exploring task: to assess the probabilities of states and availability function of IoT system for different variants.

Setting up

– to study the theoretical material contained in this manual, as well as a list of references;

– calculate the values of the probability of being in each state of the Markov model for a given initial data for each option.

Synopsis

In this lab students will become familiar with the foundation of the mathematical apparatus of Markov models for studying the reliability of systems and will use it to evaluate systems based on IoT.

2 Brief theoretical information

A random process is called a process with discrete states if the states of the system S_1, S_2, S_3, \dots can be listed one after another, and the process itself consists in the fact that from time to time the system S jumps (instantly) from one state to another.

A random process with continuous states: these processes are characterized by a gradual, smooth transition from state to state.

Definition.

A Markov chain is a sequence of random events in which the probability of each event depends only on the state in which the process is currently located and does not depend on earlier states. The final discrete chain is determined: by the set of states $S = \{S_1, \dots, S_n\}$, the event is a transition from one state to another as a result of random testing by the vector of initial probabilities (initial distribution) $p(0) = \{p^{(0)}(1), \dots, p^{(0)}(n)\}$, which determines the probabilities $p(0)(i)$ that at the initial moment of time $t = 0$ the process was in state S_i with the transition probability matrix $P = \{p_{ij}\}$, which characterizes the probability of the transition of the flowing state S_i to the following state S_j , and the sum of the probabilities of transitions from one state is equal to 1: $\sum p_{ij} = 1, i = 1 \dots n, j = 1 \dots n$.

If in the information sequence the probability of the appearance of a symbol is determined by only one previous symbol, it is said about such a sequence that it forms a Markov chain of the first order (simple or simply connected Markov chain). In the presence of a probabilistic dependence of the appearance of a symbol on K previous symbols, it is said that the information sequence forms a K -order Markov chain (K -connected Markov chain). The information source generating the K -order Markov chain is commonly called the K -order Markov source. This definition can be extended to the case of a source that does not have memory, calling such a source a zero-order Markov source.

The Markov chain is depicted as a transition graph, whose vertices correspond to the states of the chain, and the arcs correspond to transitions between them. The weight of the arc (i, j) connecting the vertices S_i and S_j will be equal to the probability $p_i(j)$ of the transition from the first state to the second.

Theoretical material for IoT based systems, safety and security issues are described in Part 34 (sections 39-42) of the book [3].

3 Execution order

Reliability can be determined using the mathematical apparatus of Markov models in this way. A random process is called a Markov process if the probability of a system transition to a new state depends only on the state of the system at the moment and does not depend on when and how the system went into that state.

A discrete Markov chain is called a random process in which the change of discrete states occurs at certain points in time. A continuous Markov process is a random process in which the change of discrete states occurs at random times. A Markov chain is said to be irreducible if any state S_j can be reached from any other state S_i over a finite number of transitions. In this case, all states of the chain are called communicating, and the transition graph is a component of strong connectivity. Markov chains are decomposable, indecomposable, periodic and ergodic.

Decomposable chain - contains irreversible (absorbing) states (sets states). No arcs emerge from such peaks. In steady state the probability of being in such a state is 1. The necessary condition is that state i is absorbing is: $p_{ii} = 1$.

Non-degradable circuit - does not contain absorbent states or absorbent subsets of nodes. Such chains are described by a strongly connected graph.

Ergodic Markov process - when from any state it is possible to move to any other state and over time the system comes to a limiting distribution, independent of the initial state.

A periodic chain is called such a chain, a sequence of state changes which change periodically. In the case of a periodic chain all states have the same period.

Example 1. For a given object, a state graph is constructed (for example, fig.1). The circles of the graph indicate the possible states of the system in question. Arrows connect them together indicating the rate of the transition from one state to another.

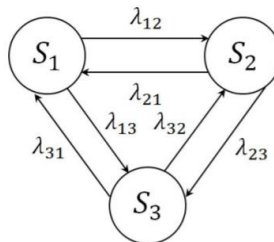


Fig. 1 – Graph of states of a Markov model

Then, according to the graph, a system of linear differential equations is compiled and initial conditions and normalization conditions are specified.

$$\begin{aligned}\frac{dP_1(t)}{dt} &= \lambda_{31} \cdot P_3(t) + \lambda_{21} \cdot P_2(t) - P_1(t) \cdot (\lambda_{12} + \lambda_{13}); \\ \frac{dP_2(t)}{dt} &= \lambda_{12} \cdot P_1(t) + \lambda_{32} \cdot P_3(t) - P_2(t) \cdot (\lambda_{21} + \lambda_{23}); \\ \frac{dP_3(t)}{dt} &= \lambda_{23} \cdot P_2(t) + \lambda_{13} \cdot P_1(t) - P_3(t) \cdot (\lambda_{31} + \lambda_{32}).\end{aligned}$$

$$\sum_{i=1}^3 P_i(t) = 1, P_1(0) = 1.$$

$$\begin{cases} x_1 = \lambda_{31} \cdot P_3(t) + \lambda_{21} \cdot P_2(t) - P_1(t) \cdot (\lambda_{12} + \lambda_{13}); \\ x_2 = \lambda_{12} \cdot P_1(t) + \lambda_{32} \cdot P_3(t) - P_2(t) \cdot (\lambda_{21} + \lambda_{23}); \\ x_3 = \lambda_{23} \cdot P_2(t) + \lambda_{13} \cdot P_1(t) - P_3(t) \cdot (\lambda_{31} + \lambda_{32}). \end{cases}$$

$x^{(i)} = (1,0,0)$ - state probability vector (shows the probability that the system will be in the i -th state).

$x^{(i)}$ - is a cross section of a random process.

Then

$$\begin{cases} 1 = -P_1(t) \cdot (\lambda_{12} + \lambda_{13}) + \lambda_{21} \cdot P_2(t) + \lambda_{31} \cdot P_3(t); \\ 0 = \lambda_{12} \cdot P_1(t) - P_2(t) \cdot (\lambda_{21} + \lambda_{23}) + \lambda_{32} \cdot P_3(t); \\ 0 = \lambda_{13} \cdot P_1(t) + \lambda_{23} \cdot P_2(t) - P_3(t) \cdot (\lambda_{31} + \lambda_{32}). \end{cases}$$

Transition probability matrix:

$$A = \begin{bmatrix} -(\lambda_{12} + \lambda_{13}) & \lambda_{21} & \lambda_{31} \\ \lambda_{12} & -(\lambda_{21} + \lambda_{23}) & \lambda_{32} \\ \lambda_{13} & \lambda_{23} & -(\lambda_{31} + \lambda_{32}) \end{bmatrix}$$

Using the normalization condition $P_1(0) = 1$, we get the matrix:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \lambda_{12} & -(\lambda_{21} + \lambda_{23}) & \lambda_{32} \\ \lambda_{13} & \lambda_{23} & -(\lambda_{31} + \lambda_{32}) \end{bmatrix}$$

To find the probabilities of each of the states of the system, the inverse matrix method is used. Consider the essence of this method.

Inverse matrix method solution algorithm.

Suppose that a system of linear algebraic equations of the following form is given:

1 Calculates the determinant of matrix A. If the determinant is zero, then the end of the solution. The system has an infinite number of solutions.

2 For a determinant other than zero, there is an inverse matrix A^{-1} through algebraic additions.

3 The solution vector $X = \{x_1, x_2, \dots, x_n\}$ is obtained by multiplying the inverse matrix by the result vector b.

The vector of unknowns is calculated by the formula $X = A^{-1} \cdot b$.

Solve the system by the inverse matrix method:

Decoupling

The coefficient matrix A and the vector of free coefficients b have the form: Inverse matrix method solution algorithm.

1 Calculate the matrix holder A. As a rule, the holder identifies zero, then ends. The system has a small number of solutions.

2 In case of a visitor to enter a zero, through the algebraic additional interchange, the A^{-1} matrix is inverted.

3 Vector solution $X = \{x_1, x_2, \dots, x_n\}$ to enter the multiples of the matrices on the vector of the result b.

Example 2. Solve the system by the inverse matrix method:

$$\begin{cases} 5 \cdot x_1 + 3 \cdot x_2 - 7 \cdot x_3 + 3 \cdot x_4 = 1 \\ x_2 - 3 \cdot x_3 + 4 \cdot x_4 = -5 \\ x_1 - 2 \cdot x_3 - 3 \cdot x_4 = -4 \\ 4 \cdot x_1 + 3 \cdot x_2 - 5 \cdot x_3 = 5 \end{cases}$$

Decision of system of linear equations can be done in MS Excel. Matrix A is in the range B1: E4, and vector b is in the range B6: B9. To solve the system of equations, it is necessary to calculate a matrix inverse to matrix A. To do this, select cells for storing the inverse matrix (G1: J4). Now let's turn to the Function Wizard, and in the Mathematical category, select the MOBR function intended to calculate the inverse of the matrix by clicking on the OK button, then go to the second step of the Function Wizard. In the dialog that appears in the second step of the function wizard, fill in the array input field.

Introduce matrix A and vector b into the worksheet of the table processor (Fig. 2).

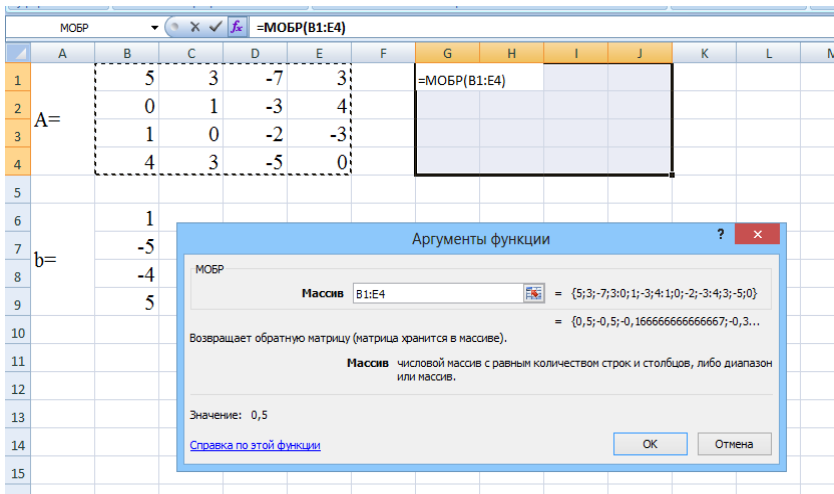


Fig. 2 - Function Wizard

This field must contain the range of cells in which the output matrix will be located (G1: J4). If the Array box is full, you can click OK. There will be a number in the first cell highlighted under the inverse of the range matrix. In order to receive the entire inverse matrix, you must press the F2 key to enter edit mode, and then simultaneously the Ctrl + Shift + Enter keys. It is now necessary to multiply the resulting inverse matrix by vector b. Select cells to store the resulting vector, for example G6: G9 (Fig. 3). Let's turn to the Function Wizard, and in the Mathematical category, select the MULTIPLE function that is intended to multiply matrices. Recall that the multiplication of matrices occurs as a rule row by column and matrix A can be multiplied by matrix B only if the number of columns of matrix A is equal to the number of rows of matrix B. In addition, the order of multipliers, $AB \neq BA$, is important.

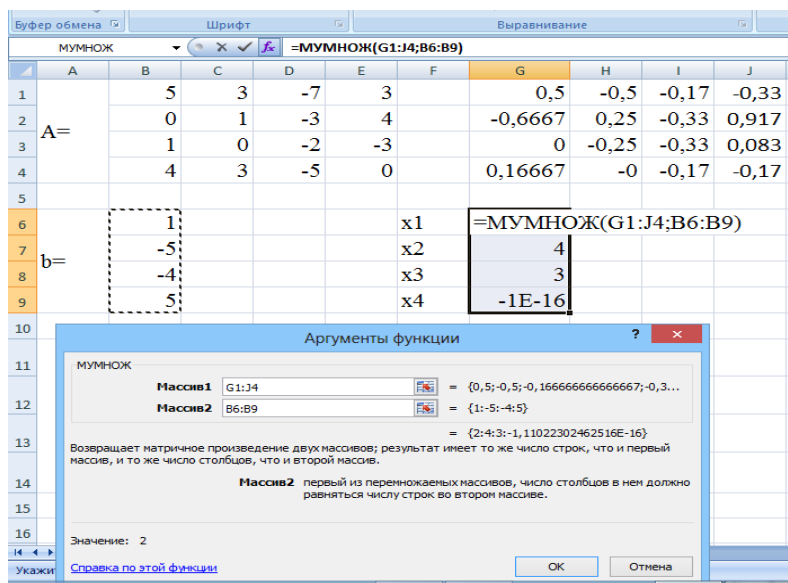


Fig. 3 – MULTIPLE function

If the input fields are full, you can click OK. The corresponding number of the resulting vector will appear in the first cell of the selected range. To get the entire vector, you must press the F2 key, and then simultaneously the Ctrl + Shift + Enter keys. The calculation results (vector x) are in cells G6: G9.

In order to check that the system of equations is solved correctly, it is necessary to multiply the matrix A by the vector x and to obtain the vector b. The multiplication of the matrix A by the vector x is performed by the MULTIPLE function.

Theoretical material for IoT based transport systems, safety and security issues are described in Part XI (sections 39-42) of the book [3].

	A	B	C	D	E	F	G	H	I	J	
1	A=	5	3	-7	3			0,50	-0,50	-0,17	-0,33
2		0	1	-3	4			-0,67	0,25	-0,33	0,92
3		1	0	-2	-3			0,00	-0,25	-0,33	0,08
4		4	3	-5	0			0,17	0,00	-0,17	-0,17
5											
6	b=	1				x1	2,00				
7		-5				x2	4,00				
8		-4				x3	3,00				
9		5				x4	0,00				
10											
11	Π ε ρ ε β ι ρ κ α	1									
12		-5									
13		-4									
14		5									

Fig. 4 – Results of calculation the final probabilities values

4 Themes of individual tasks

As a task, students choose a model in accordance with the list option in the journal and compose a system of linear Kolmogorov-Chapman differential equations for it using the inverse matrix method. Then the final probabilities are calculated and the result is presented to the teacher. Transition rates denote as λ_{ij} . The final probabilities denote as P_i .

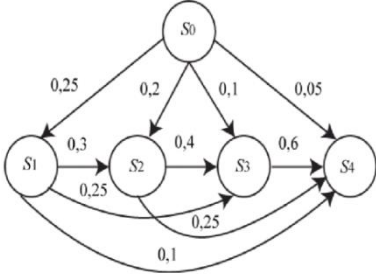
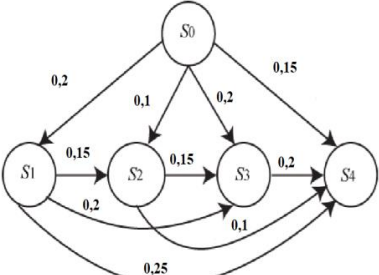
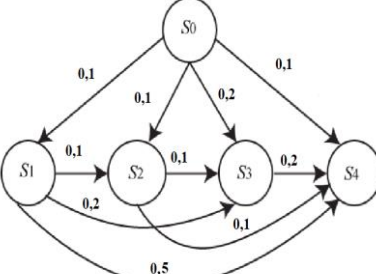
Table 1 demonstrates the variants for student’s research.

Further, students, in accordance with their option, perform modeling in one of the software products of MS Excel, Mathcad, Matlab, or by writing a program in one of the programming languages. For this, values of the transition rates variable are set, transition probabilities are calculated, and the results are entered in the table.

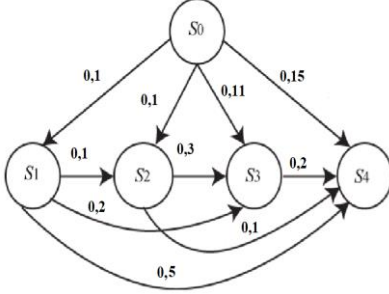
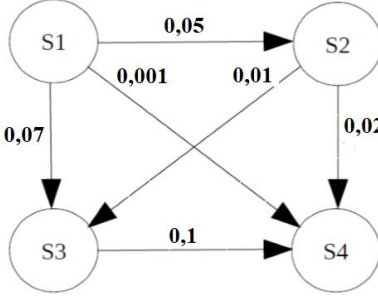
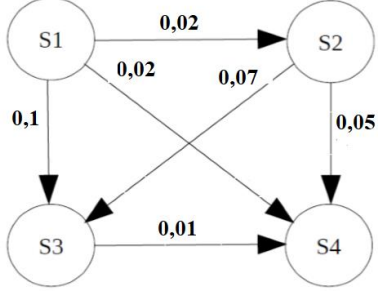
Next, the construction of the graphical dependence $P_i = f(\lambda_{ij})$. Results insert to tables 2 and 3, and build the graphical dependences $P0(\lambda_{ij})$ (if it exists in model by your variant), $P1(\lambda_{ij})$, $P2(\lambda_{ij})$, $P3(\lambda_{ij})$, $P4(\lambda_{ij})$. Students build and study the graphical dependencies of the probabilities of being in each state of the graph of the Markov model when changing the value of the transition rates from one state to

another in accordance with their version of the assignment and make practical recommendations on which probabilities and how they change when the transition rate parameter changes.

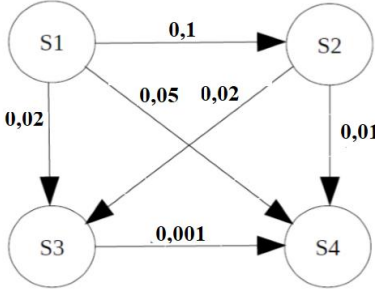
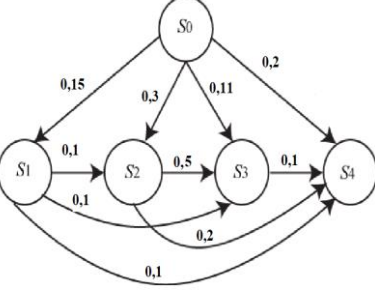
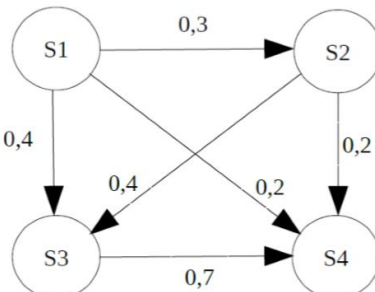
Table 1 - Variants of Markov models

№var	Graph of Markov model	Variable transition rates and probabilities
1		<p>Investigate changes in the values of the final probabilities when the transition rate $\lambda_{24} = 0,0.1,0.2,0.3,0.4,0.5$ changes first, then when the transition rate changes value</p> $\lambda_{01} = 0, 0.2,0.4,0.6,0.8,1$
2		<p>Investigate changes in the values of the final probabilities when the transition rate $\lambda_{12} = 0,0.1,0.2,0.3,0.4,0.5$ changes first, then when the transition rate changes value</p> $\lambda_{34} = 0, 0.2,0.4,0.6,0.8,1$
3		<p>Investigate changes in the values of the final probabilities when the transition rate $\lambda_{04} = 0,0.1,0.2,0.3,0.4,0.5$ changes first, then when the transition rate changes value</p> $\lambda_{13} = 0, 0.2,0.4,0.6,0.8,1$

Markov's and simulation modeling of IoT systems

4		<p>Investigate changes in the values of the final probabilities when the transition rate $\lambda_{23} = 0,0.1,0.2,0.3,0.4,0.5$ changes first, then when the transition rate changes value</p> <p>$\lambda_{13} = 0, 0.2,0.4,0.6,0.8,1$</p>
5		<p>Investigate changes in the values of the final probabilities when the transition rate $\lambda_{24} = 0,0.1,0.2,0.3,0.4,0.5$ changes first, then when the transition rate changes value</p> <p>$\lambda_{14} = 0, 0.2,0.4,0.6,0.8,1$</p>
6		<p>Investigate changes in the values of the final probabilities when the transition rate $\lambda_{13} = 0,0.1,0.2,0.3,0.4,0.5$ changes first, then when the transition rate changes value</p> <p>$\lambda_{12} = 0, 0.2,0.4,0.6,0.8,1$</p>

Markov's and simulation modeling of IoT systems

7		<p>Investigate changes in the values of the final probabilities when the transition rate $\lambda_{14} = 0,0.1,0.2,0.3,0.4,0.5$ changes first, then when the transition rate changes value</p> <p>$\lambda_{13} = 0, 0.2,0.4,0.6,0.8,1$</p>
8		<p>Investigate changes in the values of the final probabilities when the transition rate $\lambda_{03} = 0,0.1,0.2,0.3,0.4,0.5$ changes first, then when the transition rate changes value</p> <p>$\lambda_{14} = 0, 0.2,0.4,0.6,0.8,1$</p>
9		<p>Investigate changes in the values of the final probabilities when the transition rate $\lambda_{34} = 0,0.1,0.2,0.3,0.4,0.5$ changes first, then when the transition rate changes value</p> <p>$\lambda_{23} = 0, 0.2,0.4,0.6,0.8,1$</p>

10		<p>Investigate changes in the values of the final probabilities when the transition rate $\lambda_{24} = 0,0.1,0.2,0.3,0.4,0.5$ changes first, then when the transition rate changes value $\lambda_{34} = 0, 0.2,0.4,0.6,0.8,1$</p>
----	--	--

Table 2 – Results of calculation the $P_i(t)$

λ_{ij}	0	0.1	0.2	0.3	0.4	0.5
P0						
P1						
P2						
P3						
P4						

Table 3 - Results of calculation the $P_i(t)$

λ_{ij}	0	0.2	0.4	0.6	0.8	1.0
P0						
P1						
P2						
P3						
P4						

5 Requirements to the content of the report

The report should include:

- title page;
- research purpose and program;
- graph of a Markov model by variant;
- the system of differential linear equations by Kolmogorov-Chapman;
- results of the final probabilities assessment;

- tables with values of final probabilities if transition rates changes;
- graphical dependences $P_0(\lambda_{ij})$ (if it exist in model by your variant), $P_1(\lambda_{ij})$, $P_2(\lambda_{ij})$, $P_3(\lambda_{ij})$, $P_4(\lambda_{ij})$ and their analysis;
- conclusions;
- practical recommendations.

6 Testing questions

1. What does mean the term “ergodic Markov process”?
2. What is the difference between decomposable and indecomposable Markov chains?
3. What does mean the term “periodic Markov process”?
4. Please explain the essence of the inverse matrix method?
5. What does mean the term non-degradable circuit?
6. Please explain how the graph of the Markov model is constructed.
7. What formula can determine the vector of values of the final probabilities?
8. What is the difference between non-homogeneous and homogeneous Markov model?

7 Recommended literature

1. Margaret Rouse. Markov model. [<https://whatis.techtarget.com/definition/Markov-model>].
2. Charles M. Grinstead, J.Lauri Snell. Probability. The CHANCE Project1. Version dated 4 July 2006. Markov Chains. 518 p. [https://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/Chapter11.pdf].
3. Internet of Things for Industry and Human Application. In Volumes 1-3. Volume 3. Assessment and Implementation /V. S. Kharchenko (ed.) - Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019, 740 p.

Laboratory work 2

Availability function assessment of IoT based systems with use of Markov models mathematical apparatus

1 Objectives and tasks

Objectives: to study and apply the mathematical method of Markov models for the availability function assessment of IoT systems.

Learning tasks: to study the features of using Markov models mathematical apparatus for the availability function assessment;

Practical tasks: to learn how to find value of availability function with use of Markov models mathematical method.

Exploring task: to assess the probabilities of states and availability function of IoT system for different variants.

Setting up:

– to study the theoretical material contained in this manual, as well as a list of references;

- calculate the values of the availability function use the Markov model for a given initial data.

Synopsis

In the course of laboratory work, students master the mathematical apparatus of Markov models to assess the readiness function of the system based on the Internet of things.

2 Brief theoretical information

The use of the Markov model to assess the Internet of things system's availability is possible when the system failure flow is the simplest and obeys the exponential distribution law [1-2]. The simplest flow is when it is ordinary, stationary and inactivity.

A stationary process is a stochastic process in which the probability distribution does not change with time shift.

Stationarity property: the probability of occurrence of k events at any time interval depends only on the number k and on the duration t of the interval and does not depend on the origin of its counting.

The property of ordinaryness: the probability of occurrence of more than one event in an elementary period of time can be neglected in comparison with the probability of occurrence of no more than one

event in this period (that is, the probability of the simultaneous appearance of two or more events is zero).

The property of the absence of aftereffect: the probability of occurrence of k events at any time interval does not depend on whether events appeared or did not appear at time instants preceding the beginning of the considered period.

We accept the following assumptions for the model:

- the flow of hardware system failures obeys the Poisson distribution law;

- failures caused by software design faults of IoT obeys Poisson distribution, as on the results of monitoring and diagnostics, testing corrected secondary error (the result of the accumulation of the effects of primary faults, software backdoors);

- the process, which occurs in the system, it is a process without aftereffect, every time in the future behavior of the system depends only on the state of the system at this time and does not depend on how the system arrived at that state. Therefore, the process has the Markov property.

The availability function is defined as the sum of the probabilities of a system in a good working state.

Theoretical material for Markov models of IoT based systems are described in Part 18 of the book [3].

3 Execution order

Reliability can be determined using the mathematical apparatus of Markov models in this way. For the model shown in Fig. 1, compose a system of Kolmogorov-Chapman differential linear equations. For given initial conditions and initial data, get ahead of the availability function. For a given model, these are states 1 and 4.

The model for all options is selected the one, and the initial data are taken by students from the table according to their version from the list. Initial conditions: $\sum_{i=1}^6 P_i(t) = 1, P_1(0) = 1$.

Availability function $AC(t)$ can be calculated by equation: $AC(t) = P_1(t) + P_4(t)$.

To solve the task, it is necessary to find the final probabilities of finding the system in each of the states of the Markov model. To do this, use the inverse matrix method considered in laboratory work 1.

You can use any application for calculation - MS Excel, SMathStudio, Matlab, Mathcad, programming languages C++, Java and other.

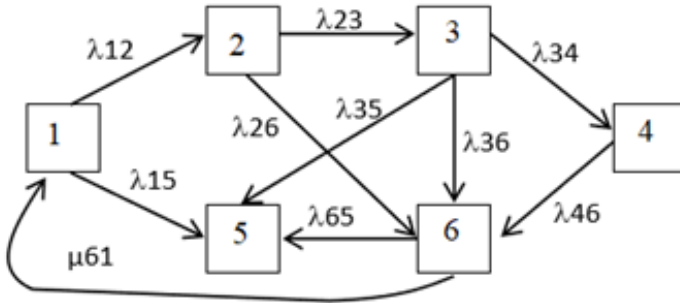


Fig. 1 – Graph of Markov model

Table 1 demonstrates the variants of transition rates of Markov model for the student's research.

Table 1 – Variants of initial data

N _o	Initial transition rates
1	$\lambda_{12} = 0,0015; \lambda_{23} = 0,001; \lambda_{15} = 0,002; \lambda_{26} = 0,013;$ $\lambda_{34} = 0,001; \lambda_{35} = 0,004; \lambda_{36} = 0,003; \lambda_{65} = 0,002;$ $\lambda_{46} = 0,025; \mu_{61} = 10$
2	$\lambda_{12} = 0,001; \lambda_{23} = 0,015; \lambda_{15} = 0,0001; \lambda_{26} = 0,003;$ $\lambda_{34} = 0,0015; \lambda_{35} = 0,002; \lambda_{36} = 0,02; \lambda_{65} = 0,001;$ $\lambda_{46} = 0,0015; \mu_{61} = 12$
3	$\lambda_{12} = 0,002; \lambda_{23} = 0,0012; \lambda_{15} = 0,001; \lambda_{26} = 0,0015;$ $\lambda_{34} = 0,0025; \lambda_{35} = 0,001; \lambda_{36} = 0,002; \lambda_{65} = 0,001;$ $\lambda_{46} = 0,003; \mu_{61} = 17$
4	$\lambda_{12} = 0,025; \lambda_{23} = 0,002; \lambda_{15} = 0,0025; \lambda_{26} = 0,001;$ $\lambda_{34} = 0,002; \lambda_{35} = 0,001; \lambda_{36} = 0,015; \lambda_{65} = 0,002;$ $\lambda_{46} = 0,00125; \mu_{61} = 15$
5	$\lambda_{12} = 0,0015; \lambda_{23} = 0,001; \lambda_{15} = 0,002; \lambda_{26} = 0,0013;$ $\lambda_{34} = 0,004; \lambda_{35} = 0,0004; \lambda_{36} = 0,003; \lambda_{65} = 0,002;$ $\lambda_{46} = 0,0025; \mu_{61} = 18$

Markov's and simulation modeling of IoT systems

6	$\lambda_{12} = 0,001; \lambda_{23} = 0,0014; \lambda_{15} = 0,0012; \lambda_{26} = 0,0015;$ $\lambda_{34} = 0,001; \lambda_{35} = 0,002; \lambda_{36} = 0,001; \lambda_{65} = 0,005;$ $\lambda_{46} = 0,0001; \mu_{61} = 20$
7	$\lambda_{12} = 0,0015; \lambda_{23} = 0,001; \lambda_{15} = 0,002; \lambda_{26} = 0,0013;$ $\lambda_{34} = 0,003; \lambda_{35} = 0,004; \lambda_{36} = 0,003; \lambda_{65} = 0,002;$ $\lambda_{46} = 0,0025; \mu_{61} = 22$
8	$\lambda_{12} = 0,0025; \lambda_{23} = 0,001; \lambda_{15} = 0,001; \lambda_{26} = 0,003;$ $\lambda_{34} = 0,001; \lambda_{35} = 0,001; \lambda_{36} = 0,002; \lambda_{65} = 0,0005;$ $\lambda_{46} = 0,00015; \mu_{61} = 16$
9	$\lambda_{12} = 0,0005; \lambda_{23} = 0,001; \lambda_{15} = 0,002; \lambda_{26} = 0,0015;$ $\lambda_{34} = 0,003; \lambda_{35} = 0,0002; \lambda_{36} = 0,0015; \lambda_{65} = 0,0001;$ $\lambda_{46} = 0,0015; \mu_{61} = 20$
10	$\lambda_{12} = 0,002; \lambda_{23} = 0,00015; \lambda_{15} = 0,00025; \lambda_{26} = 0,002;$ $\lambda_{34} = 0,00025; \lambda_{35} = 0,003; \lambda_{36} = 0,002; \lambda_{65} = 0,0015;$ $\lambda_{46} = 0,0015; \mu_{61} = 15$

The task of students is to determine the values of the availability function for given input data and to plot graphically the availability function if change the transition rate value parameter specified in the variant of the task $AC = f(\lambda_{ij})$ (Table 2).

Students should receive 4 graphic dependences of the availability function on the transition rate and analyze how a change in the transition rate value affects the value of the availability function.

Table 2 – Variants of transition rates changing

№	λ_{ij}
1	$\lambda_{12} = 0...1; \lambda_{23} = 0...0.8; \lambda_{15} = 0...1; \mu_{61} = 0...20$
2	$\lambda_{36} = 0...2; \lambda_{65} = 0...1; \lambda_{46} = 0...15; \mu_{61} = 0...12$
3	$\lambda_{23} = 0...1; \lambda_{15} = 0...1; \lambda_{26} = 0...1; \mu_{61} = 0...15$
4	$\lambda_{34} = 0...1; \lambda_{35} = 0...1; \lambda_{36} = 0...0.1; \mu_{61} = 0...20$
5	$\lambda_{36} = 0...1; \lambda_{65} = 0...2; \lambda_{46} = 0...1; \mu_{61} = 0...10$
6	$\lambda_{12} = 0...1; \lambda_{23} = 0...1; \lambda_{15} = 0...1; \mu_{61} = 0...10$
7	$\lambda_{23} = 0...1; \lambda_{15} = 0...1; \lambda_{26} = 0...1; \mu_{61} = 0...15$
8	$\lambda_{35} = 0...1; \lambda_{36} = 0...1; \lambda_{65} = 0...1; \mu_{61} = 0...25$
9	$\lambda_{15} = 0...1; \lambda_{26} = 0...1; \lambda_{34} = 0...1; \mu_{61} = 0...15$
10	$\lambda_{15} = 0,002; \lambda_{34} = 0,001; \lambda_{36} = 0,003; \mu_{61} = 0...10$

4 Requirements to the content of the report

The report should include:

- title page;
- research purpose and program;
- graph of a Markov model by variant;
- the system of differential linear equations by Kolmogorov-Chapman;
 - values of final probabilities;
 - results of the availability function assessment, graphical dependences and their analysis;
 - conclusions;
 - practical recommendations.

5 Testing questions

1. What does mean the term “simplest stream of faults”?
2. What is the difference between stationarity and ordinariness properties?
3. What does mean the term “the property of the absence of aftereffect”?
4. Please explain how can be calculate the availability function with use of mathematical apparatus of Markov models?
5. What the assumptions can be used for applying the mathematical apparatus of Markov models?

6 Recommended literature

1. Margaret Rouse. Markov model. [<https://whatis.techtarget.com/definition/Markov-model>].
2. Charles M. Grinstead, J.Lauri Snell. Probability. The CHANCE Project1. Version dated 4 July 2006. Markov Chains. 518 p. [https://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/Chapter11.pdf].
3. Internet of Things for Industry and Human Application. In Volumes 1-3. Volume 3. Assessment and Implementation /V. S. Kharchenko (ed.) - Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019, 740 p.

COURSE PROGRAM

TITLE OF THE COURSE	Code
Simulation of IoT-based Systems	PCM 1

Teacher(s)	Department
Coordinating: Prof. Dmitry Maevsky Others: Module PCM1.1: DrS, Prof. Tabunshchyyk G. Module PCM1.2: Ass. Prof., PhD Maevskaya O. Module PCM1.3: DrS, Prof. Drozd O., Ass. Prof., PhD Martinuk O. M. Module PCM1.4: DrS, Prof. Kharchenko V., Ass. Prof., PhD Kolisnyk M.	ONPU, Institute of Electro mechanics and Energetic Management; ZNTU, Software Tools Department Computer Engineering KhAI, Computer systems, networks and cyber security

Study cycle	Level of the module	Type of the module
Master	A	Full-time tuition

Form of delivery	Duration	Language(s)
Full-time tuition	One semester	English

Prerequisites	
Prerequisites: Foundation of Modeling; Computer Systems and System Analysis, Machine Learning, Modeling Foundation knowledge and skills in CAD; Computer Networks; Information-Networking Technologies, Theory of computation; Programming Languages; Digital signal processor.	Co-requisites (if necessary):

Credits of the module	Total student workload	Contact hours	Individual work hours
4	138	64	74

Aim of the module (course unit): competences foreseen by the study programme
The aim of the course is: – To give students of overview of IoT systems architecture and

classification of interaction and tools of their simulation and implementation based on case studies approach.

- To create a knowledge base on practical using of program tools for simulation in IoT. The study also expands the notion about interactions between microprocessor and sensors as well as between microprocessor and actuators

- Acquisition of knowledge in simulation of IoT and IoE-based systems and their components regarding architecture, behavior and process synchronization. Obtaining skills in simulation of IoT and IoE-based systems in the modern instrumental environments with use of UML charts, Petri nets and temporal logic.

- Acquisition of general approach to structures and models building of IoT-based industrial systems

Learning outcomes of module (course unit)	Teaching/learning methods	Assessment methods
At the end of course, the successful student will be able to: 1 Formulate the main ideas of creation, simulation and verification of IoT and IoE-based systems.	Interactive lectures, Learning in laboratories, Just-in-Time Teaching	Module Evaluation Questionnaire
2 To distinguish tasks for simulation m2m, d2d different architecture levels of IoT systems	Interactive lectures, Learning in laboratories, Just-in-Time Teaching	Module Evaluation Questionnaire
3 To use Interaction Flow Modeling Language for modelling interactions for different IoT solutions	Interactive lectures, Learning in laboratories, Just-in-Time Teaching	Module Evaluation Questionnaire
4 Features of designing and debugging programs for microprocessors ARDUINO and Raspberry Pi	Interactive lectures, Learning in laboratories, Just-in-Time Teaching	Module Evaluation Questionnaire
5 To execute simulation and verification in architecture of IoT and IoE-based systems with use of UML charts.	Interactive lectures, Learning in laboratories, Just-in-Time Teaching	Module Evaluation Questionnaire
6 To build, simulate and execute verification of the formal behavior models of an automata and Petri nets for IoT and IoE-based systems.	Interactive lectures, Learning in laboratories, Just-in-Time Teaching	Module Evaluation Questionnaire

<p>7. To execute the description of synchronization processes in IoT and IoE-based systems with use of LTL models. 8. To apply the mathematical method of Markov models for the final probabilities of model states assessment of IoT systems for the availability function assessment of IoT systems</p>	<p>Interactive lectures, Learning in laboratories, Just-in-Time Teaching</p>							<p>Module Evaluation Questionnaire</p>	
<p>Themes</p>	<p>Contact work hours</p>							<p>Time and tasks for individual work</p>	
	<p>Lectures</p>	<p>Consultations</p>	<p>Seminars</p>	<p>Practical work</p>	<p>Laboratory work</p>	<p>Placements</p>	<p>Total contact work</p>	<p>Individual work</p>	<p>Tasks</p>
<p>1 Introduction into infrastructure in the IoT systems</p>	<p>2</p>						<p>2</p>		<p>Group work</p>
<p>2 Classification of the interactions in the IoT systems</p>	<p>2</p>						<p>2</p>		<p>Group work on Patterns Analysis</p>
<p>3 Interaction Flow Modeling Language Usage of FSM models for interaction modelling</p>	<p>2</p>		<p>2</p>	<p>2</p>			<p>6</p>	<p>1 1</p>	<p>1 Individual tasks in developing IFML artifacts for IoT systems 2 Implementation of developed models in Remote Laboratory GOLDi</p>
<p>4 Verification of IoT systems. Case Studies</p>	<p>2</p>		<p>2</p>		<p>4</p>		<p>8</p>	<p>1 0</p>	<p>1 Implementation developed models in Remote Laboratories GOLDi and ISRT</p>

									2 Simulation with the Bluemix platform simulations for such systems: Smart Campus, e-Health, Intelligent Transport
5 Types of program tools for IoT simulations. 5.1 Classification and terminology. Real-world objects and kinds of it simulation. 5.2 Physical and computer simulations. 5.3 Virtual simulation. Common user interaction systems for virtual simulations.	2						2	2	1 Difference between physical and computer simulation. 2 Kinds of computer simulation
6 Simulation IoT devices based on Arduino platform. 6.1 General information about the Arduino platform. 6.2 Arduino and Arduino-compatible boards. 6.3 Arduino IDE and IO system. 6.4 Software development. Arduino C/C++ sketch. 6.5 Program tools for simulations of Arduino-based systems.	2			4			6	4	1 Simulation of the Arduino devices in Proteus 8 Professional. 2 VIRTRONIX – Simulator for Arduino. 3 Fritzing Arduino Simulator

<p>7 Simulation IoT devices based on Raspberry Pi platform. 7.1 Single-board computers Raspberry Pi – architecture. 7.2 Command system and programming features. 7.3 Autodesk Electronic Lab – simulation workbench.</p>	2			2		4	4	<p>1 RISK processors. 2 General purpose input-output (GPIO). 3 Windows 10 IoT Core operation system. 4 Simulation of IoT systems in Autodesk Electronic Lab</p>
<p>8 Application of event-oriented programming and UML diagrams for simulation and design of IoT devices 8.1 Event-oriented programming 8.2 What is UML and UML diagrams 8.3 Using of UML diagrams for simulation and design of IoT devices</p>	2			2		4	4	<p>1 UML-modelling for IoT devices 2 An Internet of Things visual domain specific modeling language</p>
<p>9 Simulation and verification in architecture of IoT and IoE-based systems with use of UML charts 9.1 Introduction to representation of architecture of IoT and IoE-based systems with use of visual UML charts (precedents, components, classes, activities, the sequences of actions). 9.2 Static visual UML charts for the</p>	2			2		4	8	<p>1 Simulation and debugging of static visual UML charts in the description of architecture of IoT and IoE-based systems 2 Features of simulation and debugging of dynamic visual UML charts in the description of architecture of IoT and IoE-based systems.</p>

<p>description of architecture of IoT and IoE-based systems.</p> <p>9.3 Features of dynamic visual UML charts for the description of architecture of IoT and IoE-based systems.</p> <p>9.4 The analysis of static visual UML charts in the description of architecture of IoT and IoE-based systems.</p> <p>9.5 Features of the analysis of dynamic visual UML charts in the description of processes into architecture of IoT and IoE-based systems.</p>									
<p>10 Simulation and verification in behavior of IoT and IoE-based systems on basis of the expanded automata and hierarchical Petri nets</p> <p>10.1 Introduction to the description of IoT and IoE-based systems behavior with use of expanded automata, networks of queuing service and the hierarchical colored Petri nets.</p> <p>10.2 The review of the instrumental environments for simulation in behavior of IoT and IoE-based systems and their components</p>	2		2		2		6	1 2	<p>1 The analysis of a correctness of the formal behavior models of an automatic class on the basis of graphs of achievable marking, invariants of line items and transitions.</p> <p>2 Verification and testing in behavior of IoT and IoE-based systems on the basis of the structural and behavioral tests synthesized for Petri nets.</p>

<p>10.3 Development and simulation of imitation models of queuing services for IoT and IoE-based systems and their components in the instrumental environments OMNet ++, GPSS World, ExtendSim.</p> <p>10.4 Development of the formal models of an automata and Petri nets for the description in behavior of IoT and IoE-based systems and their components.</p> <p>10.5 Simulation and verification of the formal behavior models of an automata and Petri nets in the instrumental environment CPNTools.</p>									
<p>11 Simulation and verification of synchronization processes in IoT and IoE-based systems on the basis of temporal logic</p> <p>11.1 Introduction to temporal logic for the description of process synchronization in IoT and IoE-based systems.</p> <p>11.2 The description of synchronization processes with use of</p>	2		2	2			6	1 2	1 Coordination in levels of simulation for IoT and IoE-based systems with use of UML charts, Petri nets and temporal logic

LTL models in the Promela language. 11.3 Features of process synchronization for IoT and IoE-based systems in the instrumental environments SPIN, XSPIN. 11.4 Temporal verification of processes in IoT and IoE-based systems and their components on the basis of temporal logic. 11.5 Composition temporal output of LTL logic on the basis of the proof of precedence for conditions and events of processes.								
12 Final probabilities assessment of IoT based systems with use of Markov models mathematical apparatus 13 Availability function assessment of IoT based systems with use of Markov models mathematical apparatus 14 Semi Markov's modeling of IoT systems	6			8	14	16	1	2 Features of Markov's modeling of IoT systems considering cyber security and availability
On the whole	28		44	28	64	74		

Assessm	Weight in %	Dead	Assessment criteria
---------	-------------	------	---------------------

ent strategy		lines	
Lecture activity, including fulfilling special self-tasks	10	7,14	<p>85% – 100% Outstanding work, showing a full grasp of all the questions answered.</p> <p>70% – 84% Perfect or near perfect answers to a high proportion of the questions answered. There should be a thorough understanding and appreciation of the material.</p> <p>60% – 69% A very good knowledge of much of the important material, possibly excellent in places, but with a limited account of some significant topics.</p> <p>50% – 59% There should be a good grasp of several important topics, but with only a limited understanding or ability in places. There may be significant omissions.</p> <p>45% – 49% Students will show some relevant knowledge of some of the issues involved, but with a good grasp of only a minority of the material. Some topics may be answered well, but others will be either omitted or incorrect.</p> <p>40% – 44% There should be some work of some merit. There may be a few topics answered partly or there may be scattered or perfunctory knowledge across a larger range.</p> <p>20% – 39% There should be substantial deficiencies, or no answers, across large parts of the topics set, but with a little relevant and correct material in places.</p> <p>0% – 19% Very little or nothing that is correct and relevant.</p>
Learning in	30	7,14	85% – 100% An outstanding piece of work, superbly organised and

<p>laborator ies</p>		<p>presented, excellent achievement of the objectives, evidence of original thought.</p> <p>70% – 84% Students will show a thorough understanding and appreciation of the material, producing work without significant error or omission. Objectives achieved well. Excellent organisation and presentation.</p> <p>60% – 69% Students will show a clear understanding of the issues involved and the work should be well written and well organised. Good work towards the objectives.</p> <p>The exercise should show evidence that the student has thought about the topic and has not simply reproduced standard solutions or arguments.</p> <p>50% – 59% The work should show evidence that the student has a reasonable understanding of the basic material. There may be some signs of weakness, but overall the grasp of the topic should be sound. The presentation and organisation should be reasonably clear, and the objectives should at least be partially achieved.</p> <p>45% – 49% Students will show some appreciation of the issues involved. The exercise will indicate a basic understanding of the topic, but will not have gone beyond this, and there may well be signs of confusion about more complex material. There should be fair work towards the laboratory work objectives.</p> <p>40% – 44% There should be some work towards the laboratory work objectives, but significant issues are</p>
--------------------------	--	--

Abstract and Contents

			likely to be neglected, and there will be little or no appreciation of the complexity of the problem. 20% – 39% The work may contain some correct and relevant material, but most issues are neglected or are covered incorrectly. There should be some signs of appreciation of the laboratory work requirements. 0% – 19% Very little or nothing that is correct and relevant and no real appreciation of the laboratory work requirements.
Module Evaluation Question	60	8,16	The score corresponds to the percentage of correct answers to the test questions

Author	Year of issue	Title	No of periodical or volume	Place of printing. Printing house or internet link
Compulsory literature				
	2015	An Internet of Things: an Overview		https://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151014_0.pdf
Г. В. Табунщик, Т.І. Каплієнко, О.А Петрова	2016	Проектування та моделювання програмного забезпечення сучасних інформаційних систем		Запоріжжя: Дике поле.
		The Interaction Flow Modeling Language™ (IFML™) Resource Page		http://www.omg.org/ifml/
		IBM Bluemix Catalog		https://console.ng.bluemix.net/catalog/

Abstract and Contents

Alexander K Hartmann	2015	Big Practical Guide to Computer Simulations 2nd Edition		World Scientific
Averill Law	2015	Simulation Modeling and Analysis		McGraw-Hill
Sivarama P. Dandamudi	2005	Guide to RISC Processors: for Programmers and Engineers		Springer
S. S. Patil, S. Katwe, U. Mudengudi, R. B. Shettar and P. Kumar	2016	Open Ended Approach to Empirical Learning of IOT with Raspberry Pi in Modeling and Simulation Lab		2016 IEEE Eighth International Conference on Technology for Education (T4E), Mumbai, 2016, pp. 258-259. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7814843&isnumber=7814775
D'Angelo, S. Ferretti and V. Ghini	2016	Simulation of the Internet of Things		2016 International Conference on High Performance Computing & Simulation (HPCS), Innsbruck, 2016, pp. 1-8. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7568309&isnumber=7568299
G. Fortino, W. Russo and C. Savaglio	2016	Agent-oriented modeling and simulation of IoT networks	Vol. 1, Nos. 1/2	2016 Federated Conference on Computer Science and Information Systems (FedCSIS),

Abstract and Contents

				Gdansk, 2016, pp. 1449-1452 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7733442&isnumber=7733200
	2017	The Unified Modeling Language		http://www.uml-diagrams.org/
	2017	Edit UML models and diagrams		https://msdn.microsoft.com/en-us/library/dd409405.aspx
	2017	UML Diagram Software - Perfect UML Diagram Examples, Templates, Knowledge, Software, Free Download		https://www.edrawsoft.com/UML-Diagrams.php
Ileana Ober – UPS, Toulouse, France http://www.irit.fr/~Ileana.Ober	2005	UML / UML 2.0 tutorial	106 P.	http://www.artist-embedded.org/docs/Events/2005/SummerSchool_Naeslingen/IleanaOber.pdf
Eran Kampf	2005	Unified Modeling Language	34 P.	http://storage.developerzen.com/UML.pdf
Ivo Adan, Jacques Resing	2015	Queueing Systems	182 P.	http://www.win.tue.nl/~iadan/queueing.pdf
Dr. Janos Sztrik	2012	Basic Queueing Theory	193 P.	http://irh.inf.unideb.hu/~jsztrik/education/16/SOR_Main_Angol.pdf
	2017	TicToc Tutorial for OMNeT++		https://omnetpp.org/doc/omnetpp/tictoc-tutorial/
	2017	ExtendSim		https://www.extend

Abstract and Contents

		Books		dsim.com/sols_books.html
G. Geeraerts		An Introduction to Petri Nets and how to analyse him...	341 P.	http://www.ulb.ac.be/di/ssd/ggeeraert/Tutorial-Petri-Nets-Geeraerts.pdf
Alessandro Artale		Formal Methods Lecture III: Linear Temporal Logic	44 P.	http://web.iitd.ac.in/~sumeet/slide2.pdf
	2017	CPNTools Documentation		http://cpntools.org/documentation/start/
Additional literature				
Marco Lützenberger and Sahin Albayrak	2000	Simulation-based development of IoT applications with multi-agent technology current problems and future research directions.		In Proceedings of the 2016 Winter Simulation Conference (WSC '16). IEEE Press, Piscataway, NJ, USA, 3690-3691
Y. Y. Haimes	2008	Models for risk management of systems	Vol. 1, Nos. 1/2	Int. J. System of Systems Engineering
Bernd Bruegge	2006	Modeling with UML: Basic Notations	36 P.	https://www.bruegge.in.tum.de/lehstuhl_1/files/teaching/ws0607/Software%20Engineering%20I/L2Dist_ModelingWithUML_Part1.pdf
	2005	StarUML 5.0 Developer Guide	123 P.	http://staruml.sourceforge.net/docs/StarUML_5.0_Developer_Guide.pdf
Theo C. Ruys	2002	SPIN Beginners Tutorial	50 P.	http://spinroot.com/spin/Doc/SpinTutorial.pdf

АНОТАЦІЯ

УДК 004.415/.416:004.94](076.5)=111

Дрозд О.В., Маєвський Д.А., Маєвська О.Ю., Мартинюк О.М., Табунщик Г.В., Колісник М.О., Степова Г.С., Харченко В.С., Чопик Ю.О., Нагачевський Н.О., Савельєв А. А., Горошко В. В. **Моделювання систем Інтернету речей. Практикум** / За ред. Д.А. Маєвського – Міністерство освіти та науки України, Одеський національний політехнічний університет, Запорізький національний технічний університет, Національний аерокосмічний університет «ХАІ»: 2019. – 130 с.

Практичні матеріали навчального PhD курсу “Моделювання систем Інтернету речей”, надані в цій книзі, розроблено в рамках проекту ERASMUS+ ALIOT 73818-EPP-1-2016-1-UK-EPPKA2-SBHE-JP “Internet of Thing: Emerging Curriculum for Industry and Human Applications”.

Книга присвячена аналізу методів для міждисциплінарних досліджень з моделювання систем Інтернету речей з використанням математичних і натурних методик. Надано навчальний план, опис лабораторних робіт та рекомендації щодо самостійного вивчення курсу.

Книга підготовлена для аспірантів університетів, які проводять дослідження з комп'ютерної безпеки, комп'ютерної та програмної інженерії, систем Інтернету речей. Вона може бути корисною для лекторів та викладачів, які проводять заняття на відповідних курсах.

Бібл. – 63, рисунків – 40, таблиць – 9.

ЗМІСТ

СКОРОЧЕННЯ	3
ВСТУП	5
1 МОДЕЛЮВАННЯ ВЗАЄМОДІЇ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ	7
Лабораторна робота 1 Моделювання взаємодії для WEB-орієнтованих систем	7
1 Теоретична інформація	7
2 Послідовність завдань	9
3 Зміст звіту	9
4 Питання для самооцінки	9
Лабораторна робота 2 Моделювання взаємодії з мобільними додатками	10
1 Теоретична інформація	10
2 Послідовність завдань	12
3 Зміст звіту	13
4 Питання для самооцінки	13
5 Рекомендована література	14
2 ПРОГРАМНІ ІНСТРУМЕНТИ ДЛЯ МОДЕЛЮВАННЯ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ	15
Лабораторна робота 1 Моделювання Arduino пристроїв у Proteus 8 Professional. Початок роботи	15
1 Стисла теоретична інформація	15
1.1 Визначення ARDUINO набору	16
1.2 Технічні характеристики Arduino Mega 2560	17
1.3 Джерело живлення Arduino Mega 2560	17
1.4 Входи та виходи Arduino Mega 2560.....	18
1.5 Arduino IDE	19
2 Програма розробок та досліджень	22
3 Рекомендації до роботи	22
4 Індивідуальні завдання	26
5 Зміст звіту до роботи	27
6 Контрольні запитання	27
7 Рекомендована література	28
Лабораторна робота 2 Моделювання Arduino пристроїв у Proteus 8 Professional. Використання світлових датчиків в інтелектуальних електричних системах	29

1 Стисла теоретична інформація	29
1.1 Принцип роботи датчика	29
1.2 Конструкція датчика світла модуля LM393.....	30
1.3 Підключення датчика до Arduino плати.....	31
1.4 Технічні характеристики Light Sensor LM393	31
2 Програма розробок та досліджень	32
3 Рекомендації до роботи	32
3.1 План завантаження	32
3.2 Підключення датчика.....	33
3.3 Тестування датчика	34
4 Індивідуальні завдання	35
5 Зміст звіту до роботи	37
6 Контрольні запитання	37
7 Рекомендована література	38
Лабораторна робота 3 Моделювання Arduino пристроїв у Proteus 8 Professional. Використання датчиків руху PIR в інтелектуальних електричних системах	39
1 Стисла теоретична інформація	39
1.1 Конструкція датчика PIR	40
1.2 Принцип роботи датчика	41
1.3 Підключення датчика до Arduino плати.....	41
1.4 Методи встановлення параметрів датчика.....	42
1.5 Режими роботи.....	43
1.6 Технічні характеристики.....	43
1.7 Особливості використання	44
2 Програма розробок та досліджень	44
3 Рекомендації до роботи	44
3.1 План завантаження	44
3.2 Підключення датчика.....	46
3.3 Тестування датчика	46
4 Індивідуальні завдання	47
5 Зміст звіту до роботи	49
6 Контрольні запитання	49
7 Рекомендована література	50
3 ТРИРІВНЕВЕ МОДЕЛЮВАННЯ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ З ВИКОРИСТАННЯМ UML МОДЕЛЕЙ, ПЕТРІ МЕРЕЖ ТА ТЕМПОРАЛЬНОЇ ЛОГІКИ	51
Лабораторна робота 1 Моделінг Інтернету речей,	51

використовуючи UML діаграми на етапах їх архітектурного проектування	
1 Мета та задачі	51
1.1 Навчальні задачі	51
1.2 Практичні задачі	52
1.3 Дослідницькі задачі	52
2 Підготовка лабораторної роботи	52
3 Виконання лабораторної роботи	52
3.1 Порядок проведення лабораторної роботи	52
3.2 Приклади задач	54
4 Висновок	60
5 Теми індивідуальних завдань	61
6 Рекомендована література	61
Лабораторна робота 2 Дослідження Інтернету речей у поведінці використанням Color Petri Net Tools	63
1 Мета та задачі	63
1.1 Навчальні задачі	63
1.2 Практичні задачі	64
1.3 Дослідницькі задачі	64
2 Підготовка лабораторної роботи	64
3 Виконання лабораторної роботи	65
3.1 Порядок проведення лабораторної роботи	65
3.2 Приклади виконання завдань	67
4 Висновок	73
5 Вимоги до змісту звіту	73
6 Теми індивідуальних завдань	73
7 Рекомендована література	74
Лабораторна робота 3 Моделінг Інтернету речей, використовуючи темпоральну логіку на етапах їх архітектурного проектування	76
1 Мета та задачі	76
1.1 Навчальні задачі	76
1.2 Практичні задачі	76
1.3 Дослідницькі задачі	77
2 Підготовка лабораторної роботи	77
3 Виконання лабораторної роботи	78
3.1 Порядок проведення лабораторної роботи	78
3.2 Приклади виконання завдань	79

4 Висновок	85
5 Вимоги до змісту звіту	85
6 Теми індивідуальних завдань	85
7 Рекомендована література	86
4 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ ТА МОДЕЛЮВАННЯ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ ЗА ДОПОМОГОЮ МАРКОВСЬКИХ МОДЕЛЕЙ	88
Лабораторна робота 1 Остаточна оцінка ймовірностей систем речей з використанням математичного апарату моделей Маркова	88
1 Мета та задачі	88
2 Стисла теоретична інформація	88
3 Порядок виконання	89
4 Теми індивідуальних завдань	95
5 Вимоги до змісту звіту	99
6 Контрольні запитання	100
7 Рекомендована література	100
Лабораторна робота 2 Оцінка функції готовності систем речей з використанням математичного апарату моделей Маркова	101
1 Мета та задачі	101
2 Стисла теоретична інформація	101
3 Порядок виконання	102
5 Вимоги до змісту звіту	105
6 Контрольні запитання	105
7 Рекомендована література	105
ПРОГРАМА КУРСУ	106
АНОТАЦІЯ	120
ЗМІСТ	121

ABSTRACT

UDC 004.415/.416:004.94](076.5)=111

Drozd O.V., Maevsky D.A., Maevskaya O.J., Martynyuk O.M., Tabunshchyk G.V., Kolisnyk M.O., Stepova H.S., Kharchenko V.S., Chopyk Y.O., Nagachevsky N.O., Savelev A.A., Goroshko V.V. **Simulation of Internet of Things based Systems.** Practicum / Edited by Maevsky D.A. – Ministry of Education and Science of Ukraine, Odessa National Polytechnic University, National Aerospace University “KhAI”, Zaporizhzhia National Technical University: 2019. – 130 p.

Practical materials of study PhD course “Simulation of IoT-based Systems” given in this book are developed within project ERASMUS+ ALIOT 73818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP Internet of Thing: Emerging Curriculum for Industry and Human Applications

The structure of work on verification of residual knowledge in the discipline, the corresponding practical material, examples of tasks and criteria of evaluation are given. In the learning process, the theoretical aspects of modeling and simulation of IoT-based systems are presented. The course focuses on the assessment of the IoT based systems by use of Petri nets, Markov models and so on. In terms of methodology the following were given: syllabus, description of laboratory works and recommendations for independent learning of course.

The book is prepared for PhD students of universities on computer security, computer and software engineering which research the systems of the Internet of Things. It could be useful for lecturers and professors who conduct classes on corresponding courses.

Ref. – 63 items, figures – 40, tables – 9.

CONTENTS

ACRONIMS AND ABBREVIATIONS	3
INTRODUCTION	5
1 INTERACTION SIMULATION FOR IOT SYSTEMS	7
Laboratory work 1 Interaction simulation for WEB-oriented systems	7
1 Theoretical information	7
2 The task workflow	9
3 The content of the report	9
4 Self-assessment questions	9
Laboratory work 2 Interaction simulation with mobile applications	10
1 Theoretical information	10
2 The task workflow	12
3 The content of the report	13
4 Self-assessment questions	13
5 Recommended literature.....	14
2 PROGRAM TOOLS FOR THE SMART SYSTEMS SIMULATION	15
Laboratory work 1 Simulation of the Arduino devices in Proteus 8 Professional. Getting started	15
1 Brief theoretical information	15
1.1 Definition of ARDUINO set.....	16
1.2 Technical characteristics Arduino Mega 2560	17
1.3 Power supply Arduino Mega 2560	17
1.4 Inputs and outputs of Arduino Mega 2560	18
1.5 Arduino IDE	19
2 Program of developments and researches	22
3 Guidelines for a work	22
4 Individual tasks	26
5 Contents of the work report	27
6 Control questions	27
7 Recommended literature.....	28
Laboratory work 2 Simulation of the Arduino devices in Proteus 8 Professional. Using of light sensors in smart electrical systems	29

1 Brief theoretical information	29
1.1 Principle of operation of the sensor.....	29
1.2 Construction of module light sensor LM393.....	30
1.3 Connecting the sensor to the Arduino boards.....	31
1.4 Technical characteristics of the Light Sensor LM393.....	31
2 Program of developments and researches	32
3 Guidelines for a work	32
3.1 Upload sketch.....	32
3.2 Connection of the sensor	33
3.3 Testing of the sensor.....	34
4 Individual tasks.....	35
5 Contents of the work report.....	37
6 Control questions.....	37
7 Recommended literature.....	38
Laboratory work 3 Simulation of the Arduino devices in Proteus 8 Professional. Using of PIR Motion Sensors in smart electrical systems	39
1 Brief theoretical information	39
1.1 Construction of the PIR sensor.....	40
1.2 Principle of operation of the sensor.....	41
1.3 Connecting the sensor to the Arduino boards.....	41
1.4 Methods for setting sensor parameters	42
1.5 Work modes.....	43
1.6 Technical characteristics.....	43
1.7 Features of using.....	44
2 Program of developments and researches	44
3 Guidelines for a work	44
3.1 Upload sketch.....	44
3.2 Connecting of the sensor	46
3.3 Testing the sensor.....	46
4 Individual tasks.....	47
5 Contents of the work report.....	49
6 Control questions.....	49
7 Recommended literature.....	50
3 THREE-LEVEL SIMULATION OF IOT-BASED SYSTEMS WITH THE USE OF UML MODELS, PETRI NETS AND TEMPORAL LOGIC	51

Laboratory work 1 Modelling Internet of Things using UML diagrams at the stages of their architectural design	51
1 The purpose and objectives	51
1.1 Teaching tasks	51
1.2 Practical tasks	52
1.3 Research tasks	52
2 Preparation for laboratory work	52
3 Execution of laboratory work	52
3.1 Procedure for laboratory work performing	52
3.2 Examples of the tasks	54
4 Conclusion	60
5 Themes of individual tasks	61
6 Recommended literature.....	61
Laboratory work 2 Research of IoT systems in behavior by using Color Petri Net Tools	63
1 The purpose and objectives	63
1.1 Teaching tasks	63
1.2 Practical tasks	64
1.3 Research tasks	64
2 Preparation for laboratory work	64
3 Perform of laboratory work	65
3.1 The laboratory work order	65
3.2 Examples of the laboratory work execution	67
4 Conclusion	73
5 Report content requirements	73
6 Themes of individual tasks	73
7 Recommended literature.....	74
Laboratory work 3 Modeling Internet of Things systems using LTL temporal logic at the stages of their architectural design .	76
1 The purpose and objectives	76
1.1 Teaching tasks	76
1.2 Practical tasks	76
1.3 Research tasks	77
2 Preparation for laboratory work	77
3. Laboratory work execution	78
3.1 The order of the laboratory work performance	78
3.2 Examples of the tasks	79

4 Conclusion	85
5 Report content requirements	85
6 Themes of individual tasks	85
7 Recommended literature.....	86
4 MARKOV'S AND SIMULATION MODELING OF IOT SYSTEMS	88
Laboratory work 1 Final probabilities assessment of IoT based systems with use of Markov models mathematical apparatus ..	88
1 Objectives and tasks	88
2 Brief theoretical information	88
3 Execution order	89
4 Themes of individual tasks	95
5 Requirements to the content of the report	99
6 Testing questions	100
7 Recommended literature.....	100
Laboratory work 2 Availability function assessment of IoT based systems with use of Markov models mathematical apparatus	101
1 Objectives and tasks	101
2 Brief theoretical information	101
3 Execution order	102
4 Requirements to the content of the report	105
5 Testing questions	105
6 Recommended literature.....	105
COURSE PROGRAM	106
ABSTRACT	125
CONTENTS	126

Дрозд Олександр Валентинович
Маєвський Дмитро Андрійович
Маєвська Олена Юрївна
Мартинюк Олександр Миколайович
Табунщик Галина Володимирівна
Колісник Марина Олександрівна
Степова Ганна Сергіївна
Харченко Вячеслав Сергійович
Чопик Юлія Олександрівна
Нагачевський Нікіта Олександрович
Савельєв Артем Андрійович
Горошко Василь Володимирович

МОДЕЛЮВАННЯ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ

Практикум

(англійською мовою)

Редактор Маєвський Д. А.

Комп'ютерна верстка

Маєвський Д.А.

Ілляшенко О.О.

Зв. план, 2019

Підписаний до друку 27.08.2019

Формат 60x84 1/16. Папір офс. No2. Офс. друк.

Умов. друк. арк. 7,26. Уч.-вид. л. 7,81. Наклад 150 прим.

Замовлення 270819-1

Національний аерокосмічний університет ім. М. Є. Жуковського
"Харківський авіаційний інститут"
61070, Харків-70, вул. Чкалова, 17
<http://www.khai.edu>

Випускаючий редактор: ФОП Голембовська О.О.

03049, Київ, Повітрофлотський пр-кт, б. 3, к. 32.

Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців,
виготовлювачів і розповсюджувачів видавничої продукції
серія ДК No 5120 від 08.06.2016 р.

Видавець: ТОВ «Видавництво «Юстон»

01034, м. Київ, вул. О. Гончара, 36-а, тел.: +38 044 360 22 66
www.yuston.com.ua

Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців,
виготовлювачів і розповсюджувачів видавничої продукції
серія ДК No 497 від 09.09.2015 р.