TARTU UNIVERSITY NARVA COLLEGE

Information technology systems development

Daria Filatova

P2NC.01.087 Digital Logic and Digital Systems course creation

Graduation Thesis (8 EAP)

Supervisor: Deniss Ruder

Narva 2023

P2NC.01.087 Digital Logic and Digital Systems course creation

Abstract:

This work is a Graduation thesis with the title P2NC.01.087 Digital Logic and Digital Systems course creation. The purpose of this work is to change the current course. The old version is about visual programming and theory, which mostly appears in the other subject of the "Information technology systems development" program - Sissejuhatus Erialaõpesse. In this new course, students will learn C++ programming language for microcontrollers. Using a traditional classroom methodology, we created a course consisting of 7 lessons and practices. The course ends up with an exam. Practice works involve using basic constructions of C++ programming language and microcontroller programming. Using the Arduino microcontroller family and its components, we created a practice course to interest and force students to solve tasks independently. This work provides reasons for changing the old version of this course and the new course structure with goals and the content of the lessons. There also were created seven presentations for each lesson.

Keywords:

C++ programming language, Arduino, College course

Table of Contents

1. Problem statement

Narva College is currently restructuring its “Information technology systems development” program, which involves reviewing almost all courses. P2NC.01.087 “Digital logic and digital systems” has been moved to the second semester of the first year of study. Therefore, it is now in its logically explained place in the curriculum. It is necessary to reconsider and redesign the course to fit the new program.

The present Graduation thesis titled “P2NC.01.087 Digital Logic and Digital Systems course creation” is dedicated to create an entirely new structure and content for the course.

The main goal of recreating this course is to create a compelling and practical course that will help students understand the basics of digital logic and digital systems without diving into a profound theory. Upon completing this course, students should be able to program circuits and understand their logic, use boolean logic and functions in practice, and understand the main ideas of digital systems.

The currently available course is not connected to the new studying program and looks like a course from a different program. If the new course starts in the second semester of the studying year 2022-2023, this is a serious issue.

According to a bunch of students we interviewed, around 65% of those who attended the spring semester of 2021-2022 passed the final exam for this course on the first attempt.

The problem is in the course goals and content. There is a lot of theory in this course rather than practice on how to use the technologies outlined in the course content. Sometimes, the theory doesn’t apply to the student's specialization or needs to be updated. It also contains some theory part, which mostly appears in the other subject of the “Information technology systems development” program - Sissejuhatus Erialaõpesse, which students learn in the first semester.

Those problems make this course non-effective and not interesting for students. In preparation for the creation of this course, a few students who have already completed it were interviewed. Their answers confirm what was said above. If we don't create a new course, students will continue to study this course in its old version, which means they will get a lot of theory information without trying it in practice.

We need to create a new course structure and content so that it can be connected with other courses, students are currently studying. Set up the new goals and outcomes for this course, to make this course more efficient and up-to-date with new ideas and technologies.

As a student who already went through the old version of this course, we want to ensure new students will be more involved in the studying process at the university. So they can get more helpful and valuable information to achieve a new level of professional competence.

2. Existing Courses

To understand what this course will be like, we need to check out existing courses like this one.

Reviewing existing courses should begin with the old version of this course. The current P2NC.01.087 course has the following goals:

- Provide an overview of the past circuit designs and embedded devices and about modern project progress, methods and tools;
- To provide a comprehensive overview of the modeling of digital systems;
- To provide a comprehensive overview of the hardware description language VHDL;
- Teach hardware description language use in various designs and abstractions;
- To provide an overview of the synthesis of digital systems and register transfer logic;
- Teach the use of industrial simulations and synthesis tools using FPGA.

So the old version of the course Digital Logic and Digital Systems covers a lot of theory related to VHDL language, circuits, and boolean logic. During an exam, students deal with a massive amount of boolean logic and numeric systems tasks, circuit schemas, and theory questions connected to the history of circuit design and its progress.

Other existing solutions can also be found in the curriculum of the University of Tartu, so this is what we will check next. The curriculum of the University of Tartu contains the course LOTI.05.041 Digital Logic (3 ECTS). Reviewing this course, it was discovered that it has the same description on SIS as the course P2NC.01.087 “Digital logic and digital systems“ (2 ECTS). This leads us to the conclusion that the same content is taught in two different courses with different time loads. We can consider this because the course LOTI.05.041 was updated earlier than P2NC.01.087. Perhaps this partly explains the low effectiveness of the course, because a course with a higher time load is taught on the same program. And so, students may be unable to cope with the extra workload on the theory and practical tasks side.

Also, during the research of existing solutions, we review online courses. One of them is the Udemy course Digital Logic Circuits and Design. The objective of the course is to make everyone design a digital circuit efficiently using various components. This course contains information about the function of logic circuits and how to design them. The course has 9 hours of video lectures, from number systems to logic designs of various circuits.

The second found course is on the Coursera, Digital Systems: From Logic Gates to Processors. Approximately 28 hours are required to complete the course. As an Udemy course, it explains the logic of different circuits and how to program them using high-level languages such as VHDL.

These online courses are up-to-date and have comprehensive information about circuits and their specifics. These courses give necessary knowledge about their subject in a short time frame, but there is a lack of option of trying all this theory in practice. These courses can be considered a good base for theory, but it is always better to learn something in practice, which we want to achieve by changing the P2NC.01.087 “Digital logic and digital systems“ course.

3. Methodology

In this work, choosing a suitable methodology is critical. It is essential to understand what will work best for the theory and practice parts of a classroom, as well as to validate students' knowledge and create tasks that fit students' capabilities.

The choice of methodology depends directly on the aims and objectives of the training. The course should be more practical than now, as students already know about hardware parts when they begin the course. We will connect the knowledge of hardware with the knowledge of software development in this course. The most significant part of the usual classroom will be practical, where students can try to practice the knowledge they just received from a small theory part.

For the course creation, we will overview several possible methodologies:

- traditional classroom with a focus on practice;
- project-based learning methodology;
- flipped learning methodology.

3.1. Traditional classroom with a focus on practice

Traditional classroom training has been and will continue to be the foundation for students' knowledge development.

In traditional lessons, a lot of information can be presented in a short amount of time. With this kind of learning, students acquire knowledge in a ready-made form. It also involves learning and reproducing knowledge and applying it in similar situations. The significant disadvantage of this type of learning is that it focuses more on memory than on thinking. This kind of learning also contributes little to developing creativity, independence, and activity. The typical tasks are: insert, highlight, underline, remember, reproduce, solve by example, etc. As practice shows, the amount of information given is greater than the capacity to absorb it

Adding more practice to these classes will help avoid traditional classrooms' passive listening moments. Students will also be motivated to think, process, and apply the information provided.

3.2. Project-based learning methodology

The project-based method is learning by setting up a problem and solving it step by step, which ultimately leads to a concrete, practical result. The main feature of the project-based learning method is the students' independent acquisition of knowledge directly through practice.

Advantages of project activities:

— self-education and self-monitoring skills;

— The real technological chain is modeled: task-result;

— helps to develop creativity, and thinking outside the box;

— interest in learning activities;

— The method simulates real-life situations.

Disadvantages of project activities:

— the teacher's workload increases;

— the student is often put in a stressful situation (overestimation of possibilities, technical overlaps);

— psychological communication problems;

— Lack of evaluation criteria since such tasks need to be standardized.

3.3. Flipped classroom learning model

Flipped learning is a model where students study theory at home as homework and then do practical exercises in face-to-face classes.

Advantages of flipped learning:

— Students gain new knowledge at their own pace. The student can watch the video at a convenient time, can pause the viewing, listen several times to unclear points, write down questions, and ask them in class;

— In the flipped classroom system, students are no longer passive listeners. They have to do their work to acquire new knowledge: watch a video, read an article, listen to an interview, or look for more information on the internet;

— the material can be studied at any time. This comes in handy if students are sick, participating in competitions, or away. Students can keep up with the rest of the class and won't miss any important information;

— Maximum practice! All lesson time is devoted to practical tasks.

Disadvantages of flipped learning:

— Not all students are sufficiently independent in learning new topics

4. Selected methodology

This course's lead methodology was chosen as a traditional classroom with a practice-oriented approach. Classes will include a brief theory for the following practice work.

Flipped classes won't work as expected, because the effectiveness of the flipped classroom could be higher. Project-based learning is not appropriate for this course also since the course will involve small practical labs rather than projects.

We also decided that the Arduino Uno microcontroller is suitable for this course. The Arduino microcontroller family consists of various microcontrollers and additional components. The Arduino Uno is simple to use, which helps us avoid theory from physics and concentrate on programming and understanding concepts of hardware programming.

5. Course structure

For the course also was chosen Arduino microcontroller, because Arduino has an incredible amount of infrastructure. This amount of Arduino resources, software and hardware make it easier to interest students in studying because students will have a lot of opportunities to make something big and individual during the course if they want to.

This course will be designed to have 9 lessons. The last one is for the exam. One lesson takes 90 minutes.

1. lesson: Introduction. Introduction to course goals, requirements, and structure. Installing necessary software. Introduction to the Arduino board. First introduction to the C++ language, variables and types in C++.

2. lesson: Cycles and if-else constructions in C++. Additional components for the Arduino board. Breadboard. Connecting breadboard with additional LED to the Arduino board.

3. lesson: Reading input from the User. Connecting and using the button to control the LED. While loops in C++.

4. lesson: Arrays in C++. How to use arrays and declare. Arrays in Arduino.

5. lesson: Interrupts. How to use interrupts. Hardware and software interrupt.

6. lesson: Single-digit display. Introduction to one digit 7-segment LED display and how to connect it. Switch construction in C++.

7. lesson: Advanced use of buttons.

8. lesson: Additional lesson to finish practice tasks and prepare for the exam.

9. lesson: The exam. There will be two parts to the exam, one for theoretical knowledge and one for practice.

5.1. General info about the course

5.1.1 Objectives

- introduce to students C++ programming language;
- give students understating of types and main concepts of programming for hardware;
- introduce hardware description languages using various design and abstraction processes;
- give students the possibility to use their knowledge in practice;

— give students an experience in programming for hardware using machine-depended technologies.

5.1.2 Learning outcomes

1. Is able to program hardware using embedded technologies.
2. Is familiar with Arduino board architecture and its additional components.
3. Can independently solve practice tasks for Arduino and create own project.
4. Is familiar with simplified C++ language syntax.
5. Is familiar with different types of variables.
6. Understands pins on the Arduino board and knows how to use the Arduino breadboard.

5.1.3 Brief description of the content

Introduction to C++ programming language main constructions and functions.

Arduino schematic overview.

Arduino basic components.

Implementation of C++ programming language constructions in the Arduino board programming.

5.2. Lesson 1. Introduction

Lesson equipment:

— the Arduino board.

Lesson plan:

— make clear to students what this course is about;

— show students how to work with variables and types in C++;

— circuit schematics;

— connect the Arduino board and write the first program (2 examples);

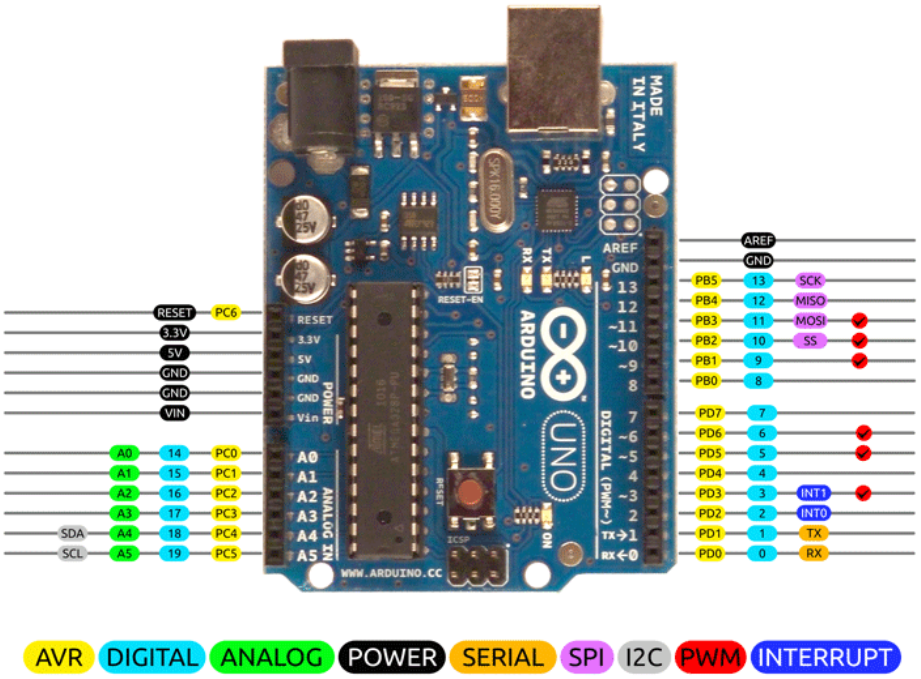
— at the end of the lesson, students should be able to solve the first practice task with a blinking LED, using variables.

Lesson structure:

1. Present the course, its goals, and outcomes for students;

2. Introduction to Arduino. Show different boards and resources to interest students in individual projects;

3. Show and explain the pins on the Arduino board (picture 1).



Picture 1. The Arduino UNO pins.

4. Installation of Arduino IDE (if not installed);

5. Connecting the board and installation of drivers;

6. Create the first program in Arduino IDE;

Launch Arduino IDE and create a blank project. Board specification in the IDE (Tools - Board - your board).

7. Introduction to void setup and void loop. Writing the first code to turn on the LED on the Arduino board (picture 2).

```
void setup() {
  pinMode(13,OUTPUT);
}

void loop(){
  digitalWrite(13,HIGH);
}
```

Picture 2. The first code with Arduino.

8. The task for students: Change the program to turn off the LED.

9. Introduction to variables and types in C++. How and where to declare and use variables in the program.

10. Write a program to see how variables work in the program (picture 3).

```
1   int counter = 0;
2
3   void setup() {
4     Serial.begin(9600);
5   }
6
7   void loop(){
8     Serial.print("Counter: ");
9     Serial.println(counter);
10    delay(10);
11    counter++;
12  }
```

Picture 3. The code shows how to use variables in the program.

11. First practice task. Write a program for the Arduino to make the LED on the board blink. Using variables, make the LED blink twice with a delay of 1 second, then turn it off for 3 seconds and turn it on again for 3 seconds. The program should be repeated every 5 seconds (picture 4).

```
int LEDpin = 13;
int waitTimeOneS = 1000;
int waitTimeThreeS = 3000;
void setup() {
  pinMode(13,OUTPUT);
}
void loop(){
  digitalWrite(13,HIGH);
  delay(waitTimeOneS);
  digitalWrite(13,LOW);
  delay(waitTimeOneS);
  digitalWrite(13,HIGH);
  delay(waitTimeOneS);
  digitalWrite(13,LOW);

  delay(waitTimeThreeS);
  digitalWrite(13,HIGH);
  delay(waitTimeThreeS);
  digitalWrite(13,LOW);
  delay(waitTimeOneS);
}
```

Picture 4. Example of the code for practice.

5.3. Lesson 2. Using A Breadboard

Lesson equipment:

— the Arduino board;

— the Breadboard;

— 2 wires;

— 1 resistor;

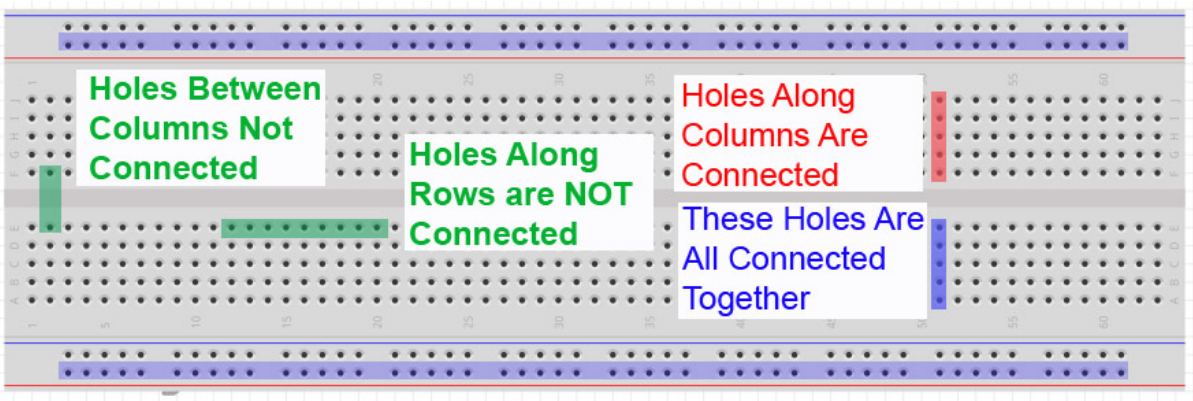
— 1 LED.

Lesson plan:

- show students how to work with cycles and if-else constructions in C++;
- circuit schematics, use an additional board - breadboard;
- connect the Arduino board to an additional board through pins;
- show students how to connect additional components to the Arduino;
- at the end of the lesson, students should be able to solve the second practice task.

Lesson structure:

1. Introduction to a breadboard. Why it is needed and how to use it with Arduino UNO. (picture 5)



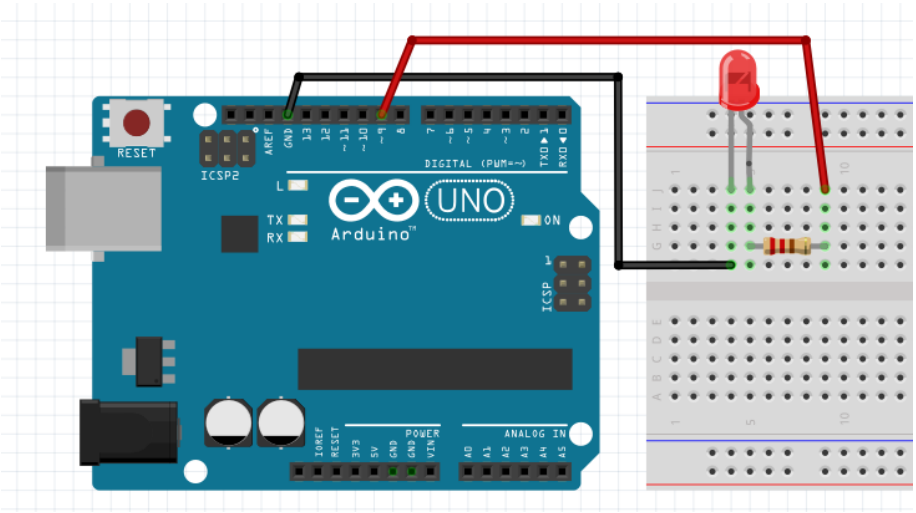
Picture 5. The breadboard.

2. Connecting a breadboard to the Arduino board

We can connect any pin between pins on the Arduino from 0 to 13 to the Breadboard. Let's use pin 9 for connection. The wire goes to pin nine and any A pin on the breadboard.

3. Connecting an additional LED to the breadboard

It is important how we connect the LED to the breadboard. One leg of the LED is longer than the other. The longer LED leg goes to the resistor, and the second to the ground. Now we need to connect the resistor. One leg goes to the column where the red wire is connected, and the second to the column where the LED's longer leg is connected. Now, let's set up the ground with the wire. We will connect the wire to the column with the shorter leg of the LEG and GND pin on the Arduino board.



Picture 6. Scheme of the connection breadboard, LED, and Arduino board.

4. Presenting if-else construction in C++ to students

5. Writing the fade-in and fade-out of the LED program with students using an if structure (picture 7)

```
int led = 9; // The digital pin to which the LED is connected
int brightness = 0; // Brightness of LED is initially set to 0
int fade = 5; // By how many points the LED should fade
void setup() {
  pinMode(led, OUTPUT); //pin of the LED is set as output pin
}
void loop() {
  analogWrite(led, brightness); // set the brightness of LED
  brightness = brightness + fade; //Increase the brightness of LED by 5 points
  if (brightness <= 0 || brightness >= 255) { // check the level of brightness
    fade = -fade;
  }
  delay(30); // Wait for 30 milliseconds
}
```

Picture 7. The fade-in and fade-out of the LED program.

6. Presenting cycle construction in C++ to students

7. Change the previous program to implement cycle construction (picture 8)

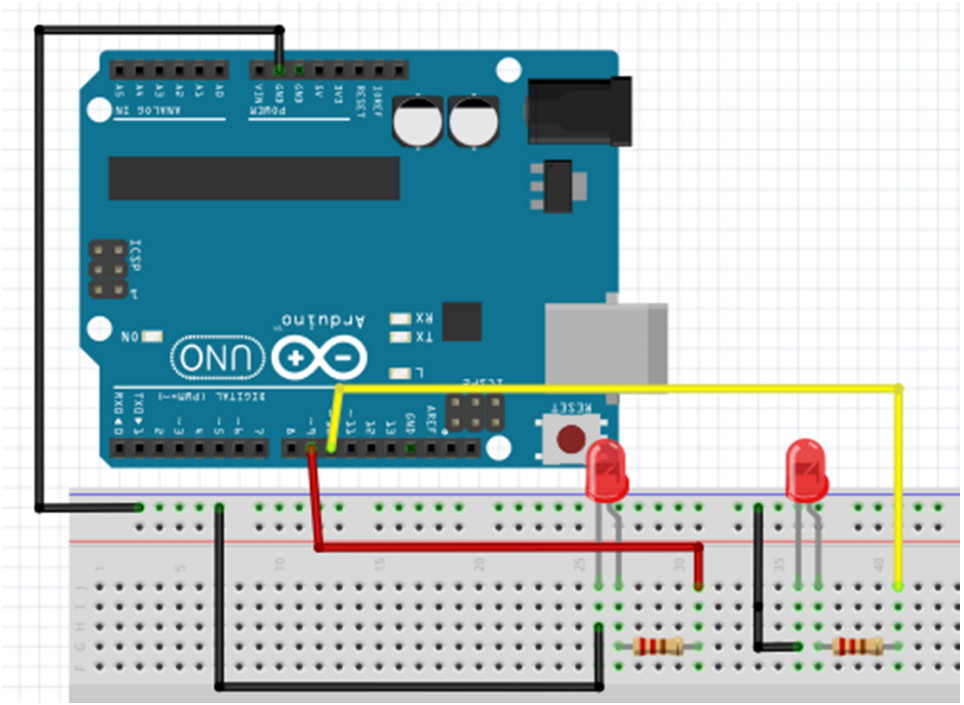
```
int led=9; // The digital pin to which the LED is connected

void setup() {
  pinMode(led, OUTPUT); //pin of the LED is set as output pin
}

void loop() { // The loop function runs again and again
  for (int fade=0; fade<=255; fade=fade+5) {
    analogWrite (led, fade); // Change the brightness of LED by 5 points
    delay (30);
  }
}
```

Picture 8. The fade-in and fade-out of the LED program within for cycle.

8. Explaining using a ground while operating with more, than 1 component to prepare students for the practice task (picture 9).



Picture 9. Connection of more than 1 component to the breadboard.

9. Practice task: Connect the second LED to the breadboard. Using cycle, make the first LED blink five times, then blink three times with the second LED (picture 10).

```
int led1 = 9;
int led2 = 10;
void setup(){
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
}
void loop() {
  for(int i = 0; i < 5; i++){
    digitalWrite(led1, HIGH);
    delay(2000);
    digitalWrite(led1, LOW);
    delay(1000);
  }
  for(int i = 0; i < 3; i++){
    digitalWrite(led2, HIGH);
    delay(2000);
    digitalWrite(led2, LOW);
    delay(1000);
  }
}
```

Picture 10. Example of the code for the second practice.

5.4. Lesson 3. Reading input from the User. Using the button

Lesson equipment:

— the Arduino board;

— the Breadboard;

— 5 wires;

— 1 resistor;

— 1 LED;

— 1 button.

Lesson plan:

- show students “how to get information from user“ construction;
- show while loop in C++;
- connecting button to the Arduino breadboard;
- using the button to turn on/off the LED;
- practice task with two LEDs controlled by a button.

Lesson structure:

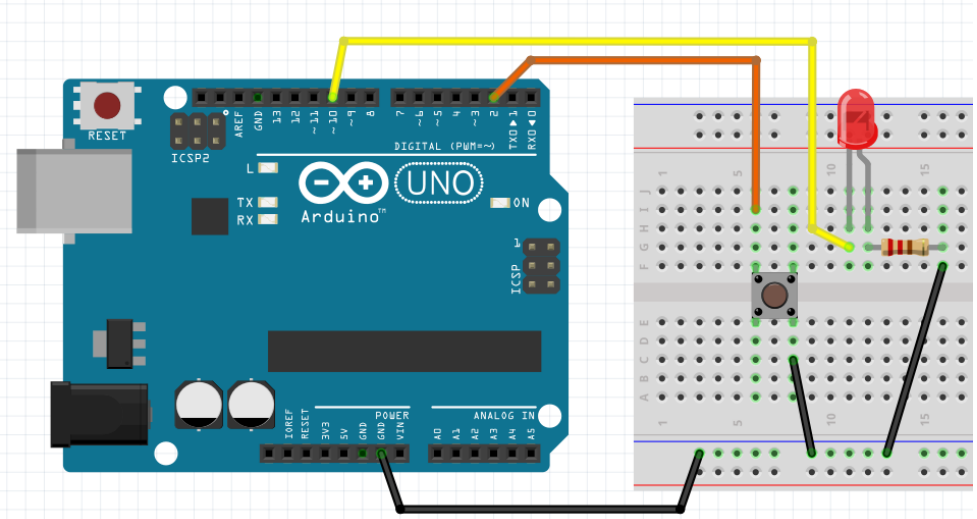
1. Introduction to while loop

2. Show how to get input from a user (picture 11)

```
int led=9;
int blinks;
void setup() {
  pinMode(led, OUTPUT);
}
void loop() {
  Serial.begin(9600);
  Serial.println("How many times do you want the LED to blink?");
  while(Serial.available()==0){
    blinks = Serial.parseInt();
  }
  if (blinks > 0){
    for(int i = 0; i < blinks; i++){
      digitalWrite(led, HIGH);
      delay(2000);
      digitalWrite(led, LOW);
      delay(1000);
    }
  }
  Serial.end();
}
```

Picture 11. The code for getting input from the user.

3. Connect the button and LED to the breadboard (picture 12)



Picture 12. Scheme of the connection button and LED to the Arduino.

4. Write a program with a button that controls the LED (picture 13)

```
bool buttonWasUp = true;
bool ledEnabled = false;
int led = 9;
int btn = 2;

void setup() {
  pinMode(led, OUTPUT);
  pinMode(btn, INPUT_PULLUP);
}

void loop() {
  bool buttonIsUp = digitalRead(btn);
  if (buttonWasUp && !buttonIsUp){
    delay (10);
    buttonIsUp = digitalRead(btn);
    if (!buttonIsUp){
      ledEnabled = !ledEnabled;
      digitalWrite(led, ledEnabled);
    }
  }
  buttonWasUp = buttonIsUp;
}
```

Picture 13. A code with an LED controlled by a button.

5. Practice task. Connect to the Arduino 2 LEDs and 1 button. One of the LEDs should be HIGH by default, and the second should be LOW. When you push the button, the first LED should be on LOW, and the second should be HIGH (picture 14).

```
bool buttonWasUp = true;
bool led1Enabled = true;
int led1 = 9;
int led2 = 10;
int btn = 2;

void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(btn, INPUT_PULLUP);
}

void loop() {
  bool buttonIsUp = digitalRead(btn);
  if (buttonWasUp && !buttonIsUp) {
    delay (10);
    buttonIsUp = digitalRead(btn);
    if (!buttonIsUp) {
      led1Enabled = !led1Enabled;
    }
  }
  buttonWasUp = buttonIsUp;
  digitalWrite(led1, led1Enabled);
  digitalWrite(led2, !led1Enabled);
}
```

Picture 14. Example of the code for the practice of lesson 3.

5.5. Lesson 4. Arrays

Lesson equipment:

— the Arduino board;

— the Breadboard;

— 13 wires;

— 4 resistors;

— 4 LEDs;

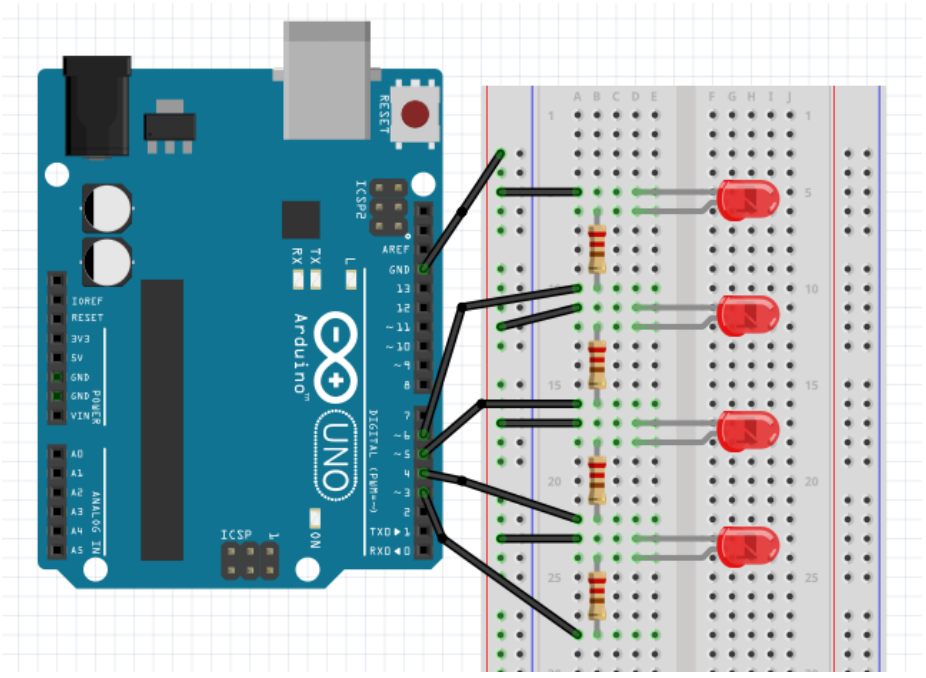
— 3 buttons.

Lesson plan:

- show students how to work with arrays in C++;
- write a program with 4 LEDS using arrays;
- practice task: write a program with a reaction game.

Lesson structure:

1. Introduction to arrays in C++. Declaration and use.
2. Connecting 4 LEDs to the Arduino breadboard (picture 15).



Picture 15. Scheme of the connection 4 LEDs.

3. Write a program with 4 LEDs that will be switched on in sequence (picture 16).

```
int ledPins[] = { 3, 4, 5, 6};
int pinCount = 4;

void setup() {
  for (int i = 0; i < pinCount; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
}

void loop() {
  for (int i = 0; i < pinCount; i++) {
    digitalWrite(ledPins[i], HIGH);
    delay(100);
    digitalWrite(ledPins[i], LOW);
  }
  for (int i = pinCount - 1; i >= 0; i--) {
    digitalWrite(ledPins[i], HIGH);
    delay(100);
    digitalWrite(ledPins[i], LOW);
  }
}
```

Picture 16. Code with LEDs in sequential turn-on.

4. Practice task: Write a program with a reaction game. Connect 3 LEDs and 3 buttons to the breadboard. The game starts five seconds after the program has started. During the game, one random LED lights up for 1 second. If the user clicks the button underneath it, he gets the point. The game ends after five tries. The score is displayed in the console.

5.6. Lesson 5. Interrupts

Lesson equipment:

— the Arduino board;

— the Breadboard;

— 6 wires;

— 1 resistor;

— 1 LED;

— 2 buttons.

Lesson plan:

- show students what are interrupts and how to use them;
- write a program with an interrupt made by a button;
- write a program with a timer interrupt and the LED;

- practice task: Program with interrupts by two buttons.

Lesson structure:

1. Introduction to interrupts and why they are needed.

2. Hardware and software interrupt.

3. Connect the button to the Arduino. Write a program with a hardware interrupt (picture 17).

```
int counter = 0;
void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(0, buttonInterrupt, FALLING);
}

void buttonInterrupt() {
  counter++;
}

void loop() {
  Serial.println(counter);
  delay(1000);
}
```

Picture 17. Code with hardware interrupt.

4. Pin change interrupts.

5. Introduction to software interrupt.

6. Timer interrupts.

7. Practice task: Write a program with two buttons. Each button has its interrupt. The right button should turn on the LED, and the second button should turn it off.

5.7. Lesson 6. Single-digit display

Lesson equipment:

— the Arduino board;

— the Breadboard;

— 19 wires;

— 9 resistors;

— 1 LED,

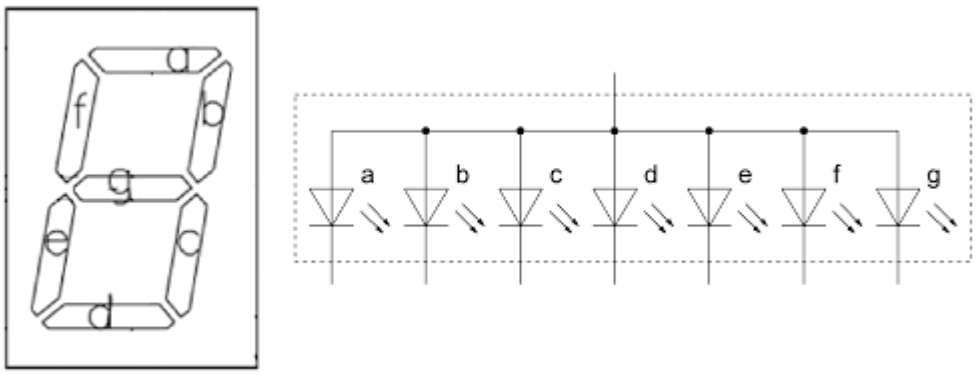
— 1 one-digit 7-segment LED display.

Lesson plan:

- show students what one digit 7-segment LED display is;
- show how to connect a single-digit display to the breadboard;
- write a program to check the right connection of the display to the breadboard (blink with a dot);
- change the program to turn on all LEDs on display;
- explaining the work of the switch construction in C++;
- Practice task: Countdown with additional LED.

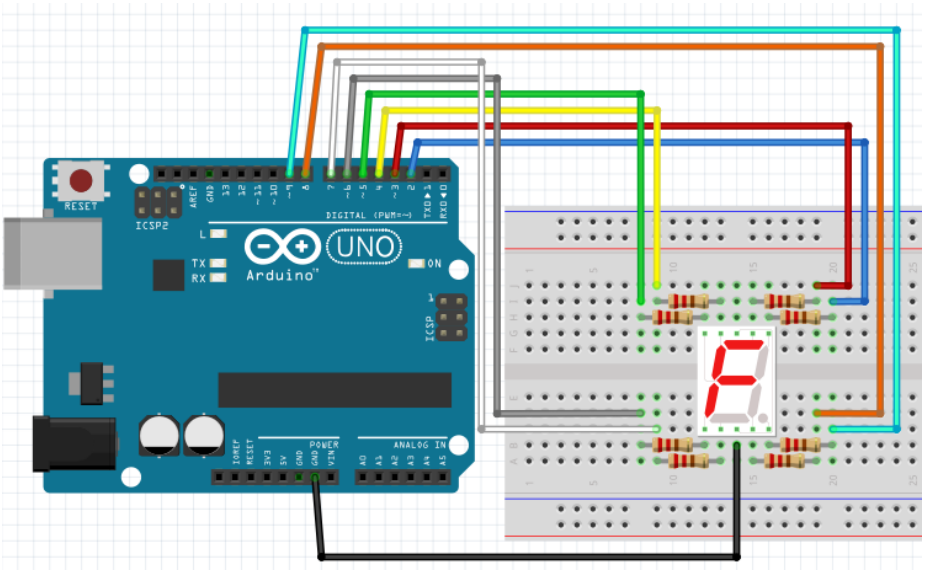
Lesson structure:

1. Introduction to one-digit 7-segment LED display and its pins (picture 18).



Picture 18. One-digit 7-segment LED display pins.

2. Connect the pins of the LED display to the breadboard and the Arduino (picture 19).



Picture 19. Scheme of the connection LED display to the Arduino.

3. Write a program to check the connection. Blink with a dot on display (picture 20).

```
int dot= 9;
int del = 300;

void setup() {
  pinMode(dot, OUTPUT);
}

void loop() {
  digitalWrite(dot, LOW);
  delay(del);
  digitalWrite(dot, HIGH);
  delay(del);
}
```

Picture 20. Code with blinking dot on the LED display.

4. Change the program to turn on all LEDs on display (picture 21).

```
int dot= 9;
int del = 300;
int leds[] = {2, 3, 4, 5, 6, 7, 8};

void setup() {
  pinMode(dot, OUTPUT);
  for (int i = 0; i < 7; i++) {
    pinMode(leds[i], OUTPUT);
    digitalWrite(leds[i], HIGH);
  }
}

void loop() {
  digitalWrite(dot, LOW);
  delay(del);
  digitalWrite(dot, HIGH);
  delay(del);
}
```

Picture 21. Code with turning on all LEDs on display.

5. Introduction to switch construction.

6. Write a program to display 1 (picture 22).

```
int dot= 9;
int del = 300;
int leds[] = {2, 3, 4, 5, 6, 7, 8};
void setup() {
  pinMode(dot, OUTPUT);
  for (int i = 0; i < 7; i++) {
    pinMode(leds[i], OUTPUT);
  }
}
void digit(int x) {
  switch (x) {
    case 1:
      digitalWrite(leds[0], HIGH);
      digitalWrite(leds[6], HIGH);
      break;
  }
}
void loop() {
  digit(1);
}
```

Picture 22. Code with turning on all LEDs on display.

7. Practice task: write a program to countdown from 9 to 0 on display using switch construction. Add the LED, which should blink on 3-2-1. When the display shows a 0 set delay for a few seconds, the LED should be turned on for these seconds. After this program should turn off the LED and turn on all LEDs on display, then blink with the dot and start again.

5.8. Lesson 7. Advanced use of buttons

Lesson equipment:

— the Arduino board;

— the Breadboard;

— 8 wires;

— 4 resistors;

— 1 LED,

— 1 button.

Lesson plan:

- Reading double click on a button;
- Reading time of pushing a button;
- Practice task: Write a program with advanced use of a button and LED.

Lesson structure:

1. Learn how to read a sequence of clicks on the button. Write a program that will count clicks on a button and display the number of last clicks.
2. Write a program. which will read short or long click on the button.
3. Practice task: Write a program with a button and LED. The LED has to have 4 states: fade-in/out, blinking, HIGH, and LOW. Long pressing the button should change the state. The delay between the first and second states should be changed by pressing the button twice.

5.9. Lesson 8. Additional lesson

If students need more time to complete their practice tasks, this lesson will help them. For students who have completed all tasks, the teacher can prepare small tasks to help them prepare for the exam and refresh their knowledge.

5.10. Lesson 9. The exam

There are two parts to the exam. The first part tests the basic theoretical knowledge of the Arduino board and C++ programming language. The second part of the exam is to solve two practical tasks. The first task will be accessible without or with one of the additional components. The second task will be more complicated - with several components connected to the Arduino board. A simple example of the exam can be found in Appendix 1.

6. Example of this course on ÕIS

Course type	**Regular course**
Frequency of teaching the course	-
Course duration in semesters	**1**
Structural unit	**Narva College (SVNC)**
Final assessment scale	**Non-differentiated (pass, fail, not present)**
Can previous learning be recognised?	**Yes**
Can be taken by continuing education learners?	**No**
Languages of instruction	**English**
Form of study	**Regular studies**
Levels of study	**Bachelor's studies**

General info

Objectives

introduce to students C++ programming language;
give students understating of types and main concepts of programming for hardware;
introduce hardware description languages using various design and abstraction processes;
give students a possibility to use their knowledge in practice;
give students an experience in programming for hardware using machine-depended technologies.

Learning outcomes

Is able to program hardware using embedded technologies.
Is familiar with Arduino board architecture and its additional components.
Can independently solve practice tasks for Arduino and create own project.
Is familiar with simplified C++ language syntax.
Is familiar with different types of variables.
Understands pins on the Arduino board and knows how to use the Arduino breadboard.

Brief description of content

Introduction to C++ programming language main constructions and functions.
Arduino schematic overview.
Arduino basic components.
Implementation of C++ programming language constructions in the Arduino board programming.

Forms and volume of study in hours

Volume 2 ECTS | 52 h

Lectures	**16 h**
Practical classes	**6 h**
Independent work (incl. e-learning)	**30 h**

Assessment

List of independent assignments and instructions for their completion

7 practice tasks

Assessment methods and criteria

Requirements to be met for final assessment

For participating in the exam student has to complete all practice task and attend at least 80% of classes

Final assessment scale

Non-differentiated (pass, fail, not present)

Assessment results available from

Info is missing

Formation of final result

Exam

Options for taking tests/exams at later date

Re-examination

4. Conditions for course completion

For participating in the exam, the student must solve all practical tasks of the course and attend at least 80% of the lessons. In case of incomplete lessons attendance, the teacher decides whether to allow the student to take the exam.

To complete the exam, students should have 75% of correct answers for the theoretical part and solve both practical tasks.

CONCLUSION

In the graduation thesis was set up a problem statement with the main problem - the old version of the course is overwhelmed with a lot of theory, and students don't find it interesting. It also covers the theory part of the other course, which students learn in the previous semester. This problem showed itself in students' low performance and attendance in the course. The course required reconstruction and the addition of practical tasks so students could try out the new knowledge on the spot.

We found and reviewed a few existing solutions and tried to find the most helpful practices from them. Also, the existing course taught in Narva College is also taught as the course with a higher amount of ECTS with the same content and outcomes. It could be a cause of the low efficiency of this course.

Then was researched different methodologies of teaching. The goal was to find the most suitable methods, which we could use during the classes. So, as the major methodology was chosen traditional classroom with a practice-oriented approach. Lessons should include a brief theory for the following practice work.

All this research helped to create a complete course with practical assignments for students. Each lesson has goals and outcomes which help students to be able to solve practice tasks at the end of the lesson independently.

To summarize, the result of the work can be interpreted as follows: was created practical course P2NC.01.087 Digital Logic and Digital Systems. This course has all the basic knowledge of C++ programming language for microcontrollers and Arduino UNO board with its components. But only with a semester of testing this course, can we be sure whether the amount of lessons and tasks is enough.

Created structure and content in this work contain 8 practice tasks and valuable knowledge. Students will acknowledge with all additional basic components of the Arduino family and the Arduino Uno board itself.

For this course we created 9 lessons. The last of them is the exam. Each lesson has an introduction to different components of the Arduino board or C++ programming language concepts. For each lesson was created presentation with theory, schemas, and examples of the code, except for the lesson before the exam. This lesson is considered for students who need more time to complete their practice tasks. The student is allowed to participate in the exam after having completed all practical assignments. For students who have completed all tasks, the teacher can prepare small tasks to help them prepare for the exam and refresh their knowledge.

To improve this course, we could add tests at the beginning of each lesson to check the learning progress. Also, we could add even more different components. To achieve this, we could increase the amount of ECTS for this course or create a following advanced course in the frame of this graduation thesis. But the main objective here is to interest students in

hardware programming and show them that their code can be used beyond PC and smartphone applications.

All in all, the goals of the problem statement were achieved, and was created a new version of the P2NC.01.087 Digital Logic and Digital Systems course.

REFERENCES

1. Simon Monk. *Programming Arduino: Getting Started With Sketches,* 2, McGraw Hill TAB, 2016.
2. John Boxall. *Arduino Workshop: A Hands-On Introduction with 65 Projects,* 2013.
3. *LOTI.05.041 Digital Logic and P2NC.01.087 Digital Logic and Digital Systems course information*. Available at https://ois2.ut.ee/, accessed November 30, 2022.
4. Sujithkumar, Last updated 12.2022. *Digital Logic Circuits and Design course.* Available at https://www.udemy.com/course/digital-logic-circuits-and-design, accessed November 30, 2022.
5. Valderrama E., Deschamps J., etc, *Digital Systems: From Logic Gates to Processors.* Available at https://www.coursera.org/learn/digital-systems, accessed November 30, 2022.
6. Randal Bryant, David O'Hallaron. *Computer Systems: A Programmer's Perspective,* 3, Pearson, 2015.
7. *Arduino official site.* Available at https://www.arduino.cc, accessed December 29, 2022.
8. Paul McWhorter, 2014. *Arduino lessons.* Available at https://toptechboy.com/arduino, accessed December 22, 2022.
9. Andrew Redfern, *The Essential Guide to Classroom Practice,* Routledge, 2015.

APPENDICES

Appendix1. Example of exam

Test. Theory part:

1. What are Arduino's two mandatory program functions?

- void onstart() and void loop()
- **void setup() and void loop()**
- void start() and void work()
- void setup() and void work()

2. Which line will pause a program for 2,5 seconds?

- **delay(2500)**
- pause(2500)
- pause(2.5)
- delay(2.5)

3. What will the next program do?

```
void setup() {
  pinMode(13,OUTPUT);
}

void loop(){
  digitalWrite(13,HIGH);
}
```

- turn off the LED
- make the LED blink
- turn on the Arduino board
- **turn on LED**

4. The … construction is used to repeat a block of statements enclosed in curly braces.

- if-else
- **for loop**
- void setup()

- switch

5. Write a HIGH or LOW value to a digital pin.

- pinMode()
- delay()
- turnOn()
- **digitalWrite()**

6. The ... function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc.

- onstart()
- **setup()**
- loop()
- void()

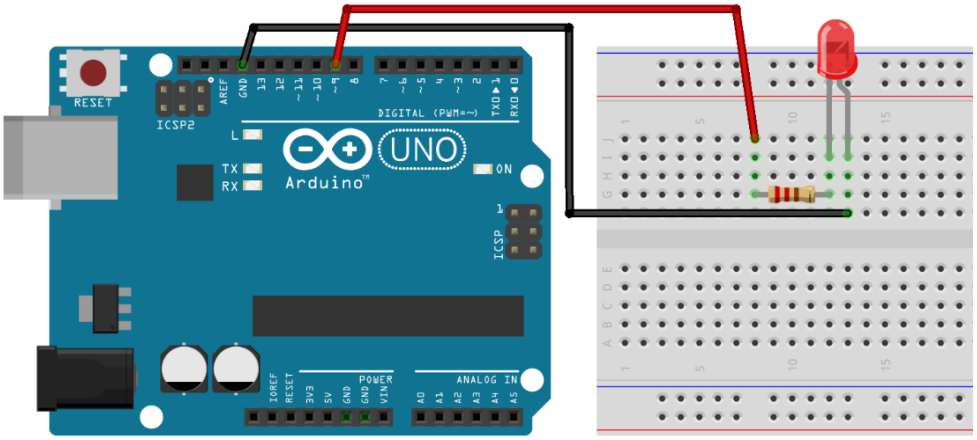
7. Configures the specified pin to behave either as an input or an output.

- **pinMode()**
- pinWork()
- pinState()
- pinSetup()

8. The ... function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc.

- onstart()
- **setup()**
- loop()
- void()

9. Which program will turn on the LED on the scheme?




```
int led = 9;
void setup()
{
pinMode(led, OUTPUT);
}
void loop()
{
  digitalWrite(led, LOW)
}
```

```
int led = 9;
void setup()
{
pinMode(led, OUTPUT);
}
void loop()
{
  digitalWrite(led, HIGH)
}
```

```
int led = 7;
void setup()
{
pinMode(led, OUTPUT);
}
void loop()
{
  digitalWrite(led, HIGH)
}
```

```
int led = 7;
void setup()
{
pinMode(led, OUTPUT);
}
void loop()
{
  digitalWrite(led, LOW)
}
```

10. … is used when two different codes have to be executed based on a condition.

- for
- **if-else**
- while
- random

Practice part:

1. Write a program that counts clicks on the button and displays the number of clicks.
2. Write a program with an LED and a button. Each button click should increase the delay of turned-off LED time.

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, _____Filatova Daria_____,
(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose _____P2NC.01.087 Digital Logic and Digital Systems course creation_____,
(lõputöö pealkiri)

mille juhendaja on _____Deniss Ruder_____,
(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Filatova Daria
26/01/2023