Naučni članci / Scientific articles

Type of the Paper: Original scientific paper
Received: 24. 4. 2024.
Accepted: 24. 5. 2024.
DOI: https://doi.org/10.69899/edtech-se-en-240402007m
UDK: 004.415.53
004.78:004.85

Creating a Java application for monitoring and controlling an Arduino microcontroller board

Marija Mojsilovic 1*, Selver Pepic 1, Muzafer Saracevic 2 and Goran Miodragovic1

[1] Academy of Professional Studies Sumadija, College in Trstenik, Trstenik, Serbia, mmojsilovic@asss.edu.rs
[1] Academy of Professional Studies Sumadija, College in Trstenik, Trstenik, Serbia, spepic@asss.edu.rs
[2] University of Novi Pazar, Department of Computer Sciences, Novi Pazar, Serbia, muzafers@uninp.edu.rs
[1] Academy of Professional Studies Sumadija, College in Trstenik, Trstenik, Serbia, gmiodragovic@asss.edu.rs
* Correspondence: mmojsilovic@asss.edu.rs; +381 63 12 08 994

Abstract:
This research paper extensively examines the process of creating an application for monitoring and controlling an Arduino board, combining the Java programming language and the Arduino development environment. Computer science and informatics have become key disciplines in today's society, enabling rapid development through continuous digital evolution. The aim is to establish efficient communication between the Arduino board and a Java desktop application for controlling LED lights and displaying sensor information. The paper is structured into five parts that describe in detail the hardware used, explore the Java programming language, the connection of components, programming in different environments (NetBeans and Arduino IDE), and project demonstration. The implemented Java application enables LED control and sensor data display in a straightforward manner. The research results indicate successful achievement of the set objectives, along with exploration of application enhancement possibilities and consideration of challenges that may arise during implementation. The research highlights the importance of integrating different technologies to achieve efficient solutions in the field of computer science and electronics. This study represents a step towards better understanding the process of creating applications for monitoring and controlling electronic components, providing a foundation for further research and development in this area.

Keywords: Java application; Arduino; Graphic interface, NetBeans, LED diodes

1. Introduction

In today's digital age, computer science and informatics have not only become highly attractive but also pivotal fields. The concept of digital computer systems now implies a machine that can be programmed to perform various tasks. A large number of authors and programmers in the creator community rely on the Arduino platform to monitor device loads. They achieve this by using standalone power measurement plugs, wireless sensor networks, and dedicated microcontrollers, which provide different levels of connectivity [1-4]. This paper focuses on exploring the combination of two programming languages and two initially independent development environments: JAVA - the Java programming language and ARDUINO - an electronic component for programming with its development environment. The control interface is created using open-source Java scripting language and available Arduino open-source hardware [5]. The main goal of this research is to establish communication between the ARDUINO board and the graphical interface written in the JAVA programming language. The planned program should enable control of two LED lights (red and blue) via standard GUI elements [6], as well as gathering information from sensors on the ARDUINO board and displaying it in a text field within the desktop application written in the JAVA programming language. The paper is structured into five parts. The first part describes the hardware used, the ARDUINO board, and the programming development environment. The second part explores the JAVA programming language, emphasizing its object-oriented approach and other characteristics. The third part describes in detail the process of connecting components into a functional unit. The fourth part deals with programming in different development environments (NetBeans and Arduino IDE), while the fifth part demonstrates the project's operation through a visual representation.

2. Methodology

U In this research, we employ a case study as the primary research method. Specifically, the focus is on a particular model of the Arduino board and the development of an application for controlling LED lights and displaying sensor information as

part of that case study. This approach enables a detailed analysis of a specific technology application scenario in a real environment, providing deeper insights into the performance, functionality, and practical application of the Arduino system. The choice of a specific model of the Arduino board allows us to concentrate on the specific features and capabilities of that model, providing a clear foundation for application development.

Through the development of the application, the goal is to demonstrate how the Arduino board can be used to control LED lights and display sensor data, exploring possibilities for interaction with the user interface and the application of the Java programming language in that context. Through detailed examination and development of the case study, we expect to gain a deeper understanding of the practical aspects of implementing Arduino systems, as well as opportunities for integration with other technologies.

This case study will enable us to identify the advantages, challenges, and potential applications of this solution in real-world situations, providing valuable insights for further development and application of Arduino technology.

2.1. Arduino

Arduino is a platform that combines open-source hardware and software, providing the capability to develop various physical computing projects. The hardware foundation consists of an Atmel AVR processor with accompanying input-output elements and a microcontroller [6]. The development software, integrated on the board itself, consists of a standard compiler and development environment. Programming Arduino is done using the Wiring programming language and accompanying libraries, which essentially resemble C++ with specific adaptations.

2.1.1. Hardware

Arduino Uno is equipped with the Atmega328 chip, making it a versatile microcontroller board. With 14 digital input/output pins, of which 6 can be used as PWM outputs, and 6 analog inputs, it provides a wide range of possibilities for projects. Additionally, it features a USB connector, a power jack, an ICSP connector, and a reset button, facilitating interaction and programming. A linear voltage regulator of 5V and a 16MHz crystal oscillator ensure stable operation [7]. Arduino microcontrollers come with a pre-programmed bootloader, making it easy to transfer code to the chip's flash memory without the need for an additional programmer, which is a significant advantage compared to other microcontrollers. In Figure 1, the layout of the Arduino Uno board is shown.

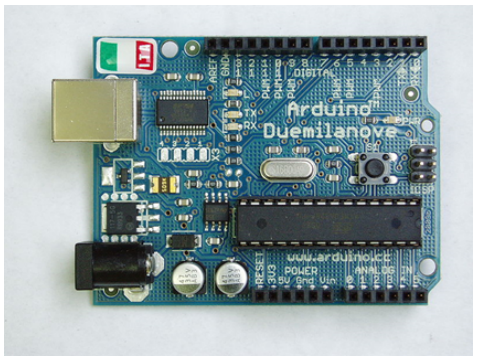


Figure 1. Arduino Uno board [8]

2.1.2. Software

Arduino development environment, written in the Java programming language, is tailored to simplify the programming process for individuals who are beginners in the world of software, including students, artists, and designers. This application contains a code editor with functionalities such as code highlighting, automatic bracket matching, automatic line indentation, and other conveniences that speed up software development.

The official programming tool is the Arduino IDE, available at arduino.cc. This environment is extremely simple and accessible, with a basic menu for key operations such as creating new projects, configuring ports, boards, and editing.

The Arduino development environment facilitates the process of uploading code to the board with a simple command, without the need for parameter adjustments. Additionally, it comes with a C/C++ library known as "Wiring," which simplifies basic input/output operations. Codes are written in the C/C++ programming language, where two key functions are defined at the beginning: SETUP() for initial settings, which is executed only once at the beginning of the program, and LOOP() which represents an infinite loop that continuously executes while the board is powered on.

2.2. Java

JAVA is a universal high-level programming language that fully supports the concept of object-oriented programming [9]. Originating in the early 1990s, JAVA is closely related to the C++ language, inheriting its roots from the C language. The first official version was released in 1995 [10].

The basic characteristic of Java is its object-oriented approach, where each class typically corresponds to one file. The source code is stored in .java files, while programs are translated into bytecode, the version of which is saved with the .class extension. To run a program, a Java virtual machine is required to compile bytecode, providing platform independence. This means that bytecode can be executed on any operating system that supports the Java virtual machine.

2.2.1. Development Environment

Two of the most commonly used development environments are Eclipse IDE and NetBeans IDE. NetBeans stands out as a comprehensive development environment that provides a wide range of capabilities, enabling efficient use of various programming languages such as C, C++, PHP, Python, and others.Basic features of the NetBeans environment:

- *Java Desktop applications*
- *Web applications*
- *Support for dynamic languages*
- *Modular organization*
- *Support for additional tools for configuration management*

In this work, code was combined with graphic design of the application using JFrameForm. After creating the application, the next step is to create a new package. Within that package, a JFrameForm is added, which also has a .java extension, but contains both design and code sections. This allows for easy addition of graphic elements by dragging components onto the window.

Basic elements, such as labels, buttons, lists, text fields, and similar, are usually located on the right side, within the "palette" tab. Programming these components is done by clicking on "Source", where the code is already generated automatically when adding components. Additional code can be easily added within the given classes to customize the application.

2.3. Connecting electronic components

As mentioned at the beginning of the project, the application needs to perform basic operations with LED diodes and read values from a light sensor.

For this project, the required components are: one Arduino UNO board, two LED diodes (red and blue), two 220 ohm resistors for the LED diodes, one light sensor, one 1K ohm resistor, as well as connectors or cables for connecting the components. These components will enable the construction of a system that uses Arduino to control LED diodes and measure light intensity using a light sensor.

Figure 2 provides a schematic representation of all the components needed for this project. This diagram describes in detail the layout and connection of each component, providing a clear visual insight into the system's structure.

The schematic was created and processed on the Circuits.io platform [11]. This platform allows users not only to add components and create schematics but also to simulate real-life situations. Through simulation, users can test and optimize their ideas before implementing them in a real environment, which often saves time and resources. This interactive capability provides users with additional security and reliability when working with electronic projects.

Figure 2. Connecting electronic components

Connecting the components, as can be seen in the diagram, is very simple. One end of each LED diode is connected to the ground, while the other end of the diode is connected via a 220 ohm resistor to digital output 12 or 13. As for the light sensor, one end is connected to a 5V power source, while the other end is connected via a 1 Kohm resistor to the ground. At the same time, this end is connected to analog input A0. This simple connection allows for precise light measurement through the sensor. Figure 3 shows the schematic of the component connections.

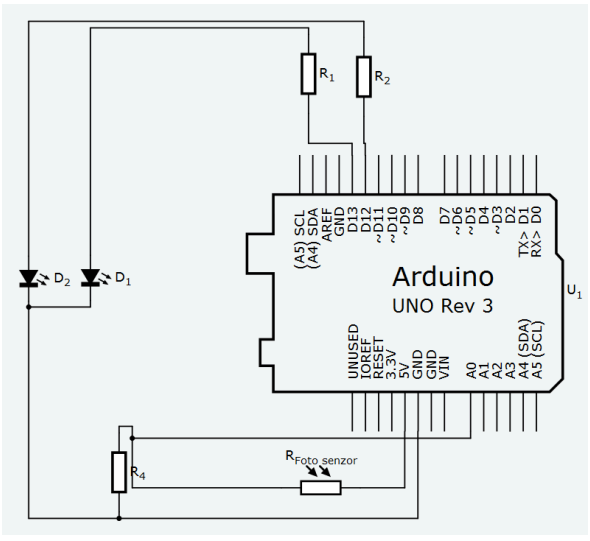


Figure 3. The schematic diagram of the component layout

3. Creating the application

After the hardware design and component connections phase, the next step is the implementation of the software part of the project. This phase involves two key segments: the first part relates to coding within the Arduino development environment (Arduino IDE), while the second part is realized through the use of NetBeans.

When it comes to programming, it is important to have a deep understanding of the hardware part of the project. This includes identifying and understanding input/output elements, as well as how they are connected to the software. This mutual connection enables efficient interaction between hardware and software, allowing them to function together in accordance with the project's set goals.

Creating the software part of the project requires programming skills and creativity to achieve the desired functionalities. Integrating this software with hardware components requires precision and careful testing to ensure that the system operates correctly and efficiently. This mutual collaboration between hardware and software is crucial for achieving success in project implementation and achieving desired goals.

3.1. Creating an Arduino application

The process of creating the application starts with defining the pins. Specifically, pins 13 and 12 are used as digital outputs, while analog input A1 is reserved for the variable "sensor". It is important to note that all variables are defined as type int. In the next step, within the "setup()" function, the initial pin configuration is performed using the pinMode function, which requires two parameters. The first parameter represents the pin name or the created variable, while the second parameter defines whether the pin is set as input or output. In this case, the LEDs are set as digital outputs, while the sensor is set as an input element.

The next part of the code deals with reading the sensor values and displaying them on the Serial monitor. Reading is achieved by creating a variable of type INT and storing information in that variable using the analogRead function. Then, the value is printed on the Serial monitor using the Serial.println function. To regulate the reading, a delay of 50ms is added at the end.

The final step in the first part of software development within the Arduino IDE involves setting up real-time communication commands between the computer and the Arduino via serial communication.

The program concept involves connecting to a JAVA application. The JAVA application will be responsible for transmitting information via the serial port to the Arduino, while the Arduino will interpret the received character and perform certain actions accordingly. For this reason, the following code is used.

First, a variable "option" of type CHAR is defined, which reads the value from the serial port. If the value "1" is received, the blue LED diode on pin 13 is turned on using the "digitalWrite" function. Similarly, if the character "0" is received, the same diode is turned off. The same principle is applied to the second red LED diode. This allows interactive control of the Arduino via serial communication. The code is shown in Figure 4.

```
int plavaDioda = 13;
int crvenaDioda = 12;
int senzor = A1;

void setup() {
  pinMode(senzor,input);
  pinMode(plavaDioda,output);
  pinMode(crvenaDioda,output);
  serial.begin(9600);
}

void loop() {
  int vrednostSenzora = analogRead(senzor);
  serial.println(vrednostSenzora);
  delay(50);
}
char opcija=serial.read();
if(opcija=='1')
  {
  digitalWrite(plavaDioda, HIGH);
  }
else if(opcija=='0')
  {
  digitalWrite(plavaDioda, LOW);
  }
else if(opcija=='2')
  {
  digitalWrite(crvenaDioda, HIGH);
  }
else if(opcija=='3')
  {
  digitalWrite(crvenaDioda, LOW);
  }
```

Figure 4. Connecting electronic components

3.2. Java application development

In the final stage of programming, it is crucial to develop a Java application that enables real-time communication with the Arduino board, allowing simultaneous issuance of commands. This application enables independent operation of the Arduino from the Arduino Integrated Development Environment (IDE), as the complete code is already present in the microcontroller. Therefore, only a Java application is required to manage the Arduino board, specifically its components, in this case, the LED diode.

Considering that the previous part of this work extensively explained how to create the project, this part focuses on the key parts of the code and the library used in the project. This allows for concise focus on the most important aspects of implementation, facilitating understanding and further work on the project.

The library used is called "jSerialComm-1.3.11.jar," which can be freely downloaded from the following link: fazecast.github.io/jSerialComm. As mentioned, this library provides independence and is based on an older version of the RxTx Java Communication API with increased performance and simpler implementation and usage.

The development of the application in the Java programming language followed a logical sequence of steps, starting with creating the graphical user interface. In this initial step, all necessary elements were added, as shown in Figure 5.

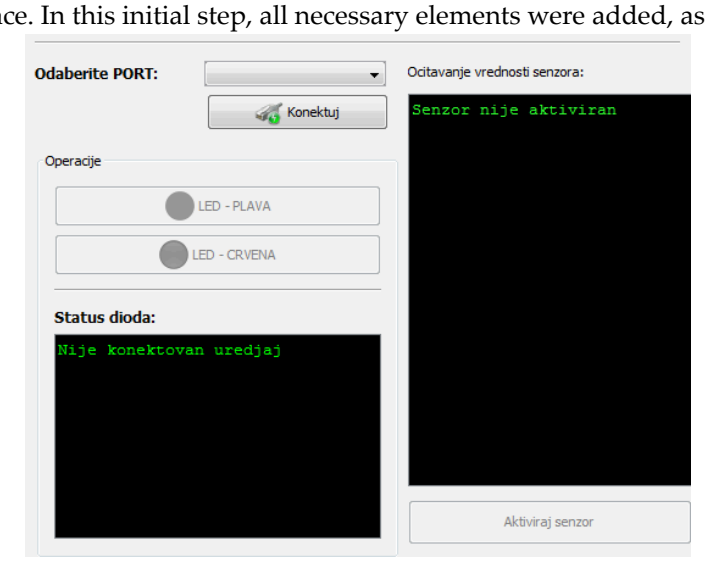


Figure 5. The layout of the arranged components before programming.

Within the main class, a public object of the SerialPort class from the previously mentioned library is created. This port will serve as the basis for further programming. Then, in the next step, a special method called "addPorts()" is added and placed within the constructor. This method is automatically called immediately after the initialization of the components.

Within this method, a TRY - CATCH block is used, which attempts to execute a certain part of the code, and in case of an error, it proceeds to the CATCH segment where the error is printed. The code itself is quite simple; an array of SerialPort objects is created with the names of available ports. Then, a FOR loop follows, which counts the available ports and adds them to the jComboBox element that displays available ports in a graphical manner. Names are added directly, where for each port, the system name is displayed using the "getSystemPortName" method.Sledeći segment koda odnosi se na funkcionalnost dugmeta za povezivanje na odabrani port.

This functionality depends on the text within the button. If the text on the button starts with "Konektuj", the previously defined "port" object is called, and the port name from the list is selected. This value is then converted to a String. If the port is open, the button state changes to "Disconnect", and other components adjust accordingly (buttons become visible, and further operations are enabled).

If the text on the button indicates "Disconnect", then the port is closed, and a message is displayed in the text field indicating that Arduino is not connected.Najvažniji deo koda odnosi se na upravljanje LED diodama.

By clicking on the button for the blue LED diode, the LED turns on or off, and this information needs to be sent to the Arduino board. According to the previously explained Arduino code, specific characters are sent to the Arduino. The initial state of the first LED is 0, indicating that the LED is off. If, after clicking the button, the state is read as 0, the value 1 is sent to the serial port, signaling the user's desire to turn on the LED, and the same is displayed in the text field. Conversely, if the state is read as 1, the value 0 is sent to the serial port, indicating that the LED is turning off. The code described above is shown in Figure 6.

```java
public SerialPort port;
    public GUI()
    {
        initComponents();
        dodavanjePortova();
    }

    public void dodavanjePortova()
    {
    try
       {
        SerialPort[] nazivPorta = SerialPort.getCommPorts();
         for(int i=0;i<nazivPorta.length;i++)
         jComboBox1.addItem(nazivPorta[i].getSystemPortName());
       }
    catch (exception e)
       {
        JOptionPane.showMessageDialog(rootPane, e);
       }
    }

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    if(jButton1.getText().equals("Konektuj"))
    {
        port=SerialPort.getCommPort(jComboBox1.getSelectedItem().toString());
        port.setComPortTimeouts(SerialPort.timeout_scanner,0,0);
        if(port.openPort())
        {
            jButton1.setText("Diskonektuj");
            jComboBox1.setEnable(false);
            jTextArea1.setText("Arduino");
            jButton2.setEnable(true);
            jButton3.setEnable(true);
            jButton.setEnable(true);
        }
    }
    else
    {
        port.closePort();
        jComboBox1.setEnable(true);
        jButton1.setText("Konektuj");
        jTextArea1.setText("Arduino nije priključen");
        jButton2.setEnable(false);
        jButton3.setEnable(false);
        jButton.setEnable(false);
    }
}
```

Figure 6. Java code

For example, if the user wants to turn on the LED, the system first checks the state of the LED. If the state of the LED is 0, indicating that it is off, the system enters the corresponding loop. After the user clicks the button, the system recognizes that the LED state is 0 and enters the first loop. Then, the system sends a signal to output 1, resulting in the LED being turned on.

This programming process is applied to the other LEDs as well, using different values: 2 for turning on and 3 for turning off. It is important to note that other characters can also be used according to the project's requirements.

It is also important to explain the process of reverse communication, i.e., reading values from the serial port. First, the port is opened for communication, and the timeout period is set. Then, an InputStream is used to directly access the input unit via port.getInputStream(). Within the TRY – CATCH block, a for loop is used to read the first 1000 values from the sensor. Each read value is immediately printed as a character using the in.read() method. After reading is complete, the InputStream is closed by calling the in.close() method, as shown in the code presented in Figure 7.

```java
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
    PrintWriter output = new PrintWriter(port.getOutputStream());
    if(led_1_stanje == 0)
    {
        output.print("1");
        output.flush();
        led_1_stanje=1;
        jTextArea1.setText(jTextArea1.getText()+"\n"+"PLAVA: UKLJUČENA");
    }
    else
    {
        output.print("0");
        output.flush();
        led_1_stanje=0;
        jTextArea1.setText(jTextArea1.getText()+"\n"+"PLAVA: ISKLJUČENA");
    }
port.openPort();
port.setComPortTimeouts(SerialPort.timeout_read_semi_blocking,100,0);
InputStream in = port.getInputStream();
try
{
    for(int j=0;j<1000;++j)
    {
        System.out.print((char)in.read());
        in.close();
    }
    jButton4.setText("Aktiviraj senzor");
}
catch(exception e)
{
    e.printStackTrace();
}
}
```

Figure 7. Java cod

4. Discussion

After the board is connected to Arduino, available ports appear in the jComboBox, such as COM 3, for example. When the user selects the desired port and clicks the "Connect" button, the other buttons that are available for use are activated. Clicking the button related to the blue LED turns it on, while the button below controls the red LED. In real-time, the user can monitor the status of the LEDs (whether they are on or off).

In parallel with these functions, it is possible to activate the sensor to monitor the room's lighting. By clicking the "Activate Sensor" button, the system reads values from the sensor for 1000 passes. Since the passes are very fast, the reading takes only 2-3 seconds. The physical layout of the component connections is shown in the image below, making it easier for users to follow and understand how the components are connected. The physical connection appears as shown in Figure 7.

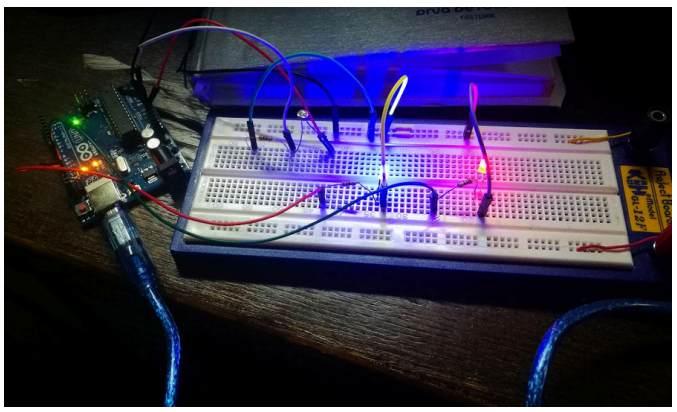


Figure 7. Arduino with connected components

5. Conclusions

The aim of this project was to provide a deeper understanding of programming fundamentals through the use of two different programming languages and their integration into a single functional unit, namely the Arduino platform. The project required knowledge of five basic elements: Arduino, Java, C/C++, communication between Arduino and Java, as well as serial ports.

Each of these components carries with it an extensive theoretical part and requires a lot of practical experiments, which can be found in available literature. Through this project, we gained knowledge in the field of electronics, developed logical thinking skills in programming, and learned how to find new creative approaches to problem-solving in the world of technology and informatics.

The final result of the project represents the successful integration of the Java programming language with the Arduino platform, based on C/C++ technology. The application functions quickly and reliably, opening the door to new possibilities for research and experimentation. By leveraging the advantages of Arduino, the Java programming language, and C/C++ technology, systems can be further improved in various fields, facilitating the operation of complex systems and providing opportunities for further development and innovation.

References

1. Shajahan, A. H., & Anand, A.: Data acquisition and control using Arduino-Android platform: Smart plug. In 2013 international conference on energy efficient technologies for sustainability, 2013, pp. 241-244, IEEE.
2. Anbya, M. F. B., Salehuddin, M., Hadisupadmo, S., & Leksono, E.: Wireless sensor network for single phase electricity monitoring system via Zigbee protocol. In 2012 IEEE Conference on Control, Systems & Industrial Informatics, 2012, pp. 261-266, IEEE.
3. Fransiska, R. W., Septia, E. M. P., Vessabhu, W. K., Frans, W., & Abednego, W.: Electrical power measurement using arduino uno microcontroller and labview. In 2013 3rd international conference on instrumentation, communications, information technology and biomedical engineering, 2013, ICICI-BME, pp. 226-229, IEEE.
4. Soliman, M., Abiodun, T., Hamouda, T., Zhou, J., & Lung, C. H.: Smart home: Integrating internet of things with web services and cloud computing. In 2013 IEEE 5th international conference on cloud computing technology and science, 2013, Vol. 2, pp. 317-320, IEEE.
5. Kvalsund, A. S., & Winkler, D.: Development of an Arduino-based, open-control interface for hardware in the loop applications, 2023. HardwareX, 16, e00488.
6. Potortì, F., La Rosa, D., & Palumbo, F.: Enerduino-pro: Smart meter led probe using Arduino. HardwareX, 2023, 15, e00461.
7. Mijailović, Đ., Karabašević, D., & Stanujkić, D.: Razvoj sistema za praćenje ambijentalnih faktora primenom arduino platforme. Trendovi u poslovanju, 2013, 2(12), 37-46.
8. Lattanzi, D., Pagliarini, M., Rebecchi, F., Frontalini, F., & Ambrogini, P.: Developing and testing an Arduino-based microcurrent stimulator to mimic marine electric pollution on benthos. Heliyon, 2024, 10(1).
9. Radosavljević, J.: Elektronske lekcije o osnovnim konceptima objektno orijentisanog programiranja u programskom jeziku Java, 2017.

10. Prečanica, Z., & Vidaković, M.: Razvoj skladištenja i pretrage digitalne biblioteke diplomskih radova upotrebom MongoDB, Elasticsearch I Blockchain TEHNOLOGIJA. Zbornik radova Fakulteta tehničkih nauka u Novom Sadu, 2019, 34(09), 1602-1605.
11. CircuitLab Inc., Online circuit simulator & schematic editor - CircuitLab, 2020-03, [Online]. Available: https://www.circuitlab.com/.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.