R E L E
(Rekayasa Elektrikal dan Energi) : Jurnal Teknik Elektro
Vol. 6, No. 2, Januari 2024, ISSN 2622 – 7002

DESIGN OF A 4 BAND COLOR CODE RESISTOR CALCULATOR APPLICATION BASED ON THE JAVA PROGRAMMING LANGUAGE

Rahmat Fauzi Siregar [1], Alkhowarizmi[2],Rohana[2]Affandi[4],Arya Rudi Nasution[5]
[1, 3]Department of Electrical Engineering, Faculty of Engineering, Universitas Muhammadiyah Sumatera Utara
[4,5]Department of Mecanical l Engineering, Faculty of Engineering, Universitas Muhammadiyah Sumatera Utara
[2] Department of Information Technology, Universitas Muhammadiyah Sumatera Utara
Jl. Kapten Muchtar Basri No.3, Glugur Darat II, Kec. Medan Tim., Kota Medan, Sumatera Utara, 20238
e-mail: rahmatfauzi @umsu.ac.id

Abstract— *This research aims to design and develop a 4 band resistor color code calculator application based on the Java programming language. This calculator is designed to help electronics technicians identify resistor values based on the color code found on 4 band resistors. This application provides an intuitive and user-friendly interface that allows users to enter the color of each band on a resistor and quickly displays the corresponding resistance value. Apart from that, this application can also provide the tolerance of a given resistor. The development method used in this research includes needs analysis, user interface design, implementation of resistance value calculation algorithms, and application testing. This application is implemented using the Java programming language and supported by the JavaFX graphic library to create an attractive interface. Test results show that this application can calculate resistance values accurately and is responsive to user input. With this application, it is hoped that users can easily and quickly identify the resistance value of 4 band resistors, increase efficiency in their work, and reduce the possibility of errors in deciphering resistor color codes.*

Keywords : *Java programming language, Resistor color code, Resistance value, User interface design, 4 band resistor, Resistor tolerance*

I. INTRODUCTION

In the realm of electronics and electrical engineering, resistors are essential components used to regulate the flow of current. To decipher a resistor's resistance value, engineers and hobbyists often refer to color codes displayed on the resistor's bands. Designing a reliable and user-friendly tool for calculating resistor values based on these color codes is crucial. This project introduces a 4 Band Color Code Resistor Calculator Application developed using the Java programming language. This application aims to simplify the process of determining resistor values by providing an intuitive and accurate solution, making it a valuable resource for individuals working with electronic components. In this document, we will explore the key features, functionalities, and design principles behind this Java-based application [1], [2].

Java is an object-oriented, general-purpose, high-level programming language created by Sun Microsystems, which is currently owned by Oracle Corporation. Java code may execute on any device that has a Java Virtual Machine (JVM) according to the design philosophy of "Write Once, Run Anywhere" (WORA). The Java Virtual Machine (JVM) runs the intermediate bytecode created by compiling the Java source code to accomplish this portability [3].

Any device that has a JVM that is compatible may run Java programmes. Java is hence incredibly adaptable and portable. Java adheres to the object-oriented programming paradigm, which promotes the modular and effective construction of code through the usage of classes and objects. Because of its syntax, Java is easily recognisable to programmers who have worked with C and C++. Java utilises an automated garbage collector to manage memory, which lowers the risk of memory leaks and makes memory management easier for developers. Java comes with built-in security tools that assist shield systems from dangerous code, such a security manager and bytecode verification. The broad standard library that comes with Java offers a wide range of capabilities, making it simpler for developers to do typical tasks without having to start from scratch with significant code. Because multithreading is integrated into Java, programmers may design concurrent, scalable, and responsive programmes. Because of Java's sizable and vibrant developer community, a plethora of tools, libraries, and frameworks are accessible for Java development. Java is used extensively in many different fields, such as corporate applications, scientific and research applications, mobile app development (android applications are written in Java or Kotlin), web development (using frameworks like Spring and JavaServer Faces), and more [4], [5], [6].

People can interact with computers and software programmes more easily and intuitively when they employ graphical user interfaces (GUIs), especially if they are unfamiliar with command-line interfaces. An interface known as a Graphical User Interface (GUI) replaces text-based user interfaces with graphical components like windows, buttons, and icons to enable

ISSN 2622 – 7002 (online), Hal 82 – 87, DOI: https://doi.org/10.30596/rele.v1i1. 17251

users to interact with software or electronic devices. GUIs employ visual representations of data and user activities to facilitate user comprehension and system interaction. Java Swing or JavaFX libraries are commonly used in Java programming to construct graphical user interfaces (GUIs). These libraries give developers access to a collection of classes and elements that let them create dynamic, eye-catching user interfaces for their applications [7], [8].

Therefore, we developed "Design of a 4 Band Color Code Resistor Calculator Application Based on the Java Programming Language" application to make it easier for users to calculate resistor values quickly.

II. RELATED RESEARCH

Komlen Lalović and M. Bogdanoski in their research is to present a novel Java GUI based software application for a comparative analysis of fingerprint and iris biometrics. They are realized in Java Programming language. In the GUI framework named swing while the rest of the paper shows in detail the advantages and disadvantages of both systems and gives scientific data on when fingerprint and iris recognition can be used to enable top level security. The main method is a well known comparative analysis. The results were obtained for both fingerprint and iris biometrics, showing the difference between the two. Different types of biometrics, based on body parts formed at different age, are given as well as the comparison of their security levels [9].

A. Kolya et al, presents the design and implementation of control strategy for both the speed and direction of a direct current (DC) motor using Android-based application in smart phone. The Raspberry Pi 3 with a motor driver controller has been used to implement the control action via Python-based user-defined programming. The Android application has been developed using Android Developer Tools (ADT) in Java platform. The Android apps work like a client and communicates with Raspberry Pi through wi-fi connectivity. Finally, a small graphical user interface (GUI) has been created in Python in order to interface and control the motor with buttons in GUI. The advantages of GUI are that it is attractive, user friendly, and even a layman can work with the application developed in GUI [10].

III. RESEARCH METHODS

A. Software and Hardware

In designing an application, software and hardware components are needed that are in accordance with the design system being created. The software used in designing this application are NetBeans IDE version 8.2, Java Development Kit (JDK) version 1.8 and Java Runtime Environment (JRE) version 1.8. All software components run on the Windows 10 operating system with 64-bit architecture. All software components are supported by hardware which has the following specifications: Central Processing Unit (CPU) Intel Core I7 7500U, Graphics Processing Unit (GPU) Nvidia Geforce 930MX.

B. Application Design

The user interface of the application is designed using JavaFX using the waterfall model method. The "Waterfall" model design method is one of the traditional approaches in software development that follows a series of linear and sequential stages. This model is known because each stage must be completed before starting the next stage, similar to a waterfall that flows from one stage to the next without going back. The block diagram of the design stages of the 4 band color coded resistor calculator application is shown in figure 1. The following are the main stages in the Waterfall design model including [11]:

1. Analysis: This stage involves a deep understanding of the needs of the software to be developed.
2. Design: Once the analysis is complete, the software designer creates a system design that includes architecture, data structure, and interface design.
3. Implementation: This stage involves coding or creating software based on the plans that have been created at the design stage.
4. Testing: The software that has been implemented is tested to ensure that it works according to the pre-defined requirements.
5. Errors and bugs are discovered and fixed during this stage.
6. Delivery: After successful testing, the software is handed over to the user or customer.
7. Maintenance: After the software is used by users,

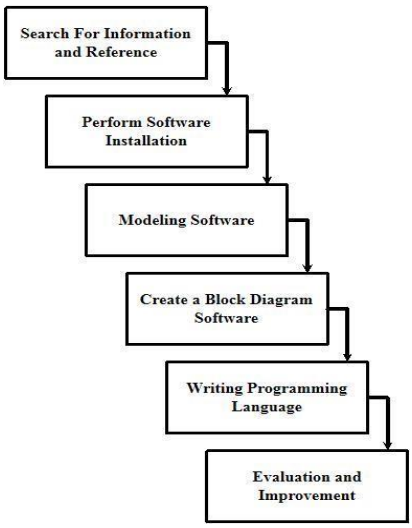

Figure 1. Block diagram design of a 4 band color code resistor calculator application

C. User Interface Design

The user interface is designed using Java Swing. Java Swing is a collection of classes used to develop GUI (Graphical User Interface) based applications. Apart from that, Java Swing can also be interpreted as one of the many solutions for developing GUI based applications [12]. This application was built using one JPanel component in the Swing Containers class, six JButtons, four JComboBoxes, eighteen Jlabels and two JTextFields component in the Swing Controls class. Writing the component program used in the user interface is shown in Figure 1.

```
// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton Band1;
private javax.swing.JButton Band2;
private javax.swing.JButton Band3;
private javax.swing.JButton Band4;
private javax.swing.JButton btnCal;
private javax.swing.JButton btnExit;
private javax.swing.JComboBox<String> cmbColor;
private javax.swing.JComboBox<String> cmbMulti;
private javax.swing.JComboBox<String> cmbNumber;
private javax.swing.JComboBox<String> cmbTolerance;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JLabel lblColor;
private javax.swing.JLabel lblMulti;
private javax.swing.JLabel lblNumber;
private javax.swing.JLabel lblTol;
private javax.swing.JLabel lblTolerance;
private javax.swing.JTextField txtHasil1;
private javax.swing.JTextField txtHasil2;
// End of variables declaration//GEN-END:variables
```

Figure 2. Writing component programs used in the user interface

After the components are written in the program, proceed with determining the background color of the user interface. The background color of the user interface used is light blue with RGB color values (0, 185, 233) as shown in Figure 2.

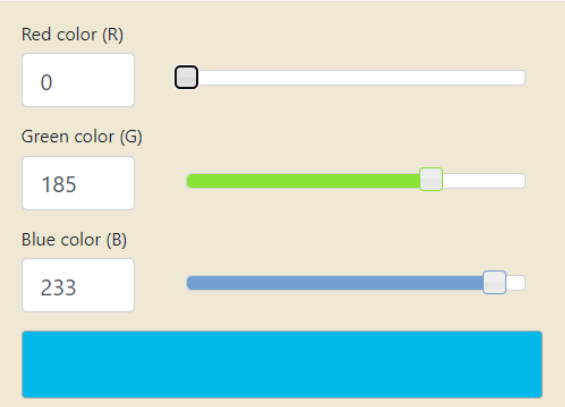

Figure 2. The RGB color values use in the background color of the user interface

D. Algorithm

A resistor with four color bands is typically used to indicate its resistance value. The color bands are used to represent numbers according to a standardized color code. The 4 - band resistor is shown in figure 3.

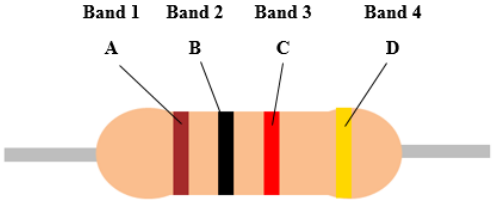

Figure 3. The *4-band resistor*

The formula to calculate the resistance value of a 4 - band resistor is as follows:

$$R = ((A \times 10 + B) \times 10^{C} \pm D) \quad (1)$$

Where, R is Resistance (ohms), A is the first band represents the first digit. B is the second band represents the second digit. C is the third band represents the multiplier, which indicates the power of 10 by which you multiply the two digits. D is the fourth band represents the tolerance, which specifies how much the actual resistance can deviate from the indicated value.

In the first digit, the variable JComboBox is replaced by cmbColor with the action event performed. Actionperformed events are the event function methods used when an Action event occurs. The class function getSelectedIndex() is applied to the first digit. The getSelectedIndex() class functions to return the selected index value. The value of the first digit changes based on the color selected. The algorithm for the first digit is shown in figure 4.

```
private void cmbColorActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(cmbColor.getSelectedIndex()==0){
        lblColor.setText("0");
        Band1.setBackground(BLACK);
    }
    else if(cmbColor.getSelectedIndex()==1){
        lblColor.setText("1");
        Band1.setBackground(BROWN);
    }
    else if(cmbColor.getSelectedIndex()==2){
        lblColor.setText("2");
        Band1.setBackground(RED);
    }
    else if(cmbColor.getSelectedIndex()==3){
        lblColor.setText("3");
        Band1.setBackground(ORANGE);
    }
```

Figure 4. The algorithm for the first digit

In the second digit, the variable JComboBox is replaced by cmbNumber with the action event performed. ActionPerformed events are the event function methods used when an Action event occurs. The class function getSelectedIndex() is applied to the second digit. The getSelectedIndex() class functions to return the selected index value. The value of the second digit changes based on the color selected. The algorithm for the second digit is shown in figure 5.

ISSN 2622 – 7002 (online), Hal 82 – 87, DOI: https://doi.org/10.30596/rele.v1i1. 17251

```
private void cmbNumberActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(cmbNumber.getSelectedIndex()==0){
        lblNumber.setText("0");
        Band2.setBackground(BLACK);
    }
    else if(cmbNumber.getSelectedIndex()==1){
        lblNumber.setText("1");
        Band2.setBackground(BROWN);
    }
    else if(cmbNumber.getSelectedIndex()==2){
        lblNumber.setText("2");
        Band2.setBackground(RED);
    }
    else if(cmbNumber.getSelectedIndex()==3){
        lblNumber.setText("3");
        Band2.setBackground(ORANGE);
    }
```

Figure 5. The algorithm for the second digit

In the multiplier, the variable JComboBox is replaced by cmbMulti with the action event performed. Actionperformed events are the event function methods used when an Action event occurs. The class function getSelectedIndex() is applied to the multiplier. The getSelectedIndex() class functions to return the selected index value. The multiplier value changes based on the selected color with a value of 10 raised to the power of the multiplier color value. The algorithm for the multiplier is shown in figure 6.

```
private void cmbMultiActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(cmbMulti.getSelectedIndex()==0){
        lblMulti.setText("1");
        Band3.setBackground(BLACK);
    }
    else if(cmbMulti.getSelectedIndex()==1){
        lblMulti.setText("10");
        Band3.setBackground(BROWN);
    }
    else if(cmbMulti.getSelectedIndex()==2){
        lblMulti.setText("100");
        Band3.setBackground(RED);
    }
    else if(cmbMulti.getSelectedIndex()==3){
        lblMulti.setText("1000");
        Band3.setBackground(ORANGE);
    }
    else if(cmbMulti.getSelectedIndex()==4){
        lblMulti.setText("10000");
        Band3.setBackground(YELLOW);
    }
```

Figure 6. The algorithm for the multiplier

In the tolerance, the variable JComboBox is replaced by cmbTolerance with the action event performed. Actionperformed events are the event function methods used when an Action event occurs. The class function getSelectedIndex() is applied to the tolerance. The getSelectedIndex() class functions to return the selected index value. The tolerance value changes based on the color selected. The use of this tolerance serves to determine the lower and upper limits of the tolerance value on the resistor. The tolerance value for resistors is generally 0% to 20%. The algorithm for the tolerance is shown in figure 7.

```
private void cmbToleranceActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(cmbTolerance.getSelectedIndex()==0){
        String coklat ="\u00B1 1%";
        String toUpperCase = coklat.toUpperCase();
        lblTolerance.setText(coklat);
        lblTol.setText("0.01");
        Band4.setBackground(BROWN);
    }
    else if(cmbTolerance.getSelectedIndex()==1){
        String merah ="\u00B1 2%";
        String toUpperCase = merah.toUpperCase();
        lblTolerance.setText(merah);
        lblTol.setText("0.02");
        Band4.setBackground(RED);
    }
    else if(cmbTolerance.getSelectedIndex()==2){
        String hijau ="\u00B1 0.5%";
        String toUpperCase = hijau.toUpperCase();
        lblTolerance.setText(hijau);
        lblTol.setText("0.005");
        Band4.setBackground(GREEN);
    }
```

Figure 7. The algorithm for the tolerance

After all the algorithms have been created for each band, proceed with determining the algorithm to calculate the results of the resistor resistance value based on equation 1. The calculation results are displayed in the JTextFields component with the variable names txtHasil1 and txtHasil2. The algorithm for calculating resistors with Ohm units is displayed in txtHasil1 and txtHasil2 are shown in figure 8.

```
if(hitung1 >= 0 && hitung1 <= 999){
    //Hasil 1
    String ohm = (String.format(((z[2]) * time) + simboll));
    String hasil1 = ohm + (" ") + lblTolerance.getText();
    txtHasil1.setText(hasil1);

    //Hasil2
    String hasilminus = (String.format(minus + simboll));
    String hasilplus =  (String.format(plus + simboll));
    String hasil2 = hasilminus + (" ") + ("to") + (" ") + hasilplus;
    txtHasil2.setText(hasil2);

}
```

Figure 8. The algorithm for calculating resistors with Ohm units is displayed in txtHasil1 and txtHasil2

The algorithm for calculating resistors with Kohm (Kilo Ohm) units is displayed in txtHasil1 and txtHasil2 are shown in figure 9.

```
else if(hitung1 >= 1000 && hitung1 <= 999999){
    //Hasil1
    String ohm = (String.format((((z[2]) * time) / 1000) + simbol2));
    String hasil1 = ohm + (" ") + lblTolerance.getText();
    txtHasil1.setText(hasil1);

    //Hasil2
    String hasilminus = (String.format((minus / 1000) + simbol2));
    String hasilplus =  (String.format((plus / 1000) + simbol2));
    String hasil2 = hasilminus + (" ") + ("to") + (" ") + hasilplus;
    txtHasil2.setText(hasil2);
}
```

Figure 9. The algorithm for calculating resistors with KOhm units is displayed in txtHasil1 and txtHasil2

The algorithm for calculating resistors with Mohm (Mega Ohm) units is displayed in txtHasil1 and txtHasil2 are shown in figure 10.

```
else{
    //Hasil1
    String ohm = (String.format((((z[2]) * time) / 1000000) + simbol3));
    String hasil1 = ohm + (" ") + lblTolerance.getText();
    txtHasil1.setText(hasil1);

    //Hasil2
    String hasilminus = (String.format((minus / 1000000) + simbol3));
    String hasilplus =  (String.format((plus / 1000000) + simbol3));
    String hasil2 = hasilminus + (" ") + ("to") + (" ") + hasilplus;
    txtHasil2.setText(hasil2);
}
```

Figure 10. The algorithm for calculating resistors with Mohm units is displayed in txtHasil1 and txtHasil2

IV. Experimental Results and Discussion

Based on the methods that have been carried out and implemented into the application, the application runs well and smoothly. The application installer was created using the Advanced Installer Architect 20.5 application. Figure 11 below shows the 4 band color code resistor calculator application installer with the .exe extension.

Figure 11. Display of the user interface of the 4 band color code resistor calculator application

When the application is run, the application user interface has several functions according to the method applied. Figure 12 below shows the user interface of the 4 band color code resistor calculator application.

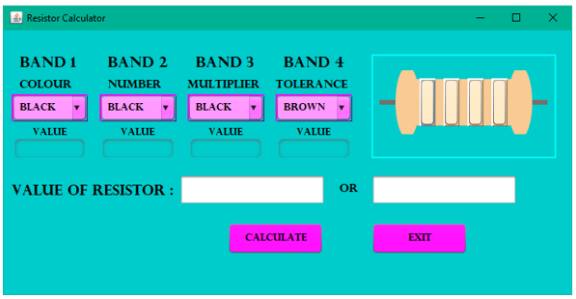

Figure 12. Display of the user interface of the 4 band color code resistor calculator application

After the main display of the application user interface is visible, then proceed with testing to calculate the resistor value based on the resistor color band as shown in table 1 below.

Table 1. 4 Band Resistor Color Code

Resistor Color Code				Value
Band 1	**Band 2**	**Band 3**	**Band 4**	
Red	Green	Blue	Gold	25 MOhm ± 5%
Yellow	Blue	Orange	Brown	46 KOhm ± 1%
Brown	Black	Red	Silver	1 KOhm ± 10%
Green	Blue	Brown	Red	560 Ohm ± 2%

The display and results of the red, green, blue, gold resistor color calculations show that the application can calculate the resistor color values from 23.75 to 26.25 MOhm with precision. Figure 13 shows the calculation of resistor values in red, green, blue, gold in the application.

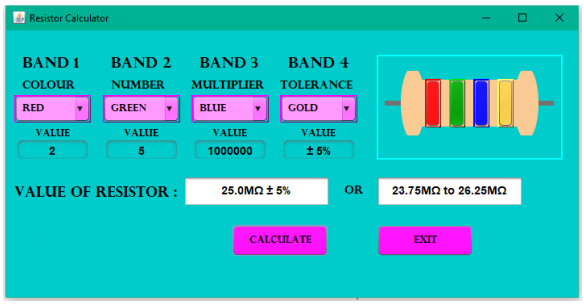

Figure 13. Calculation of resistor values in red, green, blue, gold in the application

The display and results of the yellow, blue, orange, brown resistor color calculations show that the application can calculate resistor color values from 45.54 to 46.46 KOhm with precision. Figure 14 shows the calculation of resistor values in yellow, blue, orange, brown in the application.

Figure 14. Calculation of resistor values in yellow, blue, orange, brown in the application

The display and results of the brown, black, red, silver resistor color calculations show that the application can calculate resistor color values from 0.9 to 1.1 KOhm with precision. Figure 15 shows the calculation of resistor values in brown, black, red, silver in the application.

RELE (Rekayasa Elektrikal dan Energi) : Jurnal Teknik Elektro, Vol. 6, No. 2, Januari 2024
ISSN 2622 – 7002 (online), Hal 82 – 87, DOI: https://doi.org/10.30596/rele.v1i1. 17251

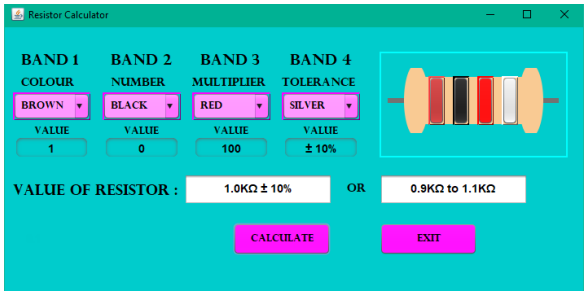

Figure 15. Calculation of resistor values in brown, black, red, silver in the application

The display and results of the green, blue, brown, red color resistor calculations show that the application can calculate resistor color values from 548.8 to 571.2 Ohm with precision. Figure 16 shows the calculation of resistor values in green, blue, brown, red in the application.

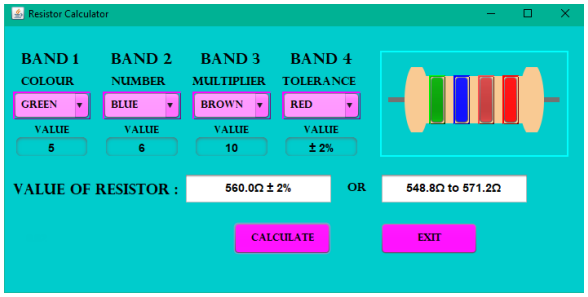

Figure 16. Calculation of resistor values in green, blue, brown, red in the application

The application is also used to test the value of each resistor color. Based on the test results, all resistor colors can be tested with correct and precise calculated values. This application was tested on windows 10 with 64-bit architecture.

V. CONCLUSION

The information supplied about the application seems to indicate that it is functioning and well-designed. It has been tested on a 64-bit Windows 10 machine. Advanced Installer Architect 20.5 was used to design the programme installer, and it has a user interface with several features. As shown in Table 1, the main goal of the programme is to compute resistor values depending on their colour bands. Furthermore, the application is capable of testing the value of each resistor color accurately. This suggests that the application can be relied upon for resistor color calculations and testing, making it a useful tool for professionals or enthusiasts working with electronic components. This application can still be updated to meet the needs of the 5 band and 6 band resistor color calculator by updating the existing program.

REFERENCES

[1] V. Kotov, A. Palagushkin, and F. A. Yudkin, "Metaphorical Modeling of Resistor Elements," *Stud. Comput. Intell.*, 2019.

[2] F. I. Pasaribu, P. Harahap, and M. Adam, "The Design of Energy Storage Circuits for Efficient Use of Electric Power on Computer Devices," *Budapest Int. Res. Exact Sci. (BirEx)Journal*, vol. 2, no. 3, pp. 368–375, 2020.

[3] P. H. Guzzi, "Computing Languages for Bioinformatics: Java," in *Encyclopedia of Bioinformatics and Computational Biology*, 2019.

[4] I. Kostaras, C. Drabo, J. Juneau, S. Reimers, M. Schröder, and G. Wielenga, "What Is Apache NetBeans," 2019.

[5] H. Liu, G. Lin, Z. Zeng, X. Li, J. Wu, and M. Ling, "Technology and Application of Electric Energy Meter with Embedded JVM," *2022 9th Int. Forum Electr. Eng. Autom.*, pp. 641–647, 2022.

[6] D. Bhattacharya, K. B. Kent, E. E. Aubanel, D. Heidinga, P. Shipton, and A. Micic, "Improving the performance of JVM startup using the shared class cache," *2017 IEEE Pacific Rim Conf. Commun. Comput. Signal Process.*, pp. 1–6, 2017.

[7] F. Belli, M. Beyazit, C. J. Budnik, and T. Tuglular, "Chapter Five - Advances in Model-Based Testing of Graphical User Interfaces," *Adv. Comput.*, vol. 107, pp. 219–280, 2017.

[8] M. C. Jayasinghe, "Graphical User Interface for the Supervisory Motion Control of BlueROV1.1," *2018 2nd Int. Conf. Electr. Eng.*, pp. 162–167, 2018.

[9] K. Lalović and M. Bogdanoski, "Java GUI application for comparing the levels of biometric security: Fingerprint vs. Iris," *Vojnoteh. Glas.*, 2021.

[10] A. K. Kolya, D. Mondal, A. Ghosh, and S. Basu, "Direction and Speed Control of DC Motor Using Raspberry PI and Python-Based GUI," *Int. J. Hyperconnectivity Internet Things*, 2021.

[11] R. Fauzi Siregar, R. Syahputra, and M. Yusvin Mustar, "Human-Robot Interaction Based GUI," *J. Electr. Technol. UMY*, vol. 1, no. 1, pp. 10–19, 2017.

[12] Al-Khowarizmi *et al.*, "The Effect of Indonesian and Hokkien Mobile Learning Application Models," *J. Comput. Sci. Inf. Technol. Telecommun. Eng.*, vol. 1, no. 1, pp. 1–7, 2020.