ISSN (e): 2250 – 3005 || Volume, 16 || Issue, 5 || May – 2026 ||
International Journal of Computational Engineering Research (IJCER)

Controlling DC motor via Arduino library in Matlab

Trung Ngo Kien
Faculty of Mechanical and Electrical Engineering, Hanoi Industry and Trade University, Hanoi, Viet Nam
Corresponding Author mail: trungnk@hict.edu.vn

ABSTRACT

Currently, servo motors are widely used in production lines due to their high accuracy and easy speed adjustment capabilities. In scientific research and training, directly connecting and controlling DC motors via software is a matter of concern. This paper introduces the Arduino library in the MATLAB environment and its application in designing a controller for servo motors.

KEYWORDS: DC Servo, controller, arduino, matlab, speed.

Date of Submission: 05-05-2026 Date of acceptance: 16-05-2026

I. INTRODUCTION

In recent years, the field of mechatronics has undergone rapid development to meet the growing demand for mechatronic systems. Microcontrollers and PLCs have been widely used in industrial production systems to improve automation and control efficiency. Research on control systems, along with modeling and simulation work on servo motor operation, is an effective method for developing controllers. Motion control for DC motors is implemented through encoder-based feedback. The position signal from the encoder is continuously compared with the reference signal input to the controller to minimize tracking error and ensure accurate output. This paper presents the Arduino library integrated in the MATLAB environment and its application in the design and development of a servo-based controller

II. ARDUINO AND MATLAB LIBRARY

2.1 Overview

The Arduino board is a microcontroller-based platform designed for programming and interfacing with hardware devices such as sensors, motors, and other peripherals. Arduino has been widely adopted as the processing unit in various applications, ranging from simple to complex systems, due to its advantages, including ease of use, a simple programming language, low cost, and its open-source nature in both hardware and software. Arduino has evolved through a collaborative development model that allows users worldwide to build, enhance, and contribute to the platform. Microcontrollers on Arduino boards can be programmed using the C++ language, with code compiled via the Arduino IDE (Integrated Development Environment) and associated compilers into binary machine code. With Arduino, control algorithms can be easily tested and validated in real-world applications, particularly through its integration with specialized software such as MATLAB, LabVIEW, and others.

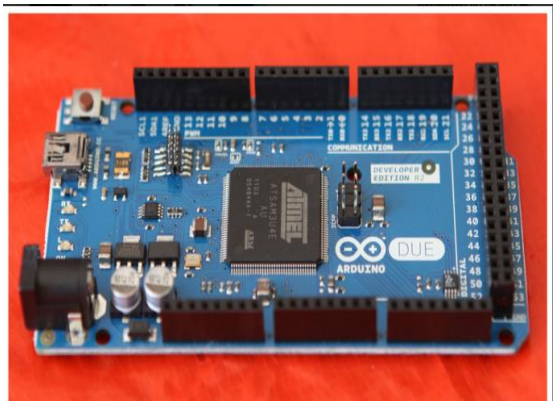


Fig.2.1. Arduino Board

The Arduino IDE (Integrated Development Environment) is the official open-source software used to facilitate code writing and compilation for Arduino boards. The main program, commonly referred to as a sketch, is developed within the IDE and compiled into a HEX file, which is then uploaded to the microcontroller on the board. The IDE environment primarily consists of two main components: a code editor and a compiler. The former is used for writing the required code, while the latter is responsible for compiling and uploading the code to the Arduino board. This environment supports both C and C++ programming languages.

Fig.2.2. Arduino IDE startup interface

MATLAB, short for Matrix Laboratory, is a software platform developed by MathWorks for programming, numerical computation, and high-level data visualization. Currently, MATLAB provides thousands of built-in commands and utility functions. In addition to its core functions, MATLAB offers specialized commands and functions through various toolboxes, which extend its capabilities to address domain-specific problems. These toolboxes are highly useful and widely utilized in applications such as digital signal processing, image processing, power electronics, fuzzy logic, and others.

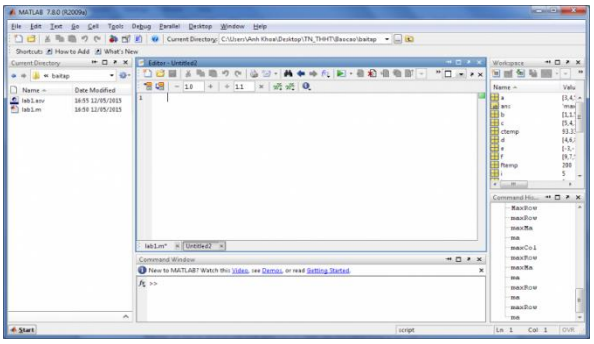


Fig.2.3. MATLAB interface

Simulink is a toolbox within MATLAB used for modeling, simulation, and analysis of dynamic systems. It provides a comprehensive library of functional blocks, including input/output blocks, signal sources, and linear and nonlinear components. Therefore, it enables users to simulate, analyze, and modify system models at any stage of development

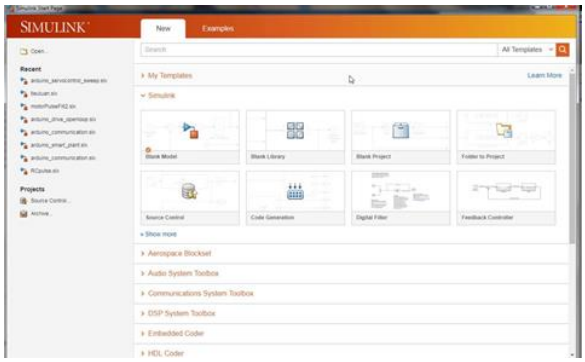


Fig.2.4. Simulink interface

2.2 Installation of Arduino Library in MATLAB/Simulink

Step 1: Launch MATLAB.

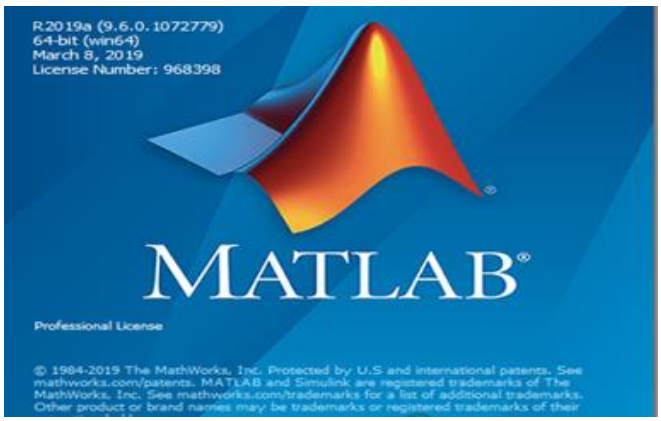


Fig.2.5. MATLAB startup interface

Step 2: Click the Add-Ons tab in the Home section.

Fig.2.6. Select the Add-Ons option

Step 3: **Select** Get Hardware Support Packages**.**

Fig.2.7. Looking For Hardware Packages

Step 4: Open the support package installer and select installation from the Internet.

Fig.2.8. Installing the support package

Step 5: A new window appears displaying all available MATLAB support packages. Locate the Arduino package in the list and click on it to proceed with the installation. A login window will then prompt for MathWorks account credentials. If an account is not available, please create one and continue. Note that both support packages, Simulink and MATLAB, should be installed.

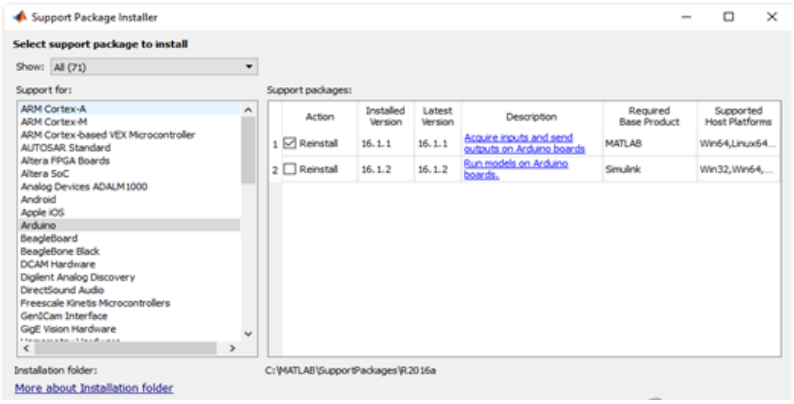


Fig.2.9. Arduino support package

Verify the installed hardware support packages: After completing the installation, it is necessary to check whether the packages are available in MATLAB by entering the following command in the Command Window:

a=arduino ()

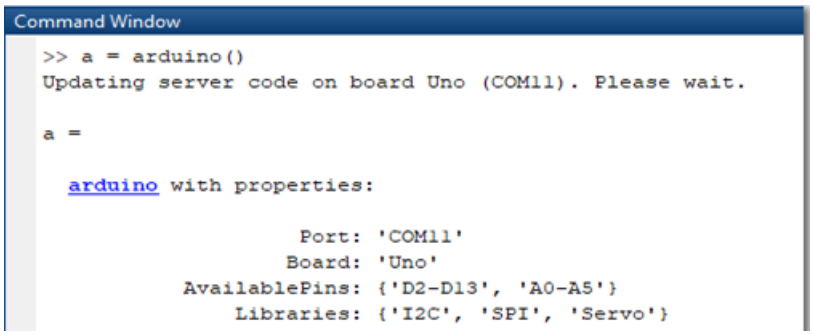


Fig.2.10. Verification of installation

If more than one microcontroller is connected to the PC, it is necessary to verify the port number. Navigate to the Control Panel, then to **Devices and Printers**, and check the assigned COM port number.

Fig.2.11. COM port number

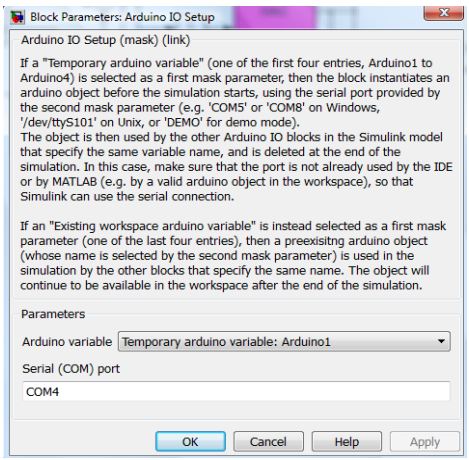
When the Arduino is connected to COM11, the port number must be updated in the code before compilation.

a = arduino (com11, uno)

```
>> a = arduino('com11', 'uno')

a =

  arduino with properties:

                Port: 'COM11'
               Board: 'Uno'
```

Fig.2.12. COM port configuration in the code

2.3 Functional Blocks



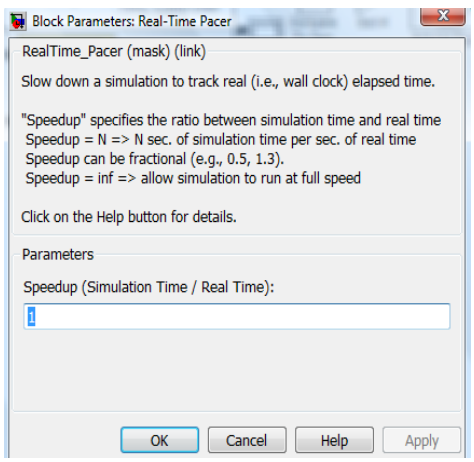


Fig.2.13.Arduino IO setup

Fig.2.14. Real-Time Pacer

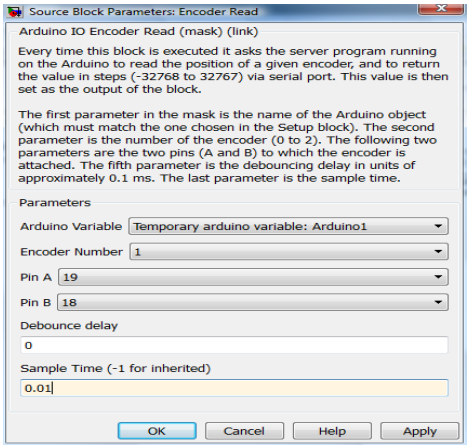


Fig.2.15.Encoder read

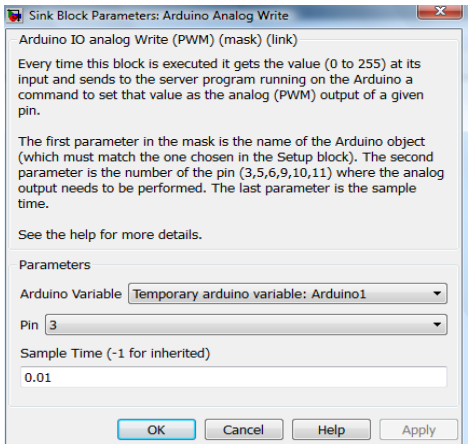


Fig.2.16. Arduino analog write

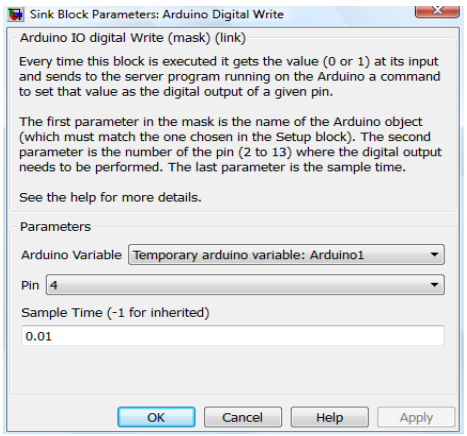


Fig.2.17.Arduino digital write

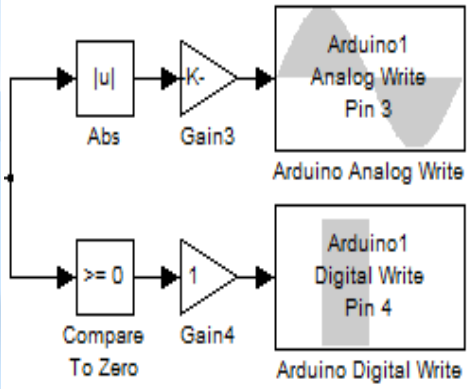


Fig.2.18. Reversing control

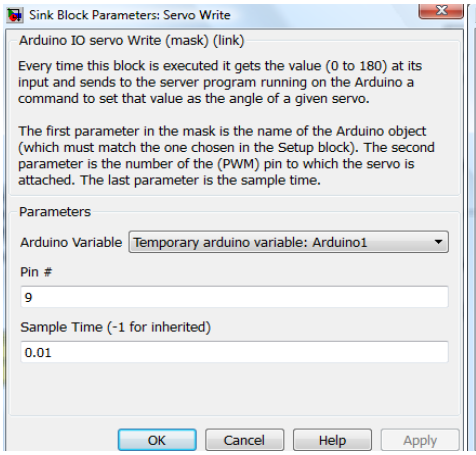


Fig.2.19. Servo write

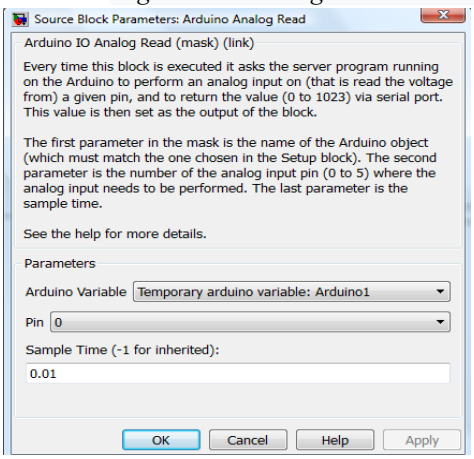


Fig.2.20. Arduino analog read

III. CONTROLLER DESIGN

3.1. DC motor

A servo motor is a key element in modern motion control systems, delivering the mechanical force required to operate machinery with high precision. Available in a wide range of shapes and sizes, servo motors are deployed across diverse industrial and technical applications. Their design variations are specifically engineered to meet demanding requirements for accuracy, high speed, high-frequency response, and precise control of position and velocity. To function effectively, a servo motor operates in conjunction with several essential components: Dedicated motion controller: Executes control programs to ensure the system meets the technical requirements of a given application.

- Motor drive (servo drive): An electronic device that supplies power to the motor in the correct manner and at the appropriate time.
- Rotary encoder: Provides real-time feedback on motor performance.

Rotary encoders typically rely on optical principles. A transparent glass disc, etched with evenly spaced radial markings, is mounted on the motor shaft and rotates with the rotor. This configuration enables accurate detection of rotational position and speed. Servo motors are broadly classified into two main types: **AC servo motors** and **DC servo motors**.

AC servo motors are capable of handling high current loads and are therefore widely used in industrial machinery. In contrast

DC servo motors are not designed for high-current applications and are generally better suited for smaller-scale uses. DC servo motors are further divided into **brushed** and **brushless** types.

A typical servo motor consists of three primary components:

- Rotor: A permanent magnet that generates a strong magnetic field.
- Stator: Contains coils wound around a core, energized to produce the force required to rotate the rotor.
- Encoder: Mounted at the rear of the motor to provide precise feedback on speed and position.

In operation, the rotor—composed of a permanent magnet—interacts with stator windings that are energized in a controlled sequence. When the timing and current supplied to these windings are accurate, the rotor's motion is governed by frequency, phase, polarity, and current flowing through the stator coils. Servo motors operate within **closed-loop feedback systems**. They receive pulse-width modulation (PWM) signals from the controller and are continuously monitored via the encoder. During operation, the motor's speed and position are fed back to the control circuit. If any disturbance causes deviation from the desired performance, the feedback mechanism signals the controller, which compares the actual and target values and makes real-time adjustments. This ensures the motor maintains optimal accuracy in both speed and positioning.

The servo drive typically employs an array of power transistors known as **insulated-gate bipolar transistors (IGBTs)** to regulate the energy supplied to the motor. Due to their fast-switching capability and ability to handle high currents, IGBTs are well-suited for this purpose. Controlled by electronic circuits, they generate specific voltage, current, frequency, polarity, and phase characteristics required by the servo motor. As a result, each servo drive is usually designed to work with a specific type of servo motor. Although the drive is powered by direct current (DC), its output is effectively an alternating waveform, enabling smooth control of speed, acceleration, and torque.

3.2. Control structure design

- The control objective in this system is to achieve precise position control through a servo motor. In the presence of any deviation, the position signal from the encoder is fed back and compared with the reference input of the system. Based on this comparison, the controller generates appropriate control signals to drive the motor such that the output closely tracks the reference signal. The control structure of the system is illustrated in Fig. 3.1.

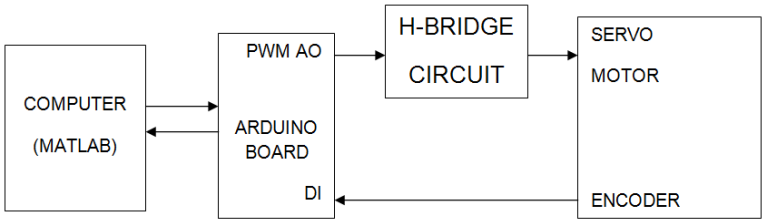


Fig. 3.1. Control structure of the system.

- - The controller employs a PID control algorithm to regulate the motor speed according to the reference value (Fig. 3.2), corresponding to the desired needle motion.

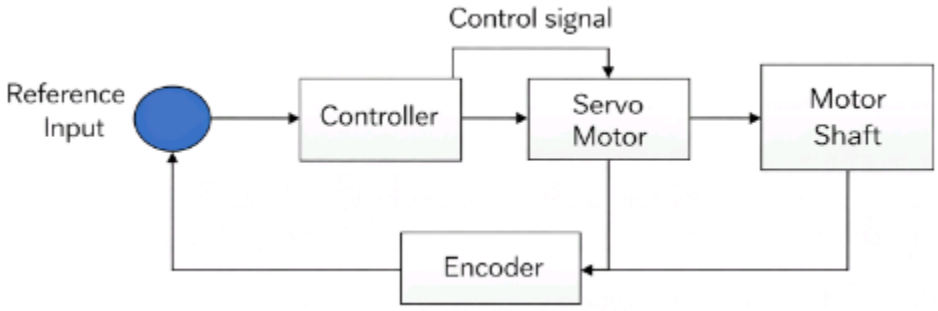


Fig. 3.2. Motor speed control structure.

- The PID controller is designed in the MATLAB/Simulink environment and is implemented in connection with the system via an Arduino board, as illustrated in Fig. 3.3.

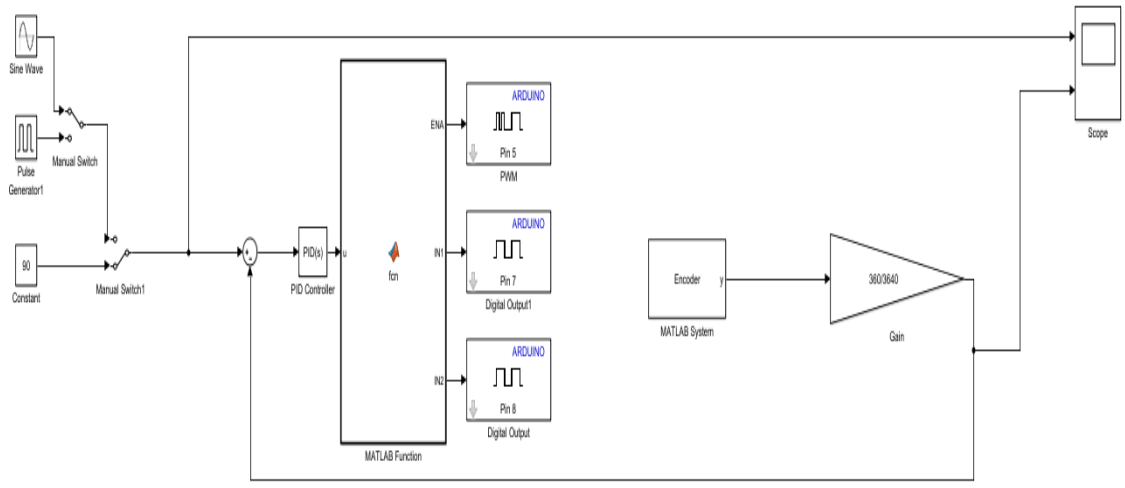


Fig. 3.3. Servo motor control system using Arduino.

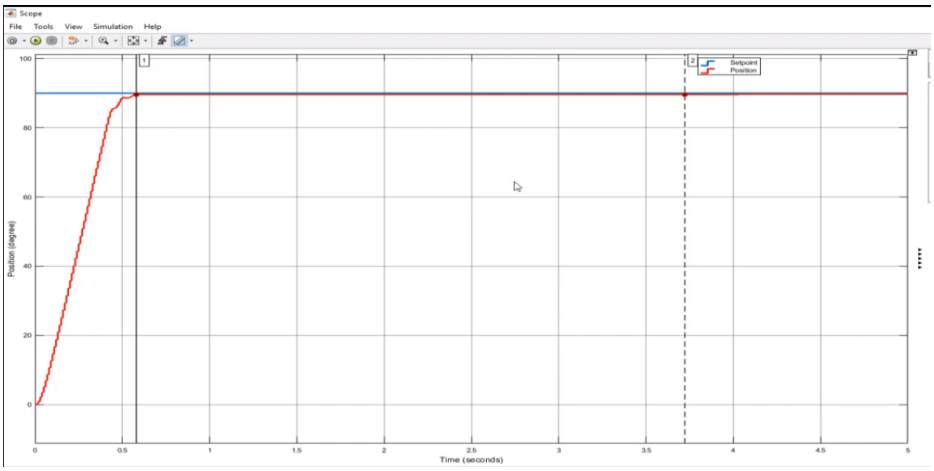


Fig. 3.4. Experimental results

Remark: Motion control in the system is achieved through servo motor actuation. The position signal obtained from the encoder is fed back and compared with the system's reference value. The controller then processes the resulting error and generates a suitable control signal, ensuring the motor output accurately follows the reference input.

IV. CONCLUSIONS

Servo motors are widely used in many industrial fields. This paper presents the Arduino library on the MATLAB platform and its application in the design of a servo motor controller. The results obtained from the system demonstrate the ability to connect via software and serve well for scientific research, teaching and training on real models.

REFERENCES

[1]. Hari K. K. et al., 2021, A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations, *Computer Science Review*, Vol.40 - 100364
[2]. Quang N. P., 2006, Simulink MATLAB Textbook, *Science and Technology Publishing House*.
[3]. Truong D. N. etal., 2018, Microcontrollers and Applications (Arduino User Guide), *Youth Publishing House.*
[4]. Trung N. K., 2022, Research and development of a microcontroller kit for training in the field of Mechatronics in the textile industry, *Hanoi University of Industry and Trade level sciencetic research project*
[5]. Vu Q. H., 2011, Electric Motor Control Engineering, *Education Publishing House*.
[6]. Trento D. et al., 2020, Application of Arduino-Based Systems as Monitoring Tools in Indoor Comfort Studies, *International Journal of Architectural Engineering Technology*, 2020, Vol.7.