Microcontroller Authentication System on Raspberry Pi Pico for IOT Devices

Adil Maidanov[1], Sabyrzhan Atanov[1], and Hüseyin Canbolat[2]

[1] *L N. Gumilyov Eurasian National University, 2 Satbaev St., Astana, 010000, Kazakhstan*
[2] *Ankara Yildirim Beyazit University, Kizilca District, Ankara, 06760, Turkey*

Abstract

The use of the MicroPython interpreter significantly simplifies the work with microcontrollers in comparison with classical integrated development environments based on C/C ++, as well as with RTOS (real-time operating systems). Using the user authentication application described in the paper, you can take advantage of the Raspberry Pi Pico microcontroller and the built-in USB HID (Human Interface Device) class that is natively supported on this board. This means it is not necessary to add any additional hardware to interact with it. Thus, this board, described in the article, can work as a HID device. You can configure it to work as a storage device, mouse, keyboard, webcam, or other IOT (the Internet of things) devices. The article describes the hardware implementation of the device for authentication. The scheme of the device and the necessary algorithm for generating and implementing a password are given. The authentication example was implemented on the Microsoft Windows operating system.

Keywords

Microcontroller, authentication system, human interface device, the Internet of things, integrated development environment

1. Introduction

Having a strong password may not always protect user accounts. Moreover, the unique and complex passwords for each electronic resource or application cannot guarantee absolute security against access of third parties. One of the possible solutions here is using of hardware security keys. A hardware security key is a device that adds another level of authentication when the users are signing to their accounts.

The aim of the research is to implement a one-time password generator using the MicroPython language and a low-cost RP2040 microcontroller (Raspberry Pi Pico) with USB HID support [1]. The procedure for generating and entering a one-time password was carried out using a microcontroller with a button and a computer. The user presses a button to generate a random password. The microcontroller is programmed to work with a keyboard, it can be connected to any device that supports a USB host, such as a PC, tablet, smartphone, etc. And no special software is required on the host device to perform the actions. The user will simply connect the microcontroller with a micro-usb cable and click on the password field. When the user presses the button in the microcontroller program, a password of the specified length will be generated.

The objectives of the research are following:

- To create an algorithm for one-time password generator in the MicroPython language;
- To assemble the hardware implementation using RP2040 microcontroller, a push button, and a computer;
- To test the resulting authentication system on Microsoft Windows;

Proceedings of the 7th International Conference on Digital Technologies in Education, Science and Industry (DTESI 2022), October 20–21, 2022, Almaty, Kazakhstan
EMAIL: makeadil@mail.ru (Adil Maidanov); atanov5@mail.ru (Sabyrzhan Atanov); huseyin.canbolat@gmail.com (Hüseyin Canbolat)
ORCID: 0000-0003-2392-5164 (Adil Maidanov); 0000-0003-2115-7130 (Sabyrzhan Atanov); 0000-0002-2577-0517 (Hüseyin Canbolat)

- To create an algorithm for one-time password generator in the Arduino IDE;
- To build the hardware implementation using ESP-32 microcontroller, a push button, a LED diode, and a computer;
- To compare gained results from different boards (Raspberry Pi Pico and Arduino);
- To illustrate the findings and incorporate them in work.

This tool is designed to generate random passwords, but it does not store or record any generated password, and the user must store the generated password somewhere to log in.

RP2040 is the first microcontroller product from Raspberry Pi. This microcontroller relates to the Raspberry Pi Pico family which contains three types: Raspberry Pi Pico, Pico H and Pico W (Figure 1). The advantages of this microcontroller include its high performance, low price, and low entry threshold for beginners in this field, due to the detailed documentation of MicroPython [2]. The comparison of Raspberry Pi Pico and Arduino microcontrollers is given in Table 1.

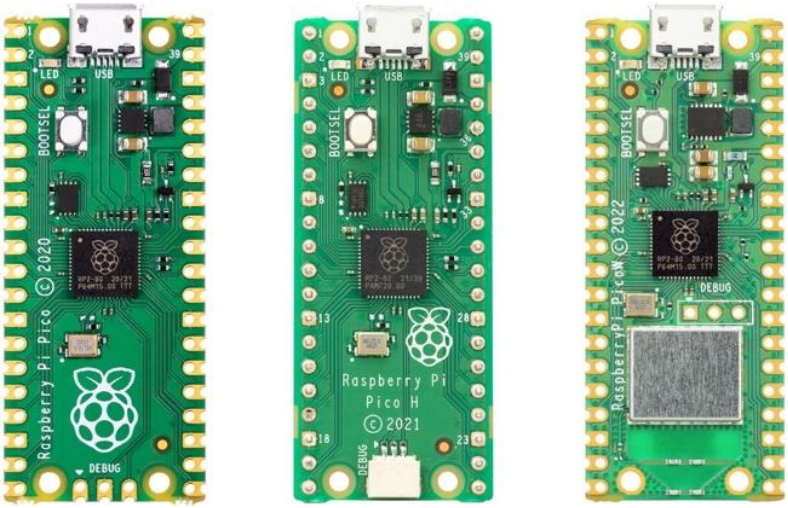

Figure 1: The types of Raspberry Pi Pico microcontroller: Raspberry Pi Pico (left), Raspberry Pi Pico H (center), and Raspberry Pi W (right)

Table 1
The comparison of Raspberry Pi Pico and Arduino microcontrollers

	Arduino	Raspberry Pi Pico
Microcontroller type	Single core Atmega328p	Dual-core RP2040
Core structure system	8-bit RISC	32-bit ARM cortex-MO+
CPU Speed	16MHz	48MHz, up to 133MHz
Random Access Memory (RAM) size	2 Kbyte	264kbyte (SRAM)
Flash Memory size	32 Kbyte	2 Mbyte (Q-SPI Flash)
E^2PROM	1 Kbyte	None
Coding	C uniform Arduino IDE	MicroPython, C/C++ languages
Power supply	5VDC through USB B	5 VDC through USB micro-B
The power of an Alternative Board	7-12 VDC through DC Socket	1.8-5VDC through VSYS pin
Microcontroller unit	5 VDC	3.3 VDC
USB interface	External USB serial IC	USB 1.1 Device and host
Program loader	USB B, virtual serial port	USB micro-B, USB mass Storage
General Purpose Input/Output and Digital	20	26
Analog-to-digital converter (ADC)	6 x 10bit	3 x 12 bit
UART	1	2

I^2C	1	2
The Serial Peripheral Interface (SPI)	1	2
Pulse width modulation (PWM)	6	16
Onboard LEDs	1 connected with D13	1 connected with GP25
Energy consumption on 12 Neopixel LEDs	5 V- 140 mA, 0.7 W	5V - 90mA, 0.45W!
Estimated cost	~109USD	~4USD

Modern technologies provide ready-made technical solutions for cryptography and data protection tasks on a computer, particularly the use of the TPM (Trusted Platform Module) architecture. TPM is a cryptoprocessor that provides a development environment for securely generating and storing encryption keys. These keys can be used to sign, encrypt/decrypt, and generate pseudo-random numbers. A computer with a TPM cryptographic module has a unique identifier built into the chip. The identifier is known to the software developer and is not subject to any configuration, i.e., it has a closed configuration. And this jeopardizes one of the virtues inherent in the Internet - anonymity. Moreover, since TPM is a complete architecture, this technology may not be effective for simple tasks requiring high data transfer rates. For example, we can cite the task of developing our algorithm for generating pseudo-random numbers for the Internet of Things proposed in this article.

The purpose of our article is to present a hardware solution for a generation and entering one-time password algorithm that can be applied to the Internet of Things technology, where the requirements for cryptographic protection are not so high. Still, it is essential to have an inexpensive hardware implementation. This approach can also be easily implemented on various microcontrollers, for example, on a TPM cryptographic module.

This article discusses the development of a hardware key on the Raspberry Pi Pico since this microcontroller has built-in USB HID support, which allows us to work with various IoT devices without using any additional hardware or special firmware. For example, if we are using serial communication, we must have a program running on the PC to receive data from the Arduino board. But this Raspberry Pi Pico board will work as a USB keyboard when using the USB HID class. Subsequently, this development can be applied to work with other peripheral devices, a webcam, and IoT devices. However, with some changes, this development can be implemented on different tools, for example, on the ESP-32 microprocessor board (Figure 2) with special firmware or using mobile robotic-controlled systems based on neural networks. As can be seen from, it should be said that the use of robotic systems significantly increases the project's cost, and this technology does not provide cryptographic data protection.

2. Materials and Methods

The relatively new development environment for Python programming, Thonny Python IDE (Figure 2), was chosen as the development environment, and the MicroPython language was chosen as the programming language.

To create a one-time password for the Internet of Things in MicroPython, we used the random module and the random element of the non-empty sequence *random.choice (sequence).*

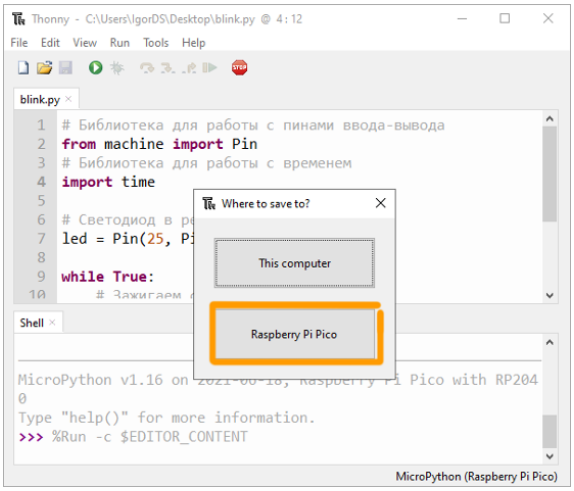

Figure 2: Thonny Python IDE programming environment

2.1. One-time password generation program on Thonny Phython IDE

The main difference between the MicroPython development environment and the Arduino IDE is the way of working with code. The process of creating code begins with editing the code in the Arduino IDE; then, it is a step of compiling the source code, from which we get a binary file. And at the end, this file is loaded into the microcontroller's Flash memory. From the Figure 3, you can see the tutorial model on Thonny Python IDE for password generation program.

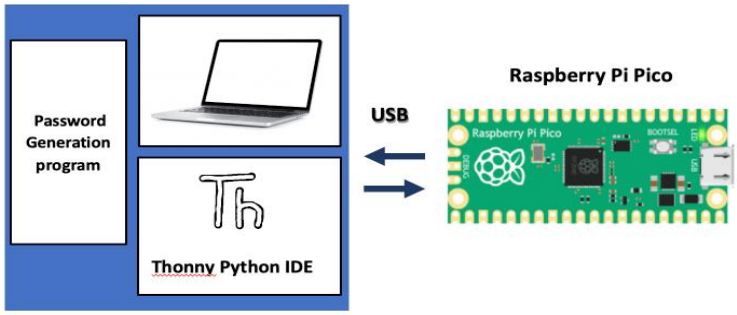

Figure 3: Tutorial model on Thonny Python IDE for password generation program

And when working with MicroPython, we edit the code using a development environment compatible with this interpreter (for example, Thonny Python) and then load the source code into the microcontroller. Since the code is now stored in the microcontroller's Flash memory, we can view and edit the source code at any time without having to compile it. In Arduino, changing the source code means re-compiling for the changes to take effect. The last step in working with the Thonny Python IDE is to convert the code into a binary file by the MicroPython interpreter (Figure 4).

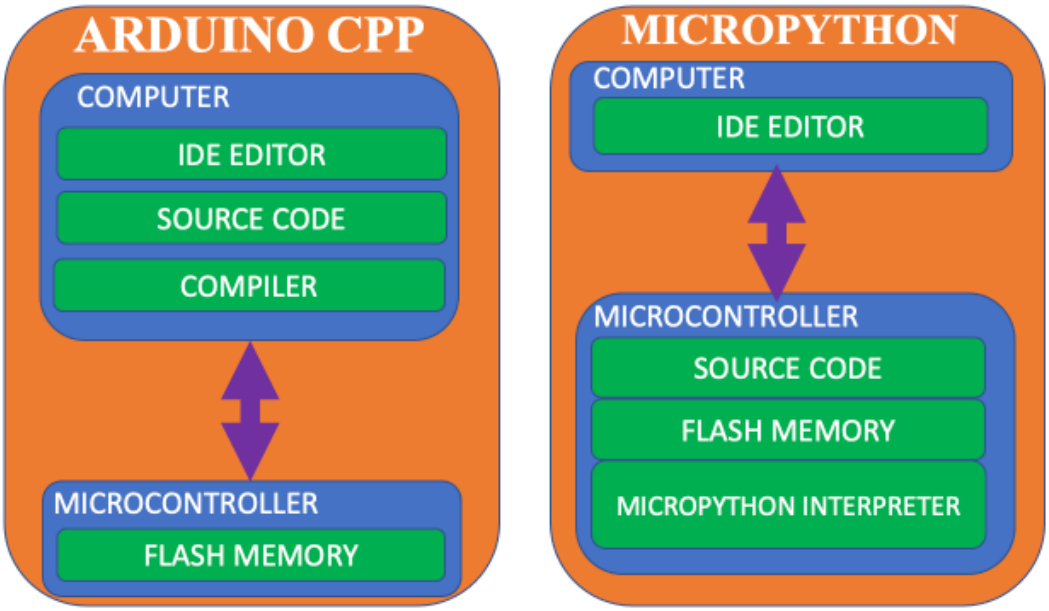

Figure 4: Differences between working on the MicroPython development environment and the Arduino IDE

MicroPython is a complete Python compiler and runtime that runs on bare metal. You get an interactive prompt by reading–the eval–print loop to implement commands instantly and the ability to execute and bring in scripts from the onboard file system. The REPL provides the user with easy interaction due to insert mode, history, auto-indentation, and tab completion. MicroPython aims to be as compatible as possible with regular Python, so by knowing Python, you can quickly obtain the MicroPython language. At the same time, the more you study MicroPython, the better your understanding of Python will become. Apart from establishing a set of Python's core libraries, MicroPython includes modules such as "machine" to access low-level equipment.

MicroPython uses many advanced programming techniques and methods to keep it compact while maintaining a complete feature set.

The following list includes some of the most popular of them:

- highly customizable because of many compile-time configuration options;
- compatible with many architectures (x86, x86-64, Xtensa, etc.);
- an extensive set of tests with over 500 tests and more than 18000 separate test cases;
- core code coverage plus advanced modules over 98%;
- quick startup time from boot to first script load;
- convenient, quick, and reliable heap garbage collector;
- Exception of MemoryError occurs if the heap is exhausted;
- Exception of RuntimeError occurs if the stack limit has been reached;
- supporting for running Python code on a hard interrupt with minimum latency;
- errors are backtraced, and the line number of the code will be reported;
- folding constants in a parser and a compiler;
- pointer marking to match strings, small integers, and class objects in machine language;
- transparent crossing from small to large integers;
- support for 64-bit Not-a-Number boxing object model;
- support for 30-bit padded floating-point numbers that don't require heaps of memory;
- cross compiler and frozen bytecode to have precompiled scripts that don't need RAM;
- multithreading through the "_thread" module, with an additional global interpreter lock;
- native emitter that aimed to native code directly, not bytecode VM;
- has an inline assembly [2].

Device algorithm for generating and entering a one-time password.

The Interrupt Service Routine (ISR) detects changes to the button (whether it is pressed or not). This is an efficient and preferred way to see an external event, such as a sensor reading or, in this case, a button press. It is constantly monitoring the output of a button and checking if it is pressed or not will waste CPU processing power. We can force our microcontroller to save battery power if it is running on battery so that it wakes up from sleep mode when a button is pressed, but this is not implemented in this algorithm.

The designed project circuit consists of a RP2040 development board Raspberry Pi Pico and a push button (Figure 5). The push button is connected to GPIO-14 of the microcontroller to the ground.

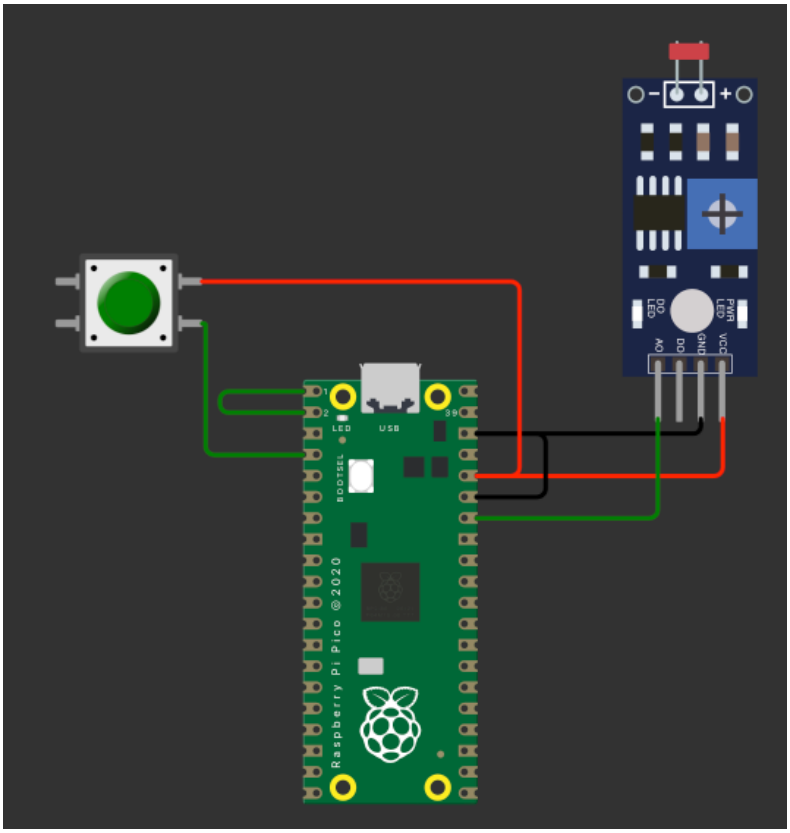

Figure 5: A circuit diagram of a designed project using Raspberry Pi Pico microcontroller

A random password is generated from letters (upper and lower case), numbers (0-9), and symbols.

Program progress:

- CPU interrupt on button press;
- Run an algorithm to suppress electromagnetic interference (EMI) when a button is pressed;
- Increase the value of the variable to 1 if the button is pressed;
- Check the value if it has changed from the previously saved value;
- If the value has been changed, run the random password generator algorithm;
- Generate n password with l length from small letters only;
- Send a randomly generated password to a USB keyboard;
- A short delay of 0.5 s.

Below you can see a flowchart of the algorithm for generating and entering a one-time password (Figure 6).

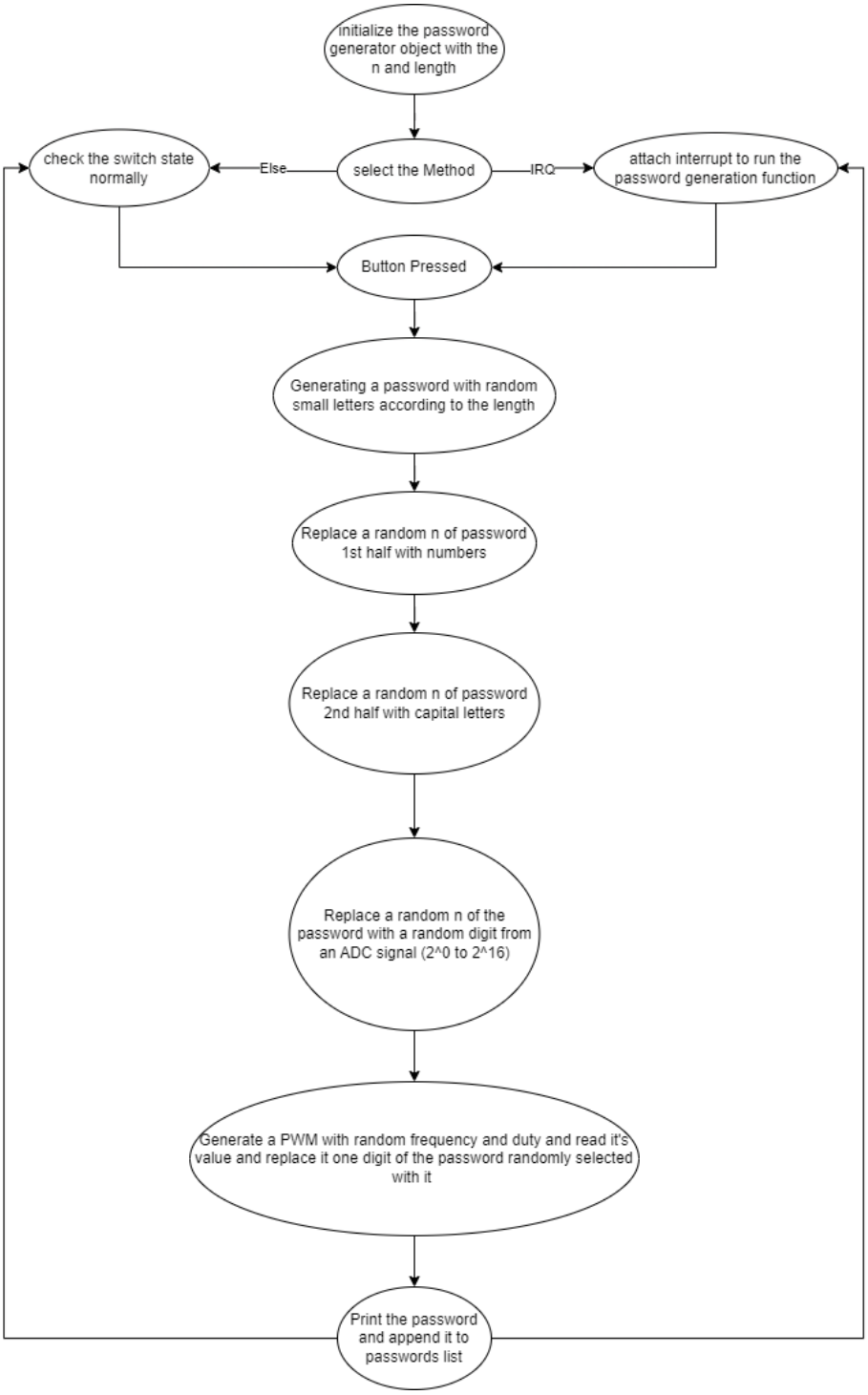

Figure 6: Algorithm for generating and entering a one-time password

2.2. One-time password generation program on Arduino IDE

To compare two different microcontrollers, we assembled a one-time password generation program also on Arduino IDE using an ESP-32 board, the push button, and the LED diode (Figure 7). The generated password will consist of upper- and lower-case letters, numbers, and symbols. User can request a new password by pressing a push button. A status led is used to display the password is ready to be used by user.

Program progress:

- Set the password length and constant values to the led pin and the push button;

- Initialize internal pseudo-random number generator by randomSeed() function;
- Initialize led row and lines;
- Increase the value of the variable to 1 if the button is pressed;
- Check the value if it has changed from the previously saved value;
- If the value has been changed, run the random password generator algorithm;
- Sent out the password to the serial port;
- Use the blink function to control the password generation.

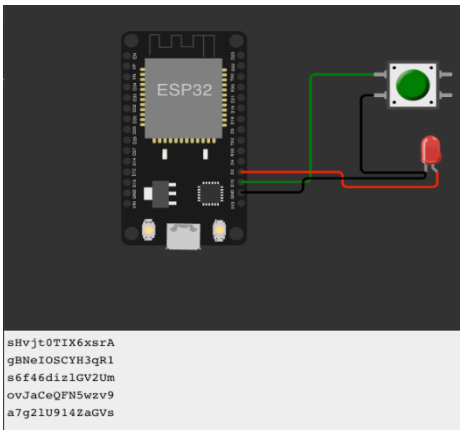

Figure 7: A circuit diagram of a designed project using ESP-32 microcontroller

3. Experimental Results

The authentication system example was implemented on the Microsoft Windows operating system. The user will connect the microcontroller with a micro-usb cable and select on the password field. When the user presses the button in the microcontroller program, a password of the specified length will be generated (Figure 8). This device is designed to generate random passwords, but it does not store or record any generated password, and the user must store the generated password somewhere to log in.

Figure 8: The implementation of the authentication system example on Microsoft Windows OS

Having performed a hardware implementation on a microcontroller and displayed the obtained data on a graph built on MS Excel (Figure 9), we saw a uniform dispersion of random numbers. Then, to plot those numbers, we used the *randomSeed()* function on Arduino IDE, which allowed us to introduce more randomness of numbers with each generation than the standard random function in Arduino. The flow-chart of using random and randomSeed functions on Arduino IDE with RP-2040 microcontroller can be seen in

Figure 10. Then, we can plot the 1800 random numbers generated by RP-2040 (Figure 11). The data on the serial plotter represent the random numbers generated by the device vs total numbers.

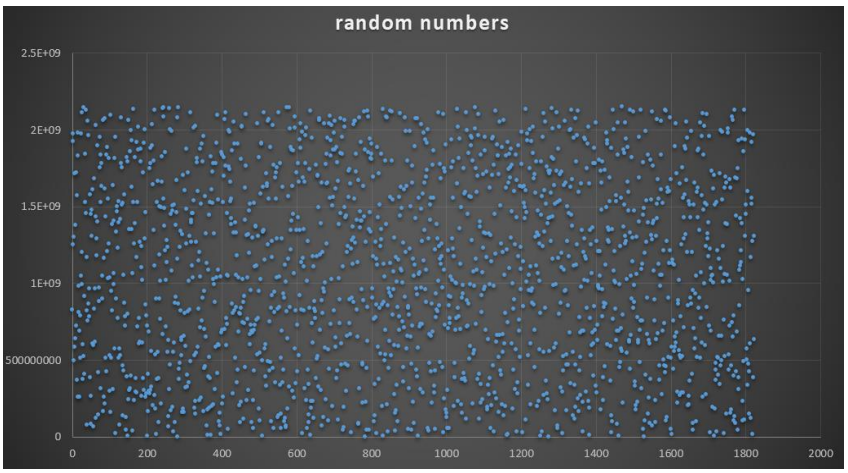

Figure 9: A graph of a series of about 1800 random numbers generated by RP-2040

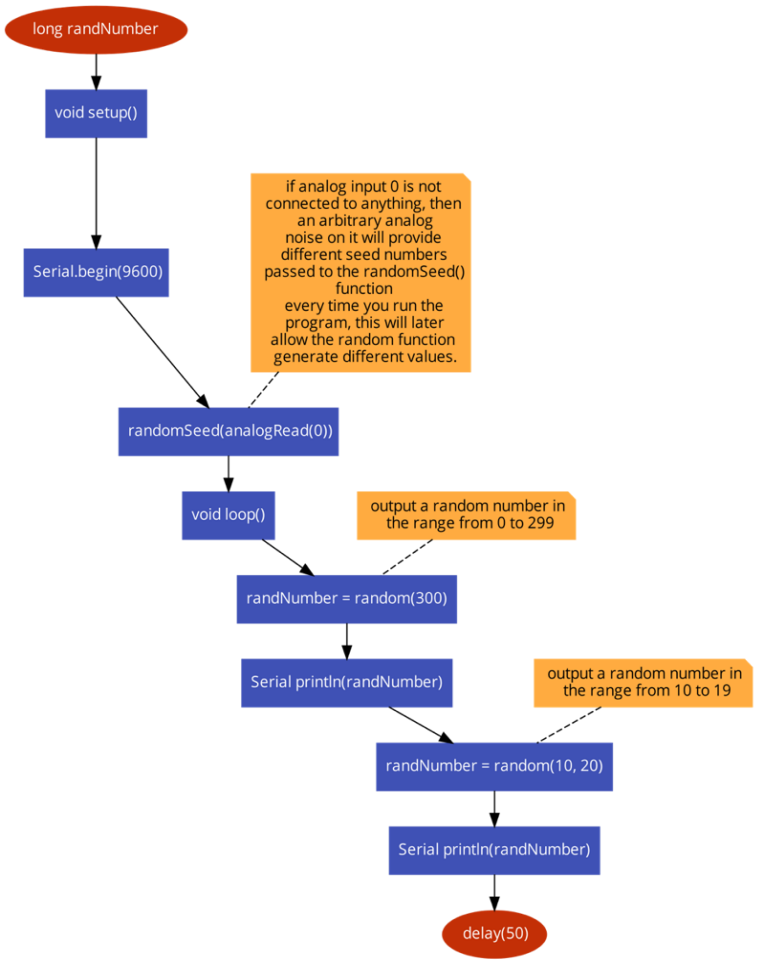

Figure 10: The flow-chart using random () function on Arduino IDE with RP-2040 microcontroller

Using the random () or randomSeed () functions is convenient for generating a password and key from random characters. The pseudo-random number generator imitates the randomness or randomness of the appearance of numbers. Still, if you analyze a number over a sufficiently long period, you can notice a particular pattern [3]. The article covers the mathematical description of such a pseudo-random

number generator well [4]. An illustration of utilization as a counter for controlling the motor working modes is considered.

It is essential to understand that when using the random () function, in the output, the same list of pseudo-random numbers will be generated each time [5].

The randomSeed () function takes a value (for example, an integer) and uses a number to modify the random list generated by the random() function. You can put randomSeed () in the setup function and use the random () function in an infinite loop. But in this case, there will be a catch: although the sequence of random numbers will be different when using the randomSeed () function, it will still be the same every time the sketch is run.

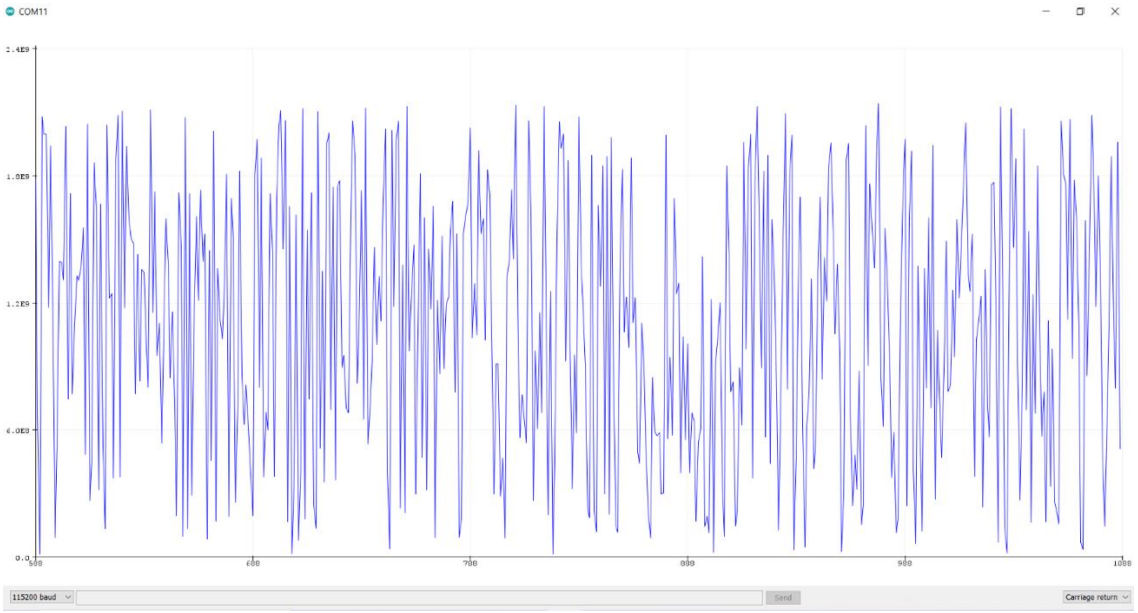

Figure 11: Serial Plotter output of random numbers In Arduino IDE

4. Conclusion

Today, information security is one of the world's most essential and relevant areas. In our research, a hardware implementation of a device based on the Raspberry Pi Pico board was implemented to generate and enter a character password, which ensured the protection of personal computer user data.

The advantages of this device include its availability, cross-platform, and convenient and intuitive programming environment Thonny Python IDE. Further development of this work can be the improvement of the encryption algorithm, as well as the creation of an application for the Windows operating system to provide greater security using an encryption key implemented on a microcontroller and connected to a specific computer via a USB connector. Also, this hardware solution can be easily implemented for removable physical media.

In the event of an attack on the random number generator, and if the attacker has access to the device's hardware, then external sources of entropy must be used very carefully since there is a risk of signal spoofing from an external source. A possible solution to this issue is using internal sources for autonomous operation without needing external power supplies.

Another good idea is to accumulate entropy all the time when it is not required and apply it when it is necessary to generate a new random number. For such situations, the Entropy pool is usually used - an array where one of the functions of the pseudo-random number generator is initialized from time to time and where data obtained from entropy sources is regularly added.

The concept of the created basic algorithm for generating pseudo-random numbers, described in this article, can be further used for any other sets of key pairs (for pairs, triples, quadruples, etc.), thereby increasing the software module and multiplying key generation without changing the algorithmic program complexity.

5. References

[1] The Technical Specification of Raspberry Pi Pico, 2022. URL: https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html#technical-specification.

[2] The Documentation of MicroPython language, 2022. URL: https://micropython.org.

[3] Forum about Arduino, 2022. URL: http://arduino.ru/forum.

[4] M. Jamshidi, S. Atanov, Z. Mukanova, Features of Hardware and Software Smoothing of Experimental Data of Gas Sensors, in: Proceedings of the IEEE International Conference on Smart Information Systems and Technologies, SIST, 2021, 407-413. doi: 10.1109/SIST50301.2021.9465981.

[5] S. K. Atanov, A. G. Bulaev, A. U. Bugubaeva, M. F. Baimuhamedov, K. M. Zhunusov, System of cryptographic protection of information based on deterministic chaos, International Journal of Innovative Technology and Exploring Engineering 8 (2019) 4495–4498. doi: 10.35940/ijitee. L3527.1081219.