ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023

THE MULTIDISCIPLINARY JOURNAL OF SCIENCE AND TECHNOLOGY

VOLUME-3, ISSUE-3

Development of a Hardware Module for Programming Microcontrollers Based on the Cortex-M Architecture

Nevliudov Igor, Yevsieiev Vladyslav, Maksymova Svitlana, Klymenko Oleksandr

Department of Computer-Integrated Technologies, Automation and Robotics, Kharkiv National University of Radio Electronics, Ukraine

Abstract:

This article analyzes modern programming interfaces of microcontrollers with Cortex-M core architecture. The existing software and hardware tools for programming microcontrollers were reviewed and analyzed. Methods of automating the microcontroller programming process were also analyzed. Based on the analysis, a structural diagram of the layout was developed, hardware and software components were selected, and a number of experiments were conducted to evaluate the execution time of the micro controller programming script and stress test the host server, which showed a good result.

Key words: Industry 4.0, IIoT, Microcontrollers, Cortex-M, JTAG, SWD, Automation, Firmware.

Introduction

The development of a hardware module for microcontrollers automating the programming is one of the most important stages in modern devices production organization within the Industry 4.0 concept [1]. Over the past few decades, there has been a positive trend towards the use of microcontroller control devices. Because of this, more and more devices in a wide range of industries have microcontrollers in them that must be programmed at device production stage. During the production of these devices, the task arises to automate the process of downloading the software to the microcontroller as part of serial or mass production. And this task should be solved not by classical approaches – programming each one separately, but by a synchronous mass approach using Industrial Internet of Things (IIoT) technologies [2], which will save production time and increase economic profitability.

Thus, the topic of this study within the framework of the development of a hardware and software module for programming microcontrollers based on the Cortex-M architecture is relevant and is found in different studies [3]-[14].

Related works

Microcontrollers are used in many everyday devices and will become more common as the Internet of Things (IoT) gains momentum, there are studies and publications related to automating the programming of microcontrollers, let's look at some of them.

Nathanael R. Weidler and others developed Return Oriented Programming (ROP), a technique used to take over the execution of a program by causing the return address of a function to be changed using an exploit vector, and then returning to small segments of innocuous code located in executable memory one after another [15]. Analyzing this method, the following conclusions can be drawn: the proposed solution makes it possible to partially and/or fully control the Tiva TM4C123GH6PM microcontroller, which uses a Cortex-M4F processor.

Per Lindgren and others propose using Real-Time For the Masses (RTFM), a set of languages and tools being developed to facilitate embedded software development and provide highly efficient implementations designed for static verification [16]. It is worth noting that the

ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023

RTFM core is an architecture designed to provide highly efficient and predictable scheduling based on stack resource policies targeting bare metal (single-core) platforms, which is not suitable for the solution in this research.

Mohammad Hossein Askari Hemmat et al.'s work redefines the term NuAC to support code generation for ARM Cortex-M processors and introduces an automated SysML activity diagram to an RTX (Keil Real-Time Operating System) code generator that uses the mapping rules expressed in NuAC [17]. This solution is widely used for modeling and analysis of complex systems and has become the de facto standard for software and embedded systems.

The research by Tomáš Jakubík was Of particular interest, which described a project for a simulator of Cortex-M microprocessors. This project is based on the Unicorn Engine, which is used to simulate the ARM core. The advantage of this project is the ability to download factory firmware and replace microprocessor peripherals. The same firmware can be executed on a physical board and the same firmware can be simulated. This enables rapid continuous integration and testing in embedded software development [18]. The proposed solution inspired the researchers to develop a hardware and software complex for programming microcontrollers based on the Cortex-M architecture.

Development of a hardware module for programming Cortex-M

Cortex-M is a family of microprocessor cores from the ARM company, which are designed for use in microcontrollers, ASIC (decrypts – "application special integrated circuit"), user-programmable gate arrays (PCVM), and systems on a crystal (SNA) [19]. Cores from the Cortex-M family are used not only as a microcontroller core, but also hidden inside an SNC, such as power management controllers, I/O port controllers, touch screen controllers, smart battery charge controllers, and sensor device controllers [20]. The Cortex-M processor family is optimized for energy-efficient microcontrollers.

In Cortex-M processors, at the system design stage, there is a choice between two protocols, Joint Test Action Group (JTAG) and Serial Wire Debug (SWD) [21]. An example of a JTAG connection is shown in Figure 1.

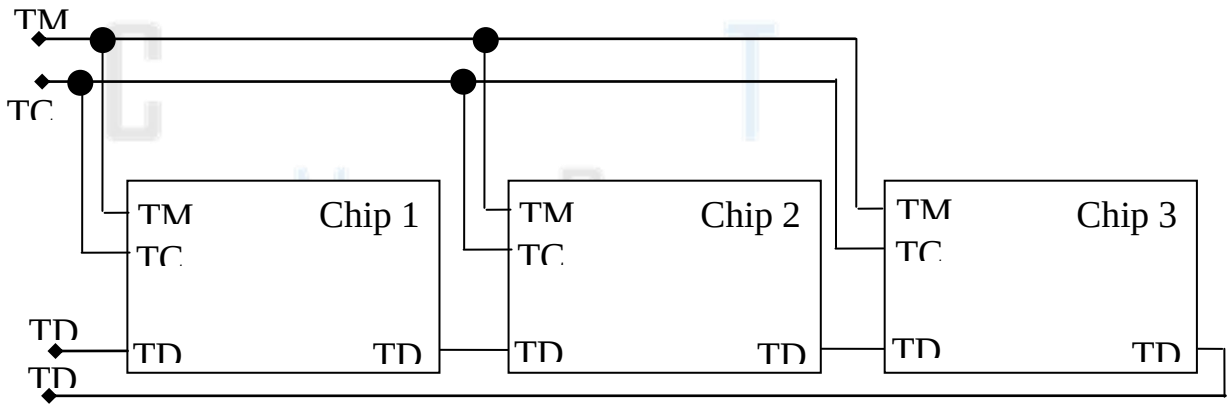

Figure 1: Connecting chips using JTAG

This interface is four or five dedicated pins of the chip:
– TCK (test clock) – a clock signal, the frequency is limited to 40 MHz;
– TDI (test data input) – sequential data input, such as control commands and data;

ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023
THE MULTIDISCIPLINARY JOURNAL OF SCIENCE AND TECHNOLOGY
VOLUME-3, ISSUE-3

– TDO (test data output) – output for serial data from the chip;

– TMS (test mode select) – allows switching chips into debugging mode and changing test/debugging modes;

– TRST (test reset) – allows you to reset the target chip to its initial state.

The second interface, SWD, requires only two pins to connect, which is ideal for devices with a limited number of pins. The SWD interface requires only the TMS and TCK pins from the JTAG interface. As a result, it is necessary to develop an experimental prototype with the possibility of universal programming of processors of the Cortex-M family, both with the JTAG and SWD interfaces.

At the first stage, we will develop a structural diagram of the hardware subsystem (Figure 2a) and the software subsystem (Figure 2b) of the layout for programming Cortex-M family processors.

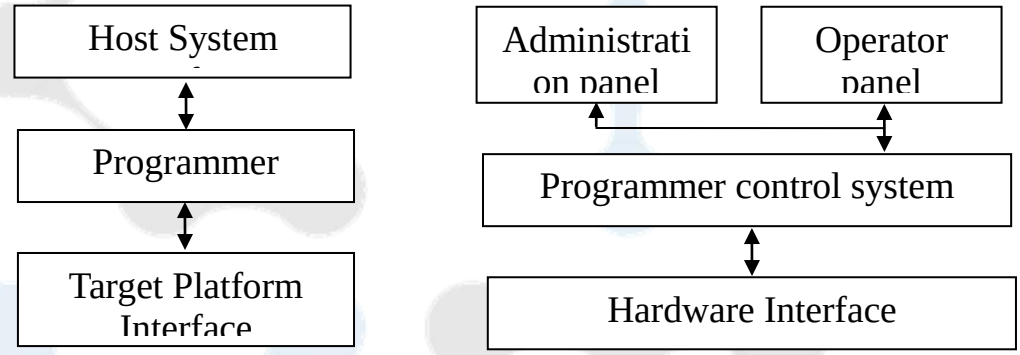

Figure 2: The structure of the hardware and software programming subsystem of Cortex-M family processors

Based on the developed structures of the hardware and software programming subsystem of the Cortex-M family processors, we will select the hardware components. One of the requirements for hardware modules is physical USB and Ethernet interfaces. USB (English Universal Serial Bus) is a serial communication interface that can be roughly placed on the first two levels of the Open Systems Interconnection (OSI) model, that is, the physical and channel levels [22]. At the physical level, USB uses 4 wires, 2 for power (5V and Gnd), and 2 for data transmission – a differential pair. At the channel level, a packet transmission protocol is used to ensure the reliability of data transmission between devices. USB devices are also on the last, seventh, application level of the OSI model, and the support of the Linux distribution OS. As a result, Raspberry Pi Zero W [23], NanoPI NEO [24] and Orange Pi Zero LTS [25] satisfy the requirements. It is worth noting that the NanoPI NEO only has a wired network connection, while the Raspberry Pi Zero W only supports a wireless Wi-Fi connection and the Orange Pi Zero LTS supports both ways to connect to the network, but it costs a lot more more than NanoPI NEO. As a result, we will choose NanoPI NEO (Figure 3.a) and the SEGGER J-Link microcontroller programmer (Figure 3b) as part of the development of the layout, the general of which is presented in Figure 3.

The next step is to calculate the value of the optimal RC circuit for the signal line of the SWD interface. A low-pass filter, also known as an RC filter, is used in data exchange lines to

ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023

cut off high-frequency interference, increases resistance to electromagnetic interference and radio interference.

a) b)

Figure 3: General view of the selected hardware modules

Since the RC filter is an aperiodic link of the first order, it is described by the following differential equation:

$$T\frac{dy}{dt} + y(t) = kx(t), \quad 1)$$

where t - time constant.

As a result, the transition function of the link is expressed:

$$h(t) = k(1 - e^{-\frac{\tau}{T}}) \cdot 1(t). \quad 2)$$

For an RC circuit, the time constant is expressed as

$$T = RC, \quad 3)$$

where R – resistor resistance; C – capacitor capacity.

The time constant is inversely related to the cutoff frequency and is depicted on the logarithmic amplitude-frequency characteristic (LAFC). Cutoff frequency ω_c - characterizes the bandwidth of the filter. When the input signal has a frequency lower than the cut-off frequency, the output signal is not changed, or is not changed significantly, otherwise the filter smoothes and changes the amplitude of the signal with a frequency higher than the cut-off frequency. The ratio of the cutoff frequency and the time constant is expressed as:

$$\omega_c = \frac{1}{T} = \frac{1}{RC}. \quad (4)$$

In order to preserve the clarity of the signal at non-ideal real values of the signal frequency, for example, due to the error of the clock signal of the control device, we will add to

ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023

the bandwidth a window of 10% of the maximum recommended frequency of the SWD interface, that is, we will add 400 kHz to the maximum 4 MHz and get a cutoff frequency of 4, 4 MHz. Analyzing formula (4), we get that in order for the bandwidth of the first-order aperiodic link to be 4400 kHz, the product of the resistance of the resistor and the capacity of the capacitor must be at least . Taking, for example, a resistor with a resistance of 1.5 kΩ from the available ratings, it will be enough to take a capacitor with a capacity of 15 nF. At the same time, this resistor will have a current limiting function. The connection of the calculated RC filter is presented in Figure 4.

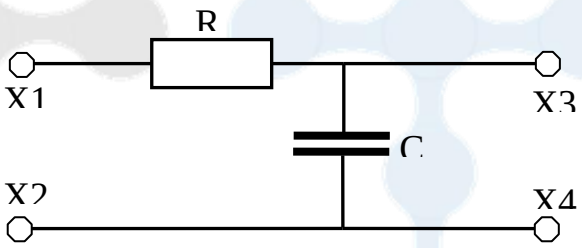

Figure 4: Schematic image of the connection of the calculated RC filter

We connect the GND contacts of the programmer and the target system to X2 and X4. We connect the input signal from the programmer to X1. At the X3 output, we receive a signal that needs to be connected to the target system.

Also, it is necessary to take into account the response delays of the target platform, which are specified in the documentation of the SWD protocol [26]. We assume that the connection to the target platform and the data transfer are error-free. Direct data transfer depends on the amount of data to be transferred. After analyzing the given information, we distinguish the permanent and variable parts of data recording to flash memory. In the permanent part, we take into account all packages of the preparatory stage. The variable part includes the first four points of the data transfer stage, for each page of the storage device. For STM32 microcontrollers, the most common flash memory page size is 2048 bytes, or 2 KB. To calculate the total transmission time, we use the following formula:

$$T_{gen} = T_{peaceful} + T_{var\_page} + T_{var\_data}, \quad 5)$$

where: $T_{peaceful}$ – constant part time [c]; T_{var_page} – variable part time per page [c]; T_{var_data} – time variable part for data transfer [c].

The time for which each data packet is transmitted is calculated according to the formula:

$$T_{package} = \frac{b_{package}}{v}, \quad 6)$$

where: $b_{package}$ – the number of bits to be transmitted; v – signal frequency.

Using formula (6), we calculate the time it takes to transmit the constant part ($T_{peaceful}$), which is sent at the beginning of each communication session

ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023

$$T_{peaceful} = 3 \cdot T_{package}. \quad 7)$$

Formula for calculating the time it will take to transfer the variable part of each page (T_{var_page}):

$$T_{var\_page} = 4 \cdot T_{package} \cdot [\frac{n}{B_{page}}], \quad 8)$$

where: n – the total number of bytes in the transfer; B_{page} – number of bytes per page.

To calculate the time it takes to transfer the variable part with data (T_{var_data}) for the target system, we use the following formula

$$T_{var\_data} = \left[\frac{8 \cdot n}{b_{package}}\right] \cdot \frac{1}{v}, \quad 9)$$

where: n – the total number of bytes in the transfer; $b_{package}$ – the number of payload bits in a packet; v – signal frequency.

Substitute formulas 6-9 into expression 5, simplify 6 and 7 using expression 6, and obtain the total transfer time (T_{gen}):

$$T_{gen} = 3 \cdot T_{package} + (4 \cdot T_{package} \cdot \left[\frac{n}{B_{page}}\right]) + (\left[\frac{8 \cdot n}{b_{package}}\right] \cdot (\frac{1}{v})) =$$
$$= \frac{3 \cdot b_{package}}{v} + (4 \cdot \frac{b_{package}}{v} \cdot \left[\frac{n}{B_{page}}\right]) + (\left[\frac{8 \cdot n}{b_{package}}\right] \cdot (\frac{1}{v})) = \quad 10)$$
$$= (\frac{1}{v}) \cdot (3 \cdot b_{package} + 4 \cdot b_{package} \cdot \left[\frac{n}{B_{page}}\right] + \left[\frac{8 \cdot n}{b_{package}}\right]).$$

When using the maximum frequency of the SWD signal – 4 MHz, and 2048 bytes per page in the memory of the target system, and also knowing that in each packet we transmit 45 bits, we can simplify formula (10) and obtain:

$$T_{gen} = (\frac{1}{4 \cdot 10^{-6}}) \cdot (3 \cdot 45 + 4 \cdot 45 \cdot \left[\frac{n}{2048}\right] + \left[\frac{8 \cdot n}{32}\right]) =$$
$$= 0{,}25 \cdot 10^{-6} \cdot (135 + 180\left[\frac{n}{2048}\right] + \left[\frac{8 \cdot n}{32}\right]). \quad 11)$$

ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023

THE MULTIDISCIPLINARY JOURNAL OF SCIENCE AND TECHNOLOGY

VOLUME-3, ISSUE-3

Using the formula (11), it is possible to calculate the total time of writing to the flash memory of an STM32 microcontroller with a core of the Cortex-M family, a test program with the size of 16724 bytes, which was 1.484 ms.

To check the correctness of the performed calculations, let's assemble a test prototype of the software and hardware module for programming microcontrollers based on the Cortex-M architecture, which is presented in Figure 5.

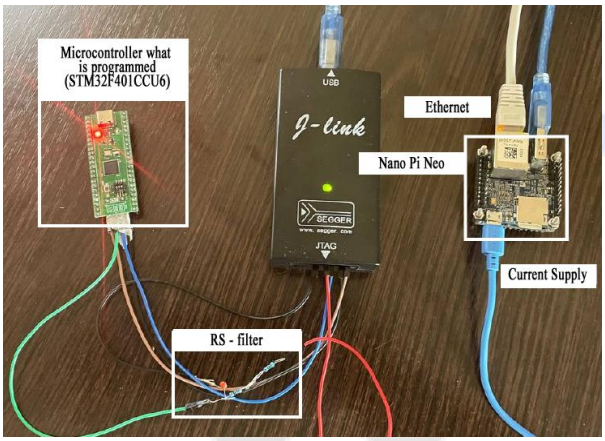

Figure 5: Test prototype of a hardware module for programming microcontrollers based on the Cortex-M architecture

Experimental studies of the developed module for programming microcontrollers based on the Cortex-M architecture

The first test will be a software test of the speed of downloading the program to the STM32F401CCU6 using OpenOCD [27]-[29]. Although we have calculated the approximate programming time, but these were the conditions of an ideal hardware part that has at least 2 processor cores, one for communication with the host platform, the other for communication with the target platform, and the write delays to the flash memory were also not taken into account of the microcontroller itself. Also, the download time depends on the speed of the host platform itself and the control program, in our case OpenOCD. To conduct this experiment, we will use the Linux utility – time, which, when passing another command as an argument to it, calculates the time it takes to execute this command. The test results are shown in Figure 6.

ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023

VOLUME-3, ISSUE-3

```
Open On-Chip Debugger 0.12.0+dev-01154-g91bd43134 (2023-04-30-11:36)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Open On-Chip Debugger 0.12.0+dev-01154-g91bd43134 (2023-04-30-11:36)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
swd
adapter speed: 4000 kHz
Info : J-Link V9 compiled May  7 2021 16:26:12
Info : Hardware version: 9.60
Info : VTarget = 3.264 V
Info : clock speed 2000 kHz
Info : SWD DPIDR 0x2ba01477
Info : [stm32f4x.cpu] Cortex-M4 r0p1 processor detected
Info : [stm32f4x.cpu] target has 6 breakpoints, 4 watchpoints
Info : starting gdb server for stm32f4x.cpu on 3333
Info : Listening on port 3333 for gdb connections
[stm32f4x.cpu] halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000324 msp: 0x20010000
Info : device id = 0x00016423
Info : flash size = 256 KiB
auto erase enabled
wrote 16384 bytes from file ./picoLED_1.hex in 0.857928s (18.650 KiB/s)
shutdown command invoked

real    0m4.912s
user    0m0.226s
sys     0m0.186s
```

Figure 6: The result of the microcontroller programming script execution time evaluation

As you can see, the result of STM32F401CCU6 microcontroller programming script execution time evaluating showed the following results: real execution time (the entire, total time when the command was active, from the beginning to the end of its execution) – 4.912s; user (the time the called command was executed outside the OS core, in user space) – 0.226s; sys (the time the called command was executed in the OS kernel, in the system space) – 0.186c.

Also, we will conduct an experiment with a stress load, the purpose of which is to check whether our system will maintain the expected response speed during a significant increase in the load on the host server. Since the system was designed for use in a small company, we will determine that the maximum level of simultaneous requests is 10. The test will be implemented using a script written in Python using the requests, time, concurrent and matplotlib libraries to display the results. We received the response time graph for the total number of 1000 requests, which is shown in Figure 7.

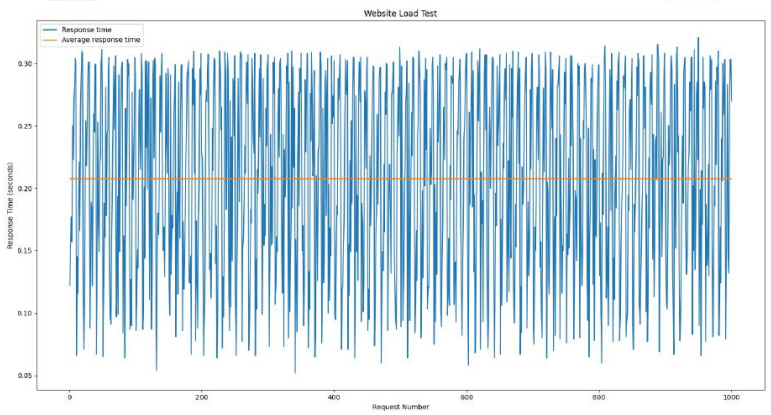

Figure 7: Host-server stress testing result

Analyzing the results of the first experiment, knowing that the complete cycle of script execution: reading the identifier, programming, writing to the history of the unique identification number takes almost 5 s, it is necessary to optimize this process, one of the solutions is to use

ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023
THE MULTIDISCIPLINARY JOURNAL OF SCIENCE AND TECHNOLOGY
VOLUME-3, ISSUE-3

multithreading and using 2 more available, but not soldered USB on the NanoPI NEO board to connect two hardware programmers to the host platform. Evaluating the result of the second experiment, it can be confirmed that the system maintains stability and the speed of returning responses to requests.

Conclusion

This article analyzes modern programming interfaces of microcontrollers with Cortex-M core architecture. The existing software and hardware tools for programming microcontrollers were reviewed and analyzed. Methods of automating the microcontroller programming process were also analyzed. Based on the analysis, a structural diagram of the prototype was developed. Also, hardware and software components were selected that correspond to the technical task and have the ability to preserve functionality when the number of users increases, on the basis of which the layout of the microcontroller programming system based on the Cortex-M architecture was assembled. Experimental studies of the speed of programming the microcontroller by the test program and the stability of the host system to an increase in the flow of requests from users were conducted.

As a result, an automated microcontroller programming system based on the ARM Cortex-M family of processors was implemented.

In the future, it is planned to expand the functionality of the software module - implement firmware and user statistics, hardware subsystem - expand the list of supported microcontrollers.

References:

1. Mijailović, Đorđe& et al. (2021). A Cloud-Based with Microcontroller Platforms System Designed to Educate Students within Digitalization and the Industry 4.0 Paradigm. Sustainability, 13(22), 12396.
2. Ala-Laurinaho, Riku, & et al. (2020). Open Sensor Manager for IIoT. Journal of Sensor and Actuator Networks, 9, 2(30).
3. Attar, H., & et al.. (2022). Zoomorphic mobile robot development for vertical movement based on the geometrical family caterpillar. Computational Intelligence and Neuroscience, 2022.
4. Tvoroshenko, I., & et al.. (2020). Modification of models intensive development ontologies by fuzzy logic. International Journal of Emerging Trends in Engineering Research, 8(3), 939-944.
5. Al-Sherrawi, M. H., & et al.. (2018). Corrosion as a source of destruction in construction. International Journal of Civil Engineering and Technology, 9(5), 306-314.
6. Dadkhah, M., & et al.. (2019). Methodology of wavelet analysis in research of dynamics of phishing attacks. International Journal of Advanced Intelligence Paradigms, 12(3-4), 220-238.
7. Attar, H., & et al.. (2022). Control System Development and Implementation of a CNC Laser Engraver for Environmental Use with Remote Imaging. Computational Intelligence and Neuroscience, 2022.
8. Abu-Jassar, A. T., & et al.. (2022). Electronic user authentication key for access to HMI/SCADA via unsecured internet networks. Computational Intelligence and Neuroscience, 2022.

ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023

9. Nevliudov, I., & et al.. (2020). Development of a cyber design modeling declarative Language for cyber physical production systems. J. Math. Comput. Sci., 11(1), 520-542.

10. Baker, J. H., & et al.. (2021). Some interesting features of semantic model in Robotic Science. SSRG International Journal of Engineering Trends and Technology, 69(7), 38-44.

11. Abu-Jassar, A. T., & et al.. (2021). Some Features of Classifiers Implementation for Object Recognition in Specialized Computer systems. TEM Journal: Technology, Education, Management, Informatics, 10(4), 1645-1654.

12. Nevliudov, I., & et al.. (2020). Method of Algorithms for Cyber-Physical Production Systems Functioning Synthesis. International Journal of Emerging Trends in Engineering Research, 8(10), 7465-7473.

13. Al-Sharo, Y. M., & et al.. (2021). Neural Networks As A Tool For Pattern Recognition of Fasteners. International Journal of Engineering Trends and Technology, 69(10), 151-160.

14. Sotnik, S., & et al.. (2020). Some features of route planning as the basis in a mobile robot. International Journal of Emerging Trends in Engineering Research, 8(5), 2074-2079.

15. Nathanael R. Weidler, & et al.. (2017). Return-Oriented Programming on a Cortex-M Processor. In 2017 IEEE Trustcom/BigDataSE/ICESS. Sydney, NSW, Australia.

16. Per Lindgren, & et al. (2016). Abstract timers and their implementation onto the ARM Cortex-M family of MCUs. ACM SIGBED Review, 13(1), 48-53.

17. Mohammad Hossein Askari Hemmat & et al. (2016). owards code generation for ARM Cortex-M MCUs from SysML activity diagrams. In 2016 IEEE International Symposium on Circuits and Systems (ISCAS). Conference Location: Montreal, QC, Canada.

18. Tomáš Jakubík. (2020). Cortex-M Simulator. In 2020 International Conference on Applied Electronics (AE). Conference Location: Pilsen, Czech Republic.

19. Lucan Orășan, & et al.. (2022). A Brief Review of Deep Neural Network Implementations for ARM Cortex-M Processor. Electronics, 11(16), 2545.

20. Amar A. Rasheed, & et al.. (2021). Clock Gating-Assisted Malware (CGAM): Leveraging Clock Gating On ARM Cortex M For Attacking Subsystems Availability. In 2021 9th International Symposium on Digital Forensics and Security (ISDFS), Conference Location: Elazig, Turkey.

21. Trevor Martin (2023). The Designer's Guide to the Cortex-M Processor Family. Elseveir Ltd, 604.

22. Amin, M.S., Rahman, S. (2023). An Introduction of Open System Interconnection (OSI) Model and its Architecture. Preprints 2023, 2023051858.

23. Gede Bagus Wirawan, & et al.. (2023). IoT based anti covid visitor management system using Raspberry pi zero W. AIP Conf. Proc. 2482, 100010.

24. Ortega, Alberto, & et al.. (2023). Design of a Standard and Programmatically Accessible Interface for Smart Meters to Allow Monitoring Automation of the Energy Consumed by the Execution of Computer Software. Sustainability, 15(3), 1900.

25. Liu, Jian, & et al.. (2022). Contour Resampling-Based Garlic Clove Bud Orientation Recognition for High-Speed Precision Seeding. Agriculture, 12(9), 1334.

ISSN: 2582-4686 SJIF 2021-3.261,SJIF 2022-2.889
ResearchBib Impact Factor: 8.848 / 2023

26. Banik, S., & Zimmer, V. (2022). System Firmware Debugging. In: Firmware Development. Apress, Berkeley, CA.

27. David Llanio Reyes, & et al.. (2023). Anomaly Detection in Embedded Devices Through Hardware Introspection. In 2023 Silicon Valley Cybersecurity Conference (SVCC). Conference Location: San Jose, CA, USA.

28. Igor Nevliudov, & et al.. (2021). Automation of Mathematical Modeling of Physical and Technological Processes in the Electronic Devices Manufacture. Proceedings of the XII International Scientific Conference «Functional Basis of Nanoelectronics» – Odessa, September 20-24, 2021, 74-77.

29. Igor Nevliudov, & et al.. (2022). The Use of Neural Networks for the Technological Objects Recognition Tasks in Computer-Integrated Manufacturing. 2022 IEEE 4th International Conference on Modern Electrical and Energy System (MEES), Kremenchuk, Ukraine, 2022, 1-5.