Zuiev, Andrii; Krylova, Viktoriia; Hapon, Anatolii et al.

Article

Research of microprocessor device and software for remote control of a robotic system

Technology audit and production reserves

UDC 004.75, 004.318
DOI: 10.15587/2706-5448.2024.297339

Andrii Zuiev,
Viktoriia Krylova,
Anatolii Hapon,
Stanislav Honcharov

RESEARCH OF MICROPROCESSOR DEVICE AND SOFTWARE FOR REMOTE CONTROL OF A ROBOTIC SYSTEM

The modern stage of the development of intelligent robotic systems is characterized by the expansion of fields of application, which is due to autonomous work and decision-making in conditions of uncertainty. The object of research is the system of remote control of robotic systems. During the remote control of robotic systems, problems arise that are associated with the use of wireless communication in real time. The article analyzes software and hardware implementations of various remote control systems suitable for use as part of autonomous robotic systems and analyzes promising microcontroller platforms for implementing a remote control device for a robotic system. A brief review of existing protocols for transmitting control signals using radio communication equipment and microprocessor platforms for the development of embedded systems is performed, among which a solution is selected for research. Several approaches to the control of a robotic system are highlighted – control using a wired connection and corresponding protocols, control via wireless communication or via the Internet, control via general-purpose network protocols. The target platform is chosen and justified, and the S.BUS protocol is analyzed with the provision of an algorithm for obtaining the values of the control channels from the S.BUS package. The structure and algorithm of functioning of the microprocessor remote control system based on the ESP32 microcontroller and the FreeRTOS OS are given. A study of the operation process of the proposed remote control system is carried out, for which it is placed on the chassis of a ground autonomous robotic system with four-wheel drive, and the delay time of the control signal from the receiver to the engine control modules is determined. According to the conducted analysis, the expediency of using specialized radio communication equipment with the S.BUS protocol for controlling executive devices as part of a robotic system, for precise movement control in real time, is shown.

Keywords: *robotic systems, IoT, microcontrollers, wireless communication channels, remote control, S.BUS, ESP32, FreeRTOS.*

Received date: 08.12.2023
Accepted date: 18.01.2024
Published date: 29.01.2024

How to cite
Zuiev, A., Krylova, V., Hapon, A., Honcharov, S. (2024). Research of microprocessor device and software for remote control of a robotic system. Technology Audit and Production Reserves, 1 (2 (75)), 31–37. doi: https://doi.org/10.15587/2706-5448.2024.297339

1. Introduction

Thanks to the development and application of autonomous robotic technologies, daily monitoring operations of industrial facilities, in particular power lines (power lines) and energy infrastructure facilities, are gradually being carried out with the help of robotic systems (RTS), which are equipped with various sensors, daylight cameras and infrared, lidars, etc. But some situations still require and will require the intervention of a human operator or manual work [1, 2]. It can be an operating robot that is located on the power line and is remotely controlled in the process of performing the task of cleaning insulators, tightening bolts, etc., or a sapper robot that neutralizes an explosive device, or a robot that welds industrial structures. This paper considers the issue of remote control of an operating robot through a wireless communication system in real time. The development and spread of embedded systems, such as, for example, Internet of Things (IoT) devices, ensures their deep penetration into various spheres of human activity: agriculture, household and communal spheres, industry, medicine. According to estimates by leading experts in the field [2, 3], by the beginning of the 20s, the number of smart devices organized into a single network around the world should reach almost 30 billion, and in areas where devices are networked and interact between without human intervention, more than 15 billion units. There is a practice of using traditional, well-proven IoT tools, which have a low price, to solve tasks in other industries, for example, for automation devices of monitoring and control systems.

Rapid prototyping using existing commercial components is a key aspect for developing innovative robots and systems [3]. Modular approaches are common for software development, requiring the design of a system framework that is based on a high-quality, real-time architecture and uses off-the-shelf basic modules (e. g., sensors, actuators, and controllers), integrating hardware and software. When designing any autonomous RTS, it is also necessary to ensure that the communication paths to all major subsystems use a reliable communication and control protocol.

The aim of research is the development and research of the software and hardware part of the microprocessor

ISSN 2664-9969

device for remote control of RTS using a wireless communication channel in real time using commercial components of IoT systems.

2. Materials and Methods

Recently, many review works have been published that consider the use of various protocols and methods of remote control of robots, RTS and autonomous systems.

Work [4] focuses on the creation of an automated robotics module used both offline and online, provides a common interface and architecture that can process data from virtual and physical sensors, and provides autonomous operation at the sensor level. A theoretical example of the use of arc welding with seam tracking for RTS is given. In work [5], issues of real-time control of RTS, which are equipped with sensors for monitoring collisions, are also considered. This document provides a robot control protocol that provides an interface to standard ABB S4 robot controllers and can be used to remotely control the robot. The results are currently being used in ongoing welding experiments. But the works actually do not consider practical issues of control protocols and application of the given RTS model, only issues of modeling and virtual simulation.

In [6], an approach to controlling robots via the Internet in real time is considered. Transport Protocol (RTP) is used as the communication protocol instead of the traditional use of TCP and UDP. Theoretical analysis, simulation, and experimental research were conducted to assess the feasibility and effectiveness of the proposed approach for practical use. But no way to solve the problem of non-guaranteed sequence of arrival of packets, which is inherent in the RTP protocol, is given. Also, there are no solutions to the problem associated with the variable delay of the arrival time of packets in wireless Wi-Fi networks, only it is noted that it can reach 120 ms.

The work [7] summarizes the requirements for communication systems in RTS and shows the limits of existing communication protocols. A description of the CAN-Bus protocol is given, which works in real-time and combines the advantages of different communication planning approaches. RTCAN takes into account time-triggered communication received from control loops, ensuring temporal determinism, as well as event-triggered communication using sensors, which is transmitted with low latency. Also given are the results of tests performed on the hardware, which demonstrate the functionality of this protocol. But it should be noted that this protocol is implemented only with the help of wired communication systems and has a very limited application for control systems of autonomous robots.

In [8], a controller for a robotic arm created using Internet of Things (IoT) technologies is considered. Such a robotic hand can be controlled via the Internet. MK Raspberry Pi is used as a controller, as well as for the operation of the web server system. All four servo motors can be individually controlled using pulse width modulation (PWM) signals. In addition, there is the ability to track and control the direction of the robotic arm, as well as perform pick and place tasks similar to the manufacturing industry. The results of this study are confirmed by practical testing. But the work also does not consider the issues of sequence and delay in the arrival of control packets, and communication reliability.

The paper [9] describes the implementation of a decentralized architecture for autonomous groups of air and ground RTS engaged in joint actions. The system provides transparent integration of information from various sensors. The information-theoretic measure of usefulness, which captures the purpose of the task and the interdependence of the robot, is considered. A distributed decision mechanism is used to determine trajectories and assignments taking into account the limitations of individual vehicles and sensor subsystems. The results of experiments on autonomous vehicles equipped with cameras are given. A radio modem with an extended spectrum is used to control and communicate with RTS agents, vehicles and the base station communicate via a Wi-Fi network. The paper also does not consider the technical points related to the control protocol of individual parts of the RTS, as well as the features of the software implementation of the control algorithms.

Thus, several approaches to RTS control can be identified:
– control using a wired connection and appropriate protocols (usually used for stationary industrial robots);
– wireless or Internet control, for example, control over general-purpose network protocols. Special mention should be made of the specialized control protocols that are becoming increasingly common these days, and which are used in commercial control systems for drones or autonomous robots.

3. Results and Discussion

3.1. RTS remote control protocols. All protocols can be conditionally divided into 3 groups:

1. Protocols that provide communication between a radio transmitter and a radio receiver.
2. Communication protocols between the radio receiver and the RTS controller.
3. Communication protocols between the RTS controller and executive mechanisms.

Usually, for autonomous RTS, the communication between the radio transmitter and the receiver is wireless (group 1 protocols). Most manufacturers of radio transmitters for control systems have their own protocols, or use open source radio systems such as ExpressLRS. Some protocols (such as DSM2) are immune to noise, interference, and other transmitters, or can use a backup frequency in case of a failure to transmit commands on the primary frequency. This significantly reduces the chance of signal loss. To implement radio communication, it is advisable to use commercial devices, for example, control panels from FPV drones of various manufacturers (Futaba, Frsky, Flysky, Spektrum, JR), which have a low price and considerable reliability.

The protocols of the second group, unlike the wireless communication between the transmitter and the receiver, use a wired transmission channel. One of the most important properties of any control protocol is the signal delay, which is essentially the time it takes for the receiver to convert the signal from the transmitter into the signal it is going to send to the RTS controller. A lower delay means that the RTS will respond more quickly to the instructions of the operator who controls it, and, as a result, to an increase in control accuracy and a decrease in the probability of emergency situations when it comes to autonomous robots. Some protocols of the second group are universal and used in receivers from different manufacturers, but some may be proprietary. The most common protocols are: universal (PWM, PPM/CPPM, Mavlink), as well as S.BUS (Futaba, Frsky), IBUS (Flysky), XBUS (JR),

FPort (Frsky), MSP (Multiwii), CRSF (ExpressLRS, TBS Crossfire and Tracer).

1) PWM – Pulse Width Modulation, the oldest control protocol that can be used directly to control servos and motors if there is no motor control controller in the RTS. The protocol requires three wires for control and power on one channel. The length of the pulse determines the output signal of the servo or throttle position. The length of the signal pulse usually ranges from 1000 μs to 2000 μs. The disadvantage of this protocol is the need for each signal to have a separate signal wire.

2) PPM – Pulse Position Modulation, also known as CPPM or PPMSUM. A PPM signal is a series of PWM signals sent one after the other on the same signal wire but modulated differently. The advantage of PPM over PWM is that only one wire is required for multiple channels, which greatly reduces the number of wires. But since the values on the communication channel do not arrive at the same time, it is not as accurate as PWM.

But serial protocols that appeared with the development of digital communication equipment are more modern. Serial protocol is a digital interference-tolerant protocol that also uses only 3 wires (signal, power, ground) for multiple channels (up to 16). Unlike PPM, which is a time-domain signal, serial protocols are completely digital. Usually, to receive serial protocols in the RTS controller, it is necessary a serial port to which the receiver is connected.

3) S.BUS or Serial BUS, commonly used in Futaba and FrSky equipment. It supports up to 16 channels and uses only one signal wire. The S.BUS signal is usually connected to the RX UART pin in the controller, this signal is inverted, and therefore the controller must have a dedicated input with an inverter for such a signal. Some MKs (STM32 F3 and F7, ESP32) have built-in inverters on all UART lines, so it is possible to connect S.BUS to any UART. It should also be noted that the CPPM and PWM signals have a delay of about 60–80 ms, while the S.BUS is only 10–20 ms.

4) IBUS, XBUS are serial protocols of Flysky and JR companies and are actually a copy of S.BUS. The latter can transmit only 14 channels, but theoretically has the lowest signal delay among serial protocols. Both protocols support two-way communication.

5) CRSF is a protocol developed by Team Black Sheep (TBS) for their Crossfire RC system. It is also very similar to S.BUS and other digital protocols in terms of coding. The main advantages of the protocol include low latency and two-way communication capabilities. With only four wires, it provides not only control, but also telemetry transmission.

6) MSP (Multiwii Serial Protocol) is created as part of the Multiwii software. It basically allows to send MSP commands to the controller input, which is usually done during the initial setup of the controller via USB, and supports 8 channels in one signal wire.

7) FPort is a protocol developed by Frsky and Betaflight for controlling drones. Normally, the control signal and telemetry data require separate connections, but FPort combines them into one bidirectional signal, making it more compact and easier to configure. Unlike S.BUS (Frsky), which is inverted, FPort is UART compatible without additional inverters.

8) MAVLINK is a telemetry protocol similar to FPort developed by the Pixhawk/ArduPilot community that provides two-way communication between the controller and the receiver.

Group 3 protocols provide communication between the controller and the executive devices (motors or servos) through a wired communication channel, and in fact they set the speed and direction of rotation of the motors. Their feature is that the controller must control the motors at a higher speed than the receiver receives commands, because in addition to receiving control commands, the RTS controller usually constantly receives a lot of data from various sensors, such as gyroscope and accelerometer, at a much higher speed (e. g., from 2 to 8 thousand times per second), which are used to correct the error of control or hold position and direction. The number of protocols of group 3 is small, these are: PWM, Oneshot, Multishot, Proshot and Dshot. It should be noted that DShot is a two-way protocol, in which not only motor control commands are issued from the controller, but it also tells the controller how fast the motors work (RPM), this data is used in feedback. S.BUS (FrSky) and PWM protocols for motor control will be considered for research into the RTS microprocessor control tool.

3.2. Selection and justification of the target platform. An analysis of promising microcontroller (MC) platforms for the implementation of the RTS remote control device was carried out (Table 1). Among all the candidates, development kits based on the MK ESP32 – which is a multifunctional system on a crystal, developed by Espressif Systems – a Chinese company based in Shanghai, drew special attention. ESP32 is positioned as an autonomous solution for the organization of Wi-Fi wireless networks, which can organize the connection of any third-party MK device to Wi-Fi, and is also able to run programs autonomously [10].

Table 1

Summary characteristics of the most popular platforms for embedded systems

Device	CPU	RAM	ROM (Flash)	Wi-Fi	Input/output lines	OS	Cost
Arduino	8–20 MHz	1–8 Kb	16–256 Kb	–	14–54	n/d	6–25 USD
ESP8266	80 MHz	80 Kb	512 Kb	+	9	FreeRTOS	3–12 USD
ESP32	160 MHz	512 Kb	4–16 Kb	+	43	FreeRTOS	5–15 USD
RPi Pico (W)	133 MHz	256 Kb	2–16 Mb	– (+)	26	n/d	6–12 USD
Omega2	580 MHz	64 Kb	16 Mb	+	12	Linux	22 USD
RPi Zero (W)	1 GHz	512 Mb	16–32 Gb	– (+)	40	Linux	15–20 USD
C.H.I.P.	1 GHz	512 Mb	4–8 Gb	+	45	Linux	50 USD

ISSN 2664-9969

The positive features of this platform include a low price and high power: the MK has 3 cores, 2 of which operate at a frequency of up to 240 MHz [11], and many different hardware modules for input and output, including UART and PWM, as well as Wi-Fi and Bluetooth modules, which opens up great opportunities for the application of this platform for the manufacture of various IoT devices [12] and automation systems in general. The ESP32 is supported by various popular development environments (IDEs) such as Arduino, PlatformIO, or can be programmed using the ESP-IDF framework from MK.

The ESP32 platform is used in various industries to receive and process data from sensors [13], to control solar panels and irrigation systems [14], for smart home systems and air quality control systems [15, 16].

Thus, this MK can be used as a controller of the remote control system, having connected to it a radio receiver that works according to the serial S.BUS protocol and engine control modules.

Let's take a closer look at what the S.BUS protocol packet looks like, which the receiver sends to the controller. The length of an S.BUS packet is 25 bytes, and each channel is 11 bytes. The total number of channels is 16. The starting byte 11110002=(240), which comes as MSB (that is, the most significant bit comes first) – on an ESP32-type MK should be checked as 000011112=(15). The last byte is 000000002 or xxxx01002. The penultimate byte (23) contains additional binary flags: 17 and 18 channels in the first and second bits, respectively, a lost frame and signal loss indication, in the third and fourth bits, respectively.

To obtain channel values, after receiving a packet and checking the first and last bytes, it is necessary to decode as follows (bytes are received in MSB format, but assembled as LSB):

$$
\begin{aligned}
&ch1 = b1 | ((b2 << 8) \,\&\, 0x07FF),\\
&ch2 = (b2 >> 3) | ((b3 << 5) \,\&\, 0x07FF),\\
&ch3 = (b3 >> 6) | (b4 << 2) | ((b5 << 10) \,\&\, 0x07FF),\\
&ch4 = (b5 >> 1) | ((b6 << 7) \,\&\, 0x07FF),\\
&ch5 = (b6 >> 4) | ((b7 << 4) \,\&\, 0x07FF),\\
&ch6 = (b7 >> 7) | (b8 << 1) | ((b9 << 9) \,\&\, 0x07FF),\\
&ch7 = (b9 >> 2) | ((b10 << 6) \,\&\, 0x07FF),\\
&ch8 = (b10 >> 5) | ((b11 << 3) \,\&\, 0x07FF),\\
&ch9 = b12 | ((b13 << 8) \,\&\, 0x07FF),\\
&ch10 = (b13 >> 3) | ((b14 << 5) \,\&\, 0x07FF),\\
&ch11 = (b14 >> 6) | (b15 << 2) | ((b16 << 10) \,\&\, 0x07FF),\\
&ch12 = (b16 >> 1) | ((b17 << 7) \,\&\, 0x07FF),\\
&ch13 = (b17 >> 4) | ((b18 << 4) \,\&\, 0x07FF),\\
&ch14 = (b18 >> 7) | (b19 << 1) | ((b20 << 9) \,\&\, 0x07FF),\\
&ch15 = (b20 >> 2) | ((b21 << 6) \,\&\, 0x07FF),\\
&ch16 = (b21 >> 5) | ((b22 << 3) \,\&\, 0x07FF),
\end{aligned}
$$

where *chn* – channel *n*, *bi* – *i*-th byte from the packet, & – bitwise AND operation, | – bitwise OR operation, >> – bitwise shift operation to the right, << – bitwise shift operation to the left.

The data transfer speed of the protocol reaches 100 Kbit/s, and is non-standard for UART. Other UART configuration parameters: parity – even, stop bits: 2, data bits: 8. The signal is inverted, which means that some ICs, such as Arduino, cannot accept this signal. A simple transistor and resistor inverter circuit can be used to invert the signal. But on most modern MKs, including ESP32, it is possible to configure the UART to read an inverted signal.

3.3. Features of implementation and choice of programming language. The process of controlling an autonomous RTS is complicated for many reasons, the main of which is the minimization of the delay in the arrival of the control signal from the remote control to the executing devices. Some MCs have advanced capabilities for parallelism, have a mechanism for changing the operating frequency: they purposefully become slower or faster, depending on the need for calculations. In the absence of load or when executing certain commands, they can significantly reduce the operating frequency, which reduces energy consumption and extends the possible time of operation from a limited power source.

Another important aspect is the choice of programming language for implementing the program on MC. Typically, this choice is driven by support for supporting hardware and peripherals such as:

– general purpose signal input/output interface (GPIO);
– analog input lines and analog-digital signal converters (ADC);
– wired digital interfaces (UART, SPI, I2C) and communication modules (Wi-Fi, Bluetooth), which are required to solve the problem. Availability of signal output hardware (PWM, RMT). MC developers almost always suggest using certain libraries of abstract representation of hardware resources (HAL) to access certain registers and peripherals for some programming languages (mostly C/C++). Another aspect is the support of one or another high-level language. Very often, too complex capabilities of a supported high-level programming language are limited or not implemented at all in the compiler version for certain hardware platforms. It is also important to consider the compiler and tools that will be used to compile the code. For embedded environments, the ability to optimize not only the performance of the compiled application, but also the size of the code that implements it is very important.

The code is executed under the control of the operating system, which also performs some other tasks, such as handling the events of transceivers such as Wi-Fi or Bluetooth modules. All this can lead to variability in the execution time of the same code.

The most expedient for the implementation of remote control algorithms in accordance with the specified conditions will be the choice of C/C++ programming languages. Development in these programming languages can be done for most platforms, they provide direct control over memory allocation, and these languages have the necessary libraries to support hardware and peripherals. In comparison, the Python language (MicroPython) does not provide control over memory allocation, and has somewhat poor performance, and the Go language (TinyGo) also lacks support for wireless peripherals (Wi-Fi and Bluetooth) on the ESP32 platform.

3.4. Practical implementation and research of the RTS control system on the ESP32 platform. FrSky R9 Mini 900 MHz was used as the receiver, which was connected to UART1 MK. FrSky Taranis X-Lite 2.4GHz ACCST Radio 16CH was used as the control panel. To control the engines, a bridge circuit module was used on the L298N for each pair of engines, to ensure the possibility of reversing the autonomous RTS.

At startup, the driver configures the UART to the following settings: 8 data bits, even, 2 stop bits, 100 Kbit/s, and sets the UART interrupt handler, which retrieves

messages from the OS queue. It also sets the flags for inverting the signals coming to the UART from the receiver.

After that, the configuration and setting of PWM hardware modules is carried out using the ESP-IDF ledc library. A separate timer is selected for each pair of control signals (for one motor), which allows each motor to be controlled independently (2 KHz PWM frequency, 10-bit timer resolution). The calculation of PWM splicing depending on the desired speed of rotation p (in percent) is calculated by the expression:

$$d = \frac{p \cdot d_{\max}}{100}, \quad (1)$$

where $d_{\max} = 2^r - 1$ – maximum value of duty cycle; $r = 10$ – PWM timer resolution.

Software-wise, each motor is a separate object with two PWM phases that are linked by a separate timer for clocking. This object has an interface that allows to set the speed and direction of rotation or disable the motor. An OS queue is connected to each object, with the help of which control commands are received, which allows to significantly reduce the reaction time in the engine control chain, due to the parallel processing of messages on different computing cores of the MC.

After receiving and checking the next control packet, which arrived via the S.BUS protocol, the value of the control channels is read: movement speed, lateral displacement, rotation and settings (direction of movement, engine blocking, etc.). The received values are sent to the RTS control routine, which decomposes them into commands for controlling individual engines, which in turn are sent to the engine control modules. The time of successful arrival of a control packet is memorized, and if no packets are received for some time, a forced command to stop the engines is issued, and the movement of the RTS is stopped until the control signals are restored. The structure of the remote control system is shown in Fig. 1.

To reduce the delay time, the PWM setpoints are cached and if the change from the next signal is less than 1 %, the motor control signal remains unchanged.

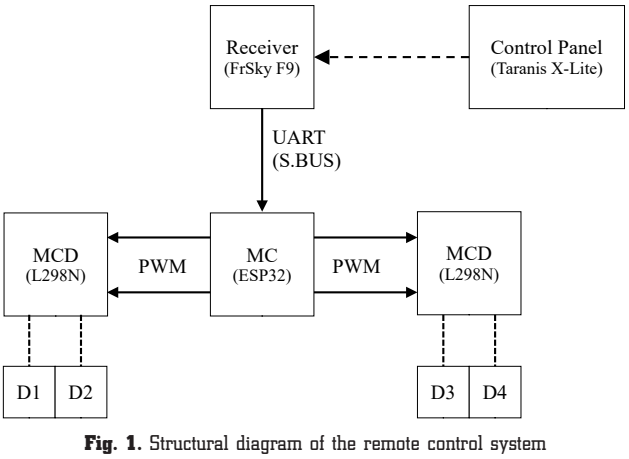

Fig. 1. Structural diagram of the remote control system

3.5. System efficiency analysis and conclusions. The MCU platform ESP32-WROOM v3.1 is chosen for the implementation of the control system: processor frequency 160 MHz, software environment ESP-IDF 5.01.0001. The software implementation of the considered control algorithms is implemented using the C++ language. The program was compiled using the GCC compiler with settings for performance (Optimize for performance-O2). The program was downloaded to the flash memory of the MK and executed under the control of the OS FreeRTOS [17] with a quantization time of 1 ms.

A study of the functioning of the proposed remote control system was carried out, for which it was placed on the chassis of a ground autonomous RTS with four-wheel drive. Steering was carried out by all wheels, regardless of the possibility of reversing, which, in combination with a special chassis, provides significant mobility of the RTS, including the possibility of lateral displacement without rotation.

In the course of the study, the time of processing control signals and their decomposition by PWM channels was measured (Table 2). Time was measured using the MC's internal counter (esp_timer_get_time() function) with a resolution of up to 1 μs.

Table 2

Calculation time for the algorithm for decomposing control commands on PWM channels (ESP32, 1 core, 160 MHz)

No.	State	Calculation time min, μs	Calculation time max, μs
1	No control (Disable)	3	3
2	Rotation	20	40
3	Side shift	25	45
4	Change of speed or direction of movement	10	15

Table 3 and Fig. 2 show the measurements of the delay time of the control signals on separate PWM channels in the mode of parallel and serial (without the use of a queue) processing at different frequencies of the processor. The delay time was measured at the limit of simplification of command queues for all motor controllers.

Also, the software implementation of the algorithm was additionally tested at a frequency of 240 MHz, from which it was found that the delay time decreases by ~1.3–1.4 times, which is practically proportional to the increase in frequency. From the obtained results, it can be seen that the delay in the case of parallel processing when using the OS queue is reduced by 1.7–2.8 times, compared to the sequential implementation of the algorithm.

Table 3

Comparison of the delay time for parallel and serial implementations of the algorithm for sending signals through PWM channels depending on the frequency of the processor

No.	State	Delay time (160 MHz), μs		Delay time (240 MHz), μs	
		min	max	min	max
1	Processing and recording	3	3	2	2
	Parallel sending of a PWM signal (queue)				
2	Change of speed or direction of movement, or lateral shift (4 channels)	40	46	29	32
3	Rotation (2 channels)	24	30	19	23
	Sequential sending of a PWM signal (no queuing)				
4	Change of speed or direction of movement, or lateral shift (4 channels)	66	130	49	73
5	Rotation (2 channels)	91	104	48	60

ISSN 2664-9969

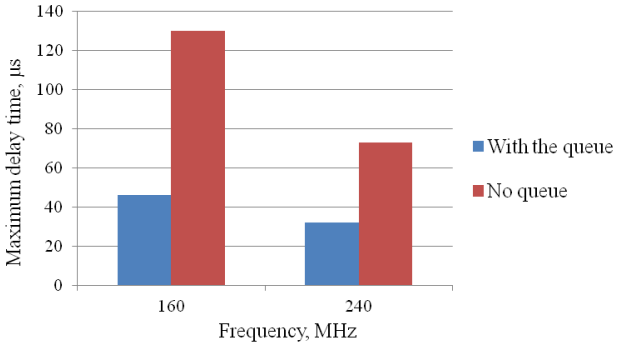

Fig. 2. Comparison of the maximum delay time for different implementations of the algorithm for sending signals through PWM channels depending on the frequency of the processor

3.6. Discussion. The results obtained in the work can be applied to the development and improvement of remote control systems for robotics and drones. Also, the proposed system can be used as part of simulation training complexes for the training of operators of robotic equipment, for which the MC, which is part of the system, must be directly connected to the control computer of the training complex using the UART/USB interface. This will ensure the uniformity of remote control and the possibility of a seamless transition from training to operation of the tool.

The proposed system uses only commercial components of general purpose with low cost, which reduces the cost of both the robotic complex and the simulation-exercise complex. In the conditions of martial law in Ukraine, the low cost and availability of the components that make up the system give significant advantages to the proposed solution.

Further research may focus on closer integration of the control system with simulators and support for digital motor control protocols, including brushless ones such as DShot.

4. Conclusions

From the obtained results, it can be seen that both implementations of the algorithm provide a negligible delay compared to the signal transcoding delay, which is tens of milliseconds. Also, the total delay time obtained is significantly less than the delays that are present when exchanging wireless communication channels via Wi-Fi or Bluetooth networks. Thus, it is possible to note the expediency of using specialized radio communication equipment with the S.BUS protocol to control the executive devices in the RTS, for accurate traffic control in real time.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

Financing

The research was performed without financial support.

Data availability

The manuscript has no associated data.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

References

1. Chen, L., Dong, X., Sun, Y., Su, Z. (2019). Real-time Image Transmission and Operation Control for Power Transmission Line Patrol using Unmanned Aerial Vehicle. *25th International Conference on Electricity Distribution. Madrid, 3–6 June 2019. Paper No. 488.* doi: https://doi.org/10.34890/79
2. Shafique, K., Khawaja, B. A., Sabir, F., Qazi, S., Mustaqim, M. (2020). Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios. *IEEE Access, 8,* 23022–23040. doi: https://doi.org/10.1109/access.2020.2970118
3. Bonarini, A., Matteucci, M., Migliavacca, M., Rizzi, D. (2014). R2P: An open source hardware and software modular approach to robot prototyping. *Robotics and Autonomous Systems, 62 (7),* 1073–1084. doi: https://doi.org/10.1016/j.robot.2013.08.009
4. Cederberg, P., Olsson, M., Bolmsjö, G. (2002). Virtual triangulation sensor development, behavior simulation and CAR integration applied to robotic arc-welding. *Journal of Intelligent and Robotic Systems, 35,* 365–379. doi: https://doi.org/10.1023/a:1022306821640
5. Bolmsjö, G., Cederberg, P., Olsson, M. (2002). Remote Control of a Standard ABB Robot System in Real Time Using the Robot Application Protocol (RAP). *Proceedings of the 33rd ISR (International Symposium on Robotics).* Stockholm.
6. Duong, P. M., Hoang, T. T., Vinh, T. Q. (2010). Control of an Internet-based Robot System Using the Real-time Transport Protocol. *Proc of the 5th Vietnam Conference on Mechatronics.* doi: https://doi.org/10.48550/arXiv.1707.05456
7. Migliavacca, M., Bonarini, A., Matteucci, M. (2013). RTCAN: a Real-Time CAN-Bus Protocol for Robotic Applications. *Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics (ICINCO-2013),* 353–360. doi: https://doi.org/10.5220/0004484303530360
8. Ishak, K. A., Ishak, M. K., Roslan, M. I. (2018). Design of Robotic Arm Controller based on Internet of Things (IoT). *Journal of Telecommunication, Electronic and Computer Engineering, 10 (2-3),* 5–9.
9. Grocholsky, B., Bayraktar, S., Kumar, V., Taylor, C. J., Pappas, G.; Ang, M. H., Khatib, O. (Eds.) (2006). Synergies in Feature Localization by Air-Ground Robot Teams. Experimental Robotics IX. *Springer Tracts in Advanced Robotics. Vol. 21.* Berlin, Heidelberg: Springer, 352–361. doi: https://doi.org/10.1007/1155224634
10. *ESP32 Technical Reference Manual. Version 5.0 Espressif Systems* (2023). Available at: https://www.espressif.com/sites/default/files/documentation/esp32technical_reference_manual_en.pdf
11. *ESP32 Series Datasheet. Version 4.3 Espressif Systems* (2023). Available at: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
12. Maier, A., Sharp, A., Vagapov, Y. (2017). Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. *2017 Internet Technologies and Applications (ITA).* IEEE: Piscataway, 143–148. doi: https://doi.org/10.1109/itecha.2017.8101926
13. Hangan, A., Chiru, C.-G., Arsene, D., Czako, Z., Lisman, D. F., Mocanu, M., Pahontu, B., Predescu, A., Sebestyen, G. (2022). Advanced Techniques for Monitoring and Management of Urban Water Infrastructures – An Overview. *Water, 14 (14),* 2174. doi: https://doi.org/10.3390/w14142174
14. Allafi, I., Iqbal, T. (2017). Design and implementation of a low cost web server using ESP32 for real-time photovoltaic system monitoring. *2017 IEEE Electrical Power and Energy Conference (EPEC).* IEEE: Piscataway. doi: https://doi.org/10.1109/epec.2017.8286184
15. Carducci, C. G. C., Monti, A., Schraven, M. H., Schumacher, M., Mueller, D. (2019). Enabling ESP32-based IoT Applications in Building Automation Systems. *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT).* IEEE: Piscataway, 306–311. doi: https://doi.org/10.1109/metroi4.2019.8792852

16. Taştan, M., Gökozan, H. (2019). Real-Time Monitoring of Indoor Air Quality with Internet of Things-Based E-Nose. *Applied Sciences, 9 (16)*, 3435. doi: https://doi.org/10.3390/app9163435
17. *FreeRTOS*. Available at: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html Last accessed: 15.11.2022

Andrii Zuiev, *PhD, Associate Professor, Department of Automation and Control in Technical Systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine, ORCID: https://orcid.org/0000-0001-8206-4304*

✉***Viktoriia Krylova,*** *PhD, Associate Professor, Department of Automation and Control in Technical Systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine, e-mail: viktoriia.krylova@khpi.edu.ua, ORCID: https://orcid.org/0000-0002-4540-8670*

Anatolii Hapon, *PhD, Associate Professor, Department of Automation and Control in Technical Systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine, ORCID: https://orcid.org/0000-0002-2582-6154*

Stanislav Honcharov, *Postgraduate Student, Department of Automation and Control in Technical Systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine, ORCID: https://orcid.org/0009-0002-9874-3189*

✉*Corresponding author*

UDC 658.51
DOI: 10.15587/2706-5448.2024.297399

Oleksandr Malyi,
Nataliia Furmanova,
Vadym Onyshchenko,
Iryna Pospeieva,
Pavlo Kostianoi

ANALYSIS OF EXPERIENCE IN OPTIMIZING THE OPERATION OF AN AUTOMATED PRODUCTION LINE FOR FOLDING CARDBOARD BOXES

The object of research is an automated system for controlling the bending mechanisms of the folding-gluing line for cardboard packaging products. In the work, the ways of optimization and the development of an automated control system for the bending mechanisms of the folding and gluing line, which makes it possible to fold various structures and standard sizes of cardboard boxes using modern automation tools based on programmable logic controllers, were carried out in the work.

A mathematical model has been proposed to describe operations on the folding-gluing line. Based on the model, a methodology has been developed for calculating the parameters of the automated control system for the box folding production line, depending on the technical parameters of the line and the parameters of the boxes to be bent. This will make it possible to choose optimal technological modes of the production process and obtain high quality parameters of cardboard packaging products. To verify the mathematical model of the production line, software has been developed in the Delphi development environment, which was applied with the developed automated production process control system based on a programmable logic controller (PLC) and an operator panel. This produced a number of results, in particular, controlling the speed of the hook bend in proportion to the speed of the line. And also provided an opportunity to increase line speed and, accordingly, production productivity. It has been possible to change the mutual location of system elements, which made it possible to reduce the distance between workpieces and increase the capacity of the line tape by 30 %.

The obtained research results were implemented at the production enterprise of typographic products «Dinas» (Zaporizhzhia, Ukraine), which contributed to the improvement and optimization of production processes. Conducting further research will provide an opportunity to expand the proposed methodology for use on all types of folding and gluing lines.

Keywords: *automated system, technological process, structural diagram, programmable logic controller, panel-controller.*

Received date: 11.12.2023
Accepted date: 23.01.2024
Published date: 29.01.2024

How to cite
Malyi, O., Furmanova, N., Onyshchenko, V., Pospeieva, I., Kostianoi, P. (2024). Analysis of experience in optimizing the operation of an automated production line for folding cardboard boxes. Technology Audit and Production Reserves, 1 (2 (75)), 37–45. doi: https://doi.org/10.15587/2706-5448.2024.297399

1. Introduction

Cardboard is often used as packaging material for various products. It has a low cost, a wide range of standard sizes, design and assembly variability. Dense paper packaging simply folds, saves space during storage, and is easily disposed of. At the same time, the boxes are quite strong, which led to their use in various industries, from the food