1. Grzegorz LENTKA[1], 2. Dariusz PALMOWSKI[1], 3. Maciej BRZESKI[1]

Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki (1)
ORCID: 1. 0000-0003-1221-7357; 2. 0000-0002-2108-0869

doi:10.15199/48.2024.12.32

Pomiar parametrów czasowych wybranych mikrosystemów operacyjnych czasu rzeczywistego

Streszczenie. *W artykule przedstawiono porównanie czterech mikrosystemów operacyjnych czasu rzeczywistego: FreeRTOS, µC/OSIII, Nuttx i Zephyr Project pod kątem wybranych parametrów czasowych dla czterech platform uruchomieniowych opartych na mikrokontrolerach STMicroelectronics: NUCLEOL073RZ, NUCLEOF103RB, NUCLEOF411CE i NUCLEOH743ZI. Opracowana metoda pomiaru i zaproponowane scenariusze testowe umożliwiają ocenę parametrów czasowych mikrosystemów operacyjnych czasu rzeczywistego.*

Abstract*. This paper presents a comparison of four real-time operating microsystems: FreeRTOS, µC/OSIII, Nuttx and Zephyr Project in terms of selected timing parameters for four development platforms based on STM32 microcontrollers: NUCLEOL073RZ, NUCLEOF103RB, NUCLEOF411CE and NUCLEOH743ZI. The developed measurement method and proposed test scenarios enable the measurement of selected timing parameters of real-time operating microsystems. (**Measurement of timing parameters of selected real-time operating microsystems**).*

Słowa kluczowe: systemy wbudowane, mikrosystemy operacyjne czasu rzeczywistego, pomiar parametrów czasowych.
Keywords: embedded systems, real-time operating microsystems, timing parameters measurement.

Wstęp

Systemy wbudowane (ang. Embedded Systems) stały się nieodzownym składnikiem dzisiejszego świata, będąc sercem wielu urządzeń elektronicznych. Występują w praktycznie każdym sektorze gospodarki i przemysłu, automatyzując i monitorując procesy przemysłowe, tworząc sieci sensorowe i łącząc się w ramach Internetu Rzeczy. Zróżnicowane zastosowania systemów wbudowanych, takie jak jednostki centralne pojazdów, przyrządy pomiarowe, urządzenia medyczne, elektronika konsumencka, czy interaktywne budki informacyjne, łączy wspólne wymaganie, jakim jest reagowanie na zdarzenia w czasie rzeczywistym, bez którego utraciłyby sens. Zadaniem postawionym przed projektantami takich systemów jest zatem spełnienie ściśle określonych wymagań czasowych związanych z obsługą tychże zdarzeń. Z pomocą przychodzą systemy operacyjne czasu rzeczywistego, pozwalając zrealizować system, który szybko i optymalnie reaguje na żądania.

Stosowanie systemów operacyjnych czasu rzeczywistego znacząco ułatwia i przyspiesza proces tworzenia oprogramowania systemów wbudowanych, a także zwiększa bezpieczeństwo aplikacji poprzez zmniejszenie odpowiedzialności projektanta za poprawne zarządzanie zadaniami. Niestety zwykle wiąże się to z dodatkowymi opóźnieniami, w których trakcie następuje np. przełączanie zadań, bądź zarządzanie dostępem do zasobów. Jednym z założeń projektowych takich systemów jest zminimalizowanie tego niezbędnego naddatku czasowego. Poszukując systemu operacyjnego, często brakuje wykazu dokładnych parametrów czasowych dla typowych zastosowań. Ich znajomość umożliwiłaby projektantom łatwe porównanie ze sobą różnych rozwiązań i podjęcie decyzji. Autorzy inspirowali się pracą [1], jednakże rozszerzyli zakres badanych parametrów i testowanych platform sprzętowych, przy czym zakres tematyczny ograniczono do mikrosystemów operacyjnych czasu rzeczywistego, a więc takich, które działają na platformach sprzętowych o ograniczonych zasobach obliczeniowych i pamięciowych (np. mikrokontrolerach).

Dobór platform sprzętowych i mikrosystemów operacyjnych czasu rzeczywistego

Kierując się rankingiem [2] postanowiono skupić się na produktach STMicroelectronics, a do dalszych prac wytypować następujące mikrokontrolery STM32 z rdzeniami ARM Cortex-M (montowane na uniwersalnych platformach uruchomieniowych z serii Nucleo):

- STM32L073RZ, rdzeń ARM CortexM0+ taktowanie max. 32 MHz, 192 KB pamięci Flash, 20 KB SRAM i 6 KB EEPROM - płytka ewaluacyjna NUCLEOL073RZ [3];
- STM32F103RB, rdzeń ARM CortexM3 taktowanie max. 72 MHz, 128 KB pamięci Flash, 20 KB SRAM - płytka ewaluacyjna NUCLEOF103RB [4];
- STM32F411CE, rdzeń ARM CortexM4, koprocesor zmiennoprzecinkowy, taktowanie max. 100 MHz, 512 KB pamięci Flash, 128 KB SRAM - płytka ewaluacyjna KANUCLEOF411CEv2 [5];
- STM32H743ZI, rdzeń ARM CortexM7, koprocesor zmiennoprzecinkowy, taktowanie max. 480 MHz, 2 MB pamięci Flash i 1024 KB SRAM, - płytka ewaluacyjna NUCLEOH743ZI [6].

Wymienione platformy, wybrane do dalszych badań, zostały uporządkowane pod względem wydajności, poczynając od tych prostszych, przeznaczonych do mniej wymagających zastosowań, do najbardziej wydajnych przeznaczonych do aplikacji zużywających więcej mocy obliczeniowej.

W rankingu [2] wysoko uplasowały się również popularne platformy sprzętowe jak Raspberry PI oparte na procesorach z rdzeniem Cortex-A, stanowiące wysoce wydajną alternatywę dla mikrokontrolerów. Ich wadą jest konieczność użycia złożonego systemu operacyjnego, co wykracza poza zakres tego artykułu.

Drugim elementem planowania był dobór mikrosystemu operacyjnego - posiłkując się [2] oraz uwzględniając tylko mikrosystemy operacyjne dla mikrokontrolerów i unikając systemów dedykowanych dla określonego producenta lub typu mikrokontrolera wytypowano niżej wymienione systemy.

FreeRTOS to system czasu rzeczywistego dla mikrokontrolerów i małych mikroprocesorów. Projekt zapoczątkowała w 2003 roku firma Real Time Engineers, a w 2017 roku został wykupiony przez Amazon Web Services. Rozprzestrzeniany jest na licencji „MIT open source". Wersja wykorzystana do badań to v10.2.1 [7, 8].

µC/OS III jest trzecią generacją mikrosystemu operacyjnego rozwijanego przez firmę Micrium od 1992 roku [9]. Po przejęciu firmy Micrium przez firmę Silicon Labs, na początku 2020, system został udostępniony na wolnej licencji Apache, a kod źródłowy znajduje się na

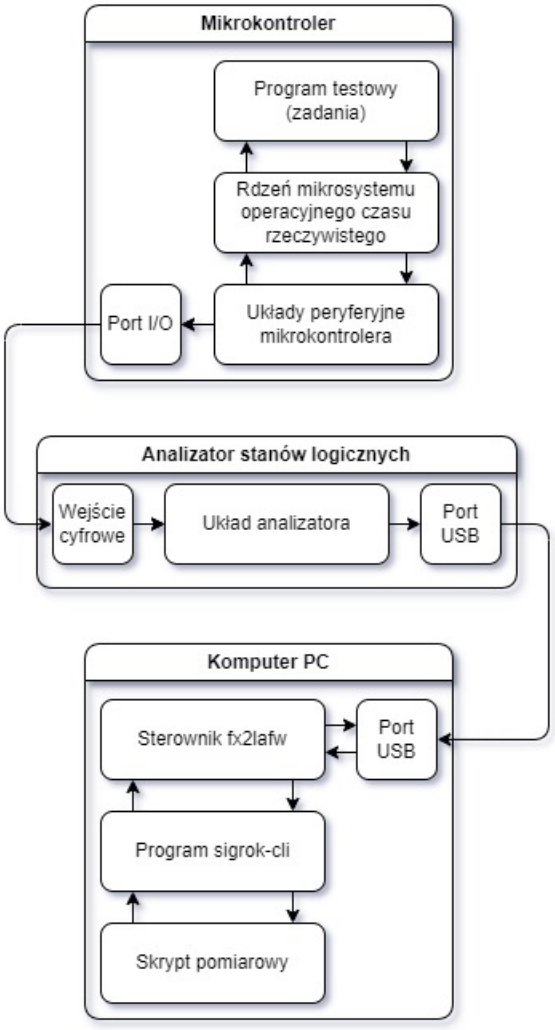
platformie GitHub. Wersja wykorzystana do badania to µC/OS III v3.8.1.

Apache Nuttx jest systemem operacyjnym opracowanym z naciskiem na zgodność ze standardami POSIX oraz ANSI C [10] i małą objętością kodu wynikowego. Po raz pierwszy zaprezentował go Gregory Nutt w 2007 roku, a od końca 2019 roku rozwijany przez The Apache Software Foundation. Dla potrzeb niniejszego artykułu wykorzystano wersję v10.1.0.

Zephyr Project to otwartoźródłowy system operacyjny czasu rzeczywistego, zoptymalizowany dla urządzeń o ograniczonych zasobach sprzętowych. Zaprezentowano go po raz pierwszy w 2016 roku - jest kontynuacją darmowego systemu Rocket od Wind River Systems - jest zarządzany przez Linux Foundation [11]. W badaniach wykorzystano wersję v2.6.0.

Stanowisko pomiarowe i scenariusze testowe

Na rysunku 1 przedstawiono schemat stanowiska pomiarowego wykorzystanego w badaniach.



Rys.1. Schemat stanowiska pomiarowego

Stanowisko składa się z mikrokontrolera z wgranym programem testowym (jedna z wymienionych wcześniej platform sprzętowych), analizatora stanów logicznych oraz komputera ogólnego przeznaczenia z oprogramowaniem pomiarowym. Program testowy służy wywoływaniu funkcji badanego systemu operacyjnego czasu rzeczywistego, w wyniku czego zmieniany jest stan logiczny końcówki wyjściowej, do której podłączony jest analizator. Sygnał ten zostaje przetworzony i przesłany przez USB do PC, gdzie następuje jego akwizycja i przetwarzanie.

Podstawowym testem przeprowadzonym dla wszystkich systemów było badanie obsługi (zmiany stanu) wyprowadzenia GPIO mikrokontrolera – dla określenia czasu niezbędnego na zmianę stanu końcówki mikrokontrolera w danym systemie – wartość ta będzie uwzględniana w dalszych testach, tak aby uwzględniać tylko czas interesującej operacji.

Ponadto przygotowano kilka scenariuszy testowych skupiając się na użyciu semaforów i kolejek komunikatów, ponieważ stanowią podstawowe narzędzie komunikacji międzyzadaniowej i są dostępne w każdym badanym systemie. Wśród przygotowanych scenariuszy warto wymienić:

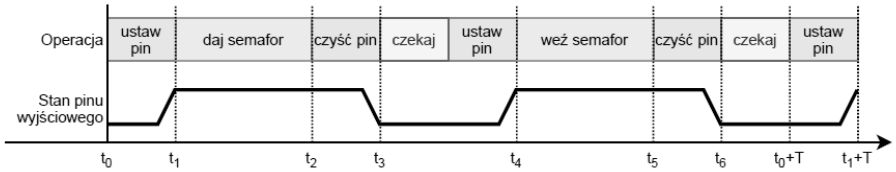
1. test obsługi semafora/kolejki – dla określenia czasu potrzebnego na elementarną obsługę semafora, czyli jego pozyskiwanie i zwalnianie lub kolejki, czyli wysłanie i odebranie komunikatu. W tym celu wywoływane po kolei zostaną obie te funkcjonalności na nowo utworzonym semaforze/kolejce, na które nie oczekuje żadne zadanie. Najpierw zostaje utworzony nowy semafor/nowa kolejka, następnie w nieskończonej pętli jest on na przemian zwalniany i pozyskiwany lub wysyłany i odbierany jest komunikat. Celem zmierzenia czasu trwania tych wywołań, otoczone są one funkcjami zmieniającymi stan końcówki wyjściowej. Schematyczny przebieg tego testu dla semafora przedstawiono na rysunku 2a. Przebieg testu dla kolejki wygląda identycznie.
2. Test wywłaszczania z wykorzystaniem semafora/kolejki:
   a) opóźnienie wywłaszczenia zadania o niższym priorytecie przez zadanie o wyższym priorytecie (rys. 2b) - celem jest zmierzenie czasu potrzebnego na wywłaszczenie zadania o niskim priorytecie przez zadanie o wyższym priorytecie z użyciem semafora/kolejki, które informują o zdarzeniu, na które ma zareagować system;
   b) opóźnienie powrotu z wywłaszczenia do zadania o niższym priorytecie z zadania o wyższym priorytecie (rys. 2c) - celem jest zmierzenie czasu potrzebnego na powrót z zadania o wyższym priorytecie do zadanie o niższym priorytecie (zakończenie wywłaszczenia).

Na rysunkach 2b i 2c pokazano przebieg scenariusza dla użycia semafora, przebieg dla kolejki wygląda identycznie.

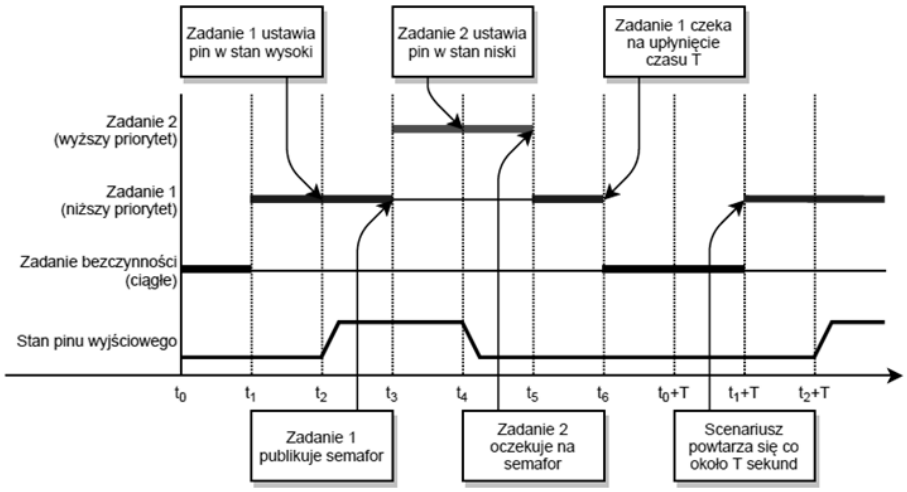
Warto tu nadmienić, że dla większości typowych mikrosystemów operacyjnych czasu rzeczywistego, wywłaszczenie jest jedną z podstawowych funkcjonalności systemu. Do wywłaszczenia dochodzi, gdy podczas wykonywania zadania o niskim priorytecie, następuje uaktywnienie zadania o wysokim priorytecie (z powodu zajścia zdarzeń w systemie, np. ustawienia semafora lub pojawienia się komunikatu w kolejce). W takim przypadku system operacyjny czasu rzeczywistego zatrzymuje wykonywanie zadania o niskim priorytecie (następuje wywłaszczenie zadania o niskim priorytecie), a przechodzi do wykonywania zadania o wysokim priorytecie. Scenariusz 2a pozwala przetestować czas od uaktywnienia zadania o wysokim priorytecie do przejścia do jego wykonania, w dwóch wariantach zależnych od tego co było powodem wywłaszczenia – albo semafor albo kolejka.

Po zakończeniu zadania o wysokim priorytecie system powraca do wykonywania zadania o niskim priorytecie – następuje powrót z wywłaszczenia. Scenariusz 2b pozwala przetestować czas tego powrotu również w 2 wariantach zależnych od przyczyny wywłaszczenia: semafora lub kolejki.

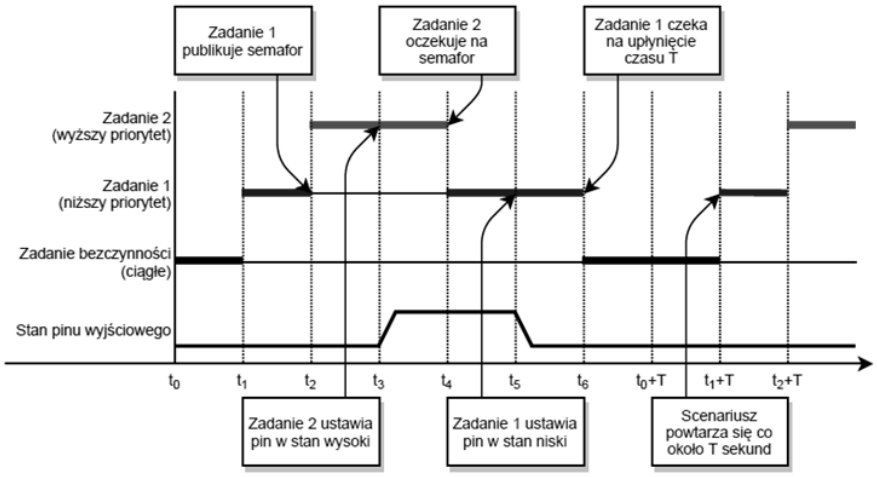
a)



b)



c)



Rys.2. Schemat zależności logicznych i czasowych podczas a) obsługi semafora, b) wywłaszczenia zadania o niższym priorytecie z użyciem semafora, c) powrotu z wywłaszczenia

Wyniki badań i podsumowanie

Przeprowadzono opisane wcześniej scenariusze testowe dla wszystkich 4 platform sprzętowych pod kontrolą wszystkich 4 mikrosystemów operacyjnych. Mikrokontrolery skonfigurowano tak, aby pracowały z relatywnie niskim zegarem 8 MHz. Wyniki podano albo jako wartość odpowiedniego czasu wyrażoną w µs albo po przeliczeniu na liczbę taktów zegara taktującego dany mikrokontroler (podaną bezwymiarowo) – rozwiązanie to pozwala również porównywać pomiary wykonywane dla różnych częstotliwości zegarów mikrokontrolerów.

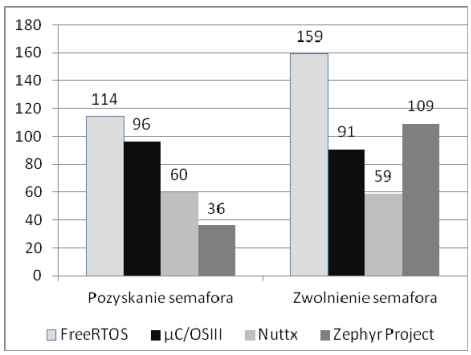
W tabeli 1 pokazano przykładowe wyniki pojedynczej serii pomiarów wykonanych dla konkretnej platformy sprzętowej (STM32L073RZ) i konkretnego systemu operacyjnego (FreeRTOS), analogiczne pomiary wykonywano dla wszystkich kombinacji.

Tabela 1. Przykładowe wyniki pojedynczej serii pomiarowej dla platformy uruchomieniowej STM32L073RZ i mikrosystemu operacyjnego FreeRTOS v10.2.1

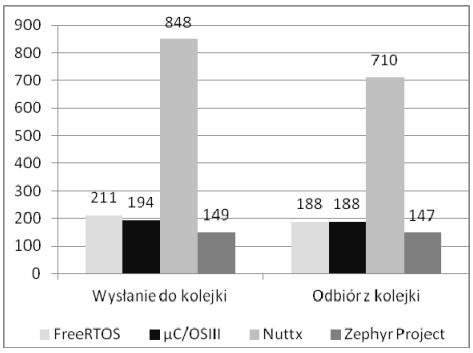
Parametr	Czas [µs]	Liczba cykli
Ustawienie GPIO w stan H	1,625	13
Ustawienie GPIO w stan L	1,625	13
Zwrot semafora	21,500	172
Pobranie semafora	15,875	127
Wysłanie komunikatu do kolejki	29,250	234
Odczyt komunikatu z kolejki	25,125	201
Wywłaszczenie z użyciem semafora	68,625	536
Powrót z wywłaszczenia z użyciem semafora	113,125	892
Wywłaszczenie z użyciem kolejki	85,750	673
Powrót z wywłaszczenia z użyciem kolejki	112,500	887

Na rysunku 3 pokazano zestawienie wyników dla testów semafora (rys. 3a) i kolejki komunikatów (rys. 3b) dla testowanych systemów na platformie STM32L073RZ.

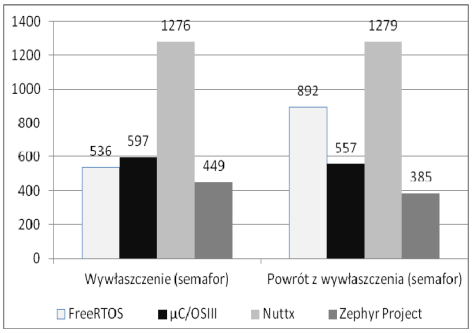
a)



b)



c)



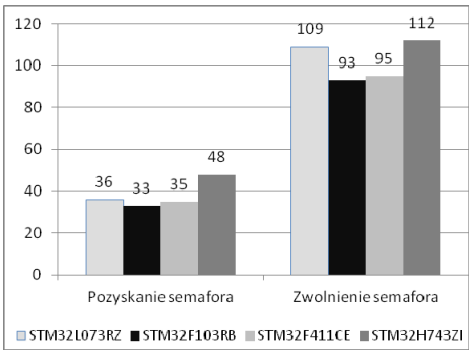
Rys.3. Czas obsługi (wyrażony w cyklach zegara) a) semafora, b) kolejki, c) wywłaszczenia z użyciem semafora w testowanych mikrosystemach operacyjnych czasu rzeczywistego

Testy wykazały, że najlepsze parametry czasowe spośród badanych mikrosystemów operacyjnych czasu rzeczywistego uzyskał Zephyr Project. Dobrym wynikiem badań wydajnościowych wykazał się również niegdyś komercyjny system µC/OSIII. Nieco gorsze parametry czasowe uzyskał system FreeRTOS, który stanowi najpopularniejszy mikrosystem operacyjny używany w systemach wbudowanych. Na ostatnim miejscu znalazł się Nuttx, który wbrew deklarowanej wysokiej wydajności, uzyskał znacznie gorsze parametry czasowe.

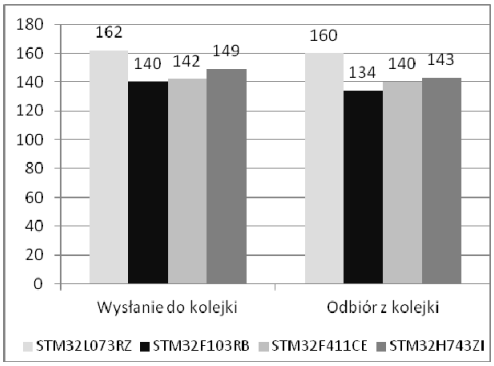
Na rysunku 4 pokazano wyniki testów semafora (rys. 4a) i kolejki komunikatów (rys. 4b) dla testowanych platform sprzętowych.

Porównując ze sobą wybrane platformy sprzętowe, dla mikrosystemu operacyjnego Zephyr Project wytypowanego jako optymalny w pierwszym etapie, uzyskiwane różnice nie były aż tak znaczące – jednak zadziwiająco dobrze wypadł najprostszy rdzeń Cortex-M0+ (z platformy STM32L073RZ) z najkrótszym potokiem, a stosunkowo najsłabiej wypadł najbardziej zaawansowany rdzeń Cortex-M7 (z platformy STM32H743ZI), który posiada najdłuższy potok instrukcji.

a)



b)



Rys.4. Czas obsługi (wyrażony w cyklach zegara) a) semafora, b) kolejki w systemie Zephyr Project dla wszystkich testowanych platform sprzętowych

***Autorzy**: dr hab. inż. **Grzegorz Lentka**, Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, Katedra Metrologii i Optoelektroniki, ul. Narutowicza 11/12, 80-233 Gdańsk, E-mail: grzegorz.lentka@pg.edu.pl;*
*mgr inż. **Dariusz Palmowski**, Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, Katedra Metrologii i Optoelektroniki, ul. Narutowicza 11/12, 80-233 Gdańsk, E-mail: dariusz.palmowski@pg.edu.pl;*
*mgr inż. **Maciej Brzeski**, Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, absolwent, ul. Narutowicza 11/12, 80-233 Gdańsk, E-mail: s165396@student.pg.edu.pl*

LITERATURA

[1] Ungurean I, Timing comparison of the realtime operating systems for small microcontrollers, *Symmetry*, 2020; 12(4):592. https://doi.org/10.3390/sym12040592
[2] AspenCore, 2019 Embedded Markets Study – Integrating IoT and Advanced Technology Designs, Application Development & Processing Environments. embedded.com, 2019.
[3] STMicroelectronics, Opis płytki ewaluacyjnej NUCLEOL073RZ. https://www.st.com/en/evaluationtools/nucleol073rz.html
[4] STMicroelectronics, Opis płytki ewaluacyjnej NUCLEOF103RB. https://www.st.com/en/evaluationtools/nucleof103rb.html
[5] Kamami, Opis płytki ewaluacyjnej KANUCLEOF411CEv2. http://wiki.kamamilabs.com/index.php/KANUCLEOF411CEv2_(PL)
[6] STMicroelectronics, Opis płytki ewaluacyjnej NUCLEOH743ZI. https://www.st.com/en/evaluationtools/nucleoh743zi.html
[7] Amazon Web Services, The FreeRTOS Reference Manual version 10.2.1, Amazon Inc., 2017.
[8] Barry R., Mastering the FreeRTOS™ Real Time Kernel. 2016.
[9] Labrosse J. J., µC/OS-III The Real-Time Kernel, Micriµm Press, 2011.
[10] Dokumentacja online systemu Nuttx, https://nuttx.apache.org/docs/latest/
[11] Strona główna systemu Zephyr Project. https://www.zephyrproject.org/