Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej Nr 55

XXI Międzynarodowe Seminarium Metrologów

MSM 2017

Rzeszów - Czerniowce, 12-15 września 2017

APLIKACJA MONITORUJĄCA I STERUJĄCA SYSTEMEM MIKROKONTROLEROWYM

Beata PAŁCZYŃSKA[1], **Dorota RABCZUK**[1], **Jarosław FORNALSKI**[2]

1. Katedra Telekomunikacji Morskiej, Wydział Elektryczny, Akademia Morska w Gdyni
   tel.: 58 5586552 e-mail: palbeata@am.gdynia.pl, dorota.rabczuk@we.am.gdynia.pl
2. Wydział Elektryczny, Akademia Morska w Gdyni, e-mail: fornal3@gmail.com

Streszczenie: W artykule zaprezentowano aplikację, zaprojektowaną w środowisku programistycznym LabVIEW, do monitorowania stanów wejść i sterowania wyjściami analogowymi i cyfrowymi systemu kontrolno-pomiarowego z mikrokontrolerem 8-bitowym. Aplikacja monitorująca składa się z szeregu przyrządów wirtualnych, których zadaniem jest przetwarzanie i interpretacja danych binarnych oraz bajtowych. Stworzone przyrządy wirtualne pozwalają na monitorowanie przykładowych reprezentatywnych urządzeń wejściowych (przełączniki, enkoder obrotowy, potencjometr, joystick, klawiatura analogowa, żyroskop) oraz sterowanie urządzeniami analogowymi i cyfrowymi na liniach wyjściowych mikrokontrolera (diody, wyświetlacz siedmio-segmentowy, silnik krokowy).

Słowa kluczowe: system kontrolno-pomiarowy, LabVIEW, przyrządy wirtualne, system mikrokontrolerowy

1. WPROWADZENIE

Systemy mikrokontrolerowe wyposażone w urządzenia wejściowe i wyjściowe na liniach GPIO (*ang. General Purpose Input Output*) wymagają narzędzi do monitorowania ich stanu. Do najwygodniejszych narzędzi należą tzw. programy terminalowe pozwalające na wyprowadzanie informacji tekstowych na port szeregowy. Istnieje szereg propozycji programów terminalowych np. Realterm [1], niektóre z nich są wbudowane w środowiska programistyczne np. Serial Monitor w Arduino IDE [2].

Wzbogaceniem wizualizacji w stosunku do informacji tekstowej jest graficzna prezentacja stanów urządzeń wejściowych i wyjściowych układu. Aplikacja użytkownika z możliwością monitorowania i sterowania jest naturalnym dopełnieniem systemu mikrokontrolerowego [3-7]. Można ją napisać np. w Microsoft Visual Studio [8] z wykorzystaniem komponentów do komunikacji w standardzie RS-232 – wymaga to jednak znajomości podstaw programowania w językach tekstowych. Wymagań tych nie stawia alternatywna, bardziej intuicyjna metoda projektowania graficznego, której przykładem jest środowisko programistyczne LabVIEW [9].

Prezentowany projekt ma charakter dydaktyczny. Stworzono w LabVIEW reprezentatywny wybór przyrządów wirtualnych do monitorowania i sterowania urządzeniami analogowymi i cyfrowymi. Przyrządy są łatwo skalowalne do potrzeb innych urządzeń typowo podłączanych w systemach mikrokontrolerowych.

2. TRANSMISJA SZEREGOWA Z KOMPUTEREM PC

Do przesyłania danych w obu kierunkach między aplikacją monitorującą a mikrokontrolerem wykorzystano transmisje szeregową i interfejs UART (*ang. Universal Asynchronous Receiver and Transmitter*).

Transmisja szeregowa między mikrokontrolerem a światem zewnętrznym odbywa się po dwóch liniach interfejsu UART w trybie asynchronicznym. Sygnał ten może być wprowadzony na łącze COM komputera po konwersji poziomów napięć do standardu RS232C lub podany na adapter USB, którego wyjście podłączone do portu USB komputera tworzy port wirtualny. Do obsługi transmisji UART standardowy mikrokontroler ma co najmniej dwa wektory przerwań. Wektor odbiorczy sygnalizuje odebranie jednego bajtu i stanowi wezwanie do odczytania go z bufora wejściowego, wektor nadawczy sygnalizuje zakończenie nadawania poprzedniego bajtu i stanowi zachętę do podania kolejnego bajtu do bufora wyjściowego. Wykorzystanie przerwań do obsługi transmisji szeregowej zleca wykonanie tej pracy do interfejsu i tym samym zwalnia jednostkę centralną z konieczności sprawdzania stanu modułu UART Oprogramowanie mikrokontrolera (na przykładzie ATMega328) polega na dokonaniu odpowiednich wpisów do rejestrów interfejsu UART – należy uzupełnić rejestr prędkości transmisji UBRR (*UART Baud Rate Register*), rejestr parametrów UCSR0C (*USART Control and Status Register C*), do którego należy wpisać m.in. liczbę bitów w znaku, liczbę bitów stopu, parzystość, oraz rejestr UCSR0B (*USART Control and Status Register B*), w którym włącza się funkcję nadawania i odbioru oraz przerwania interfejsu. Bufory wejściowy i wyjściowy o pojemności 1 bajtu każdy występują pod tą samą nazwą UDR0 (*UART Data Register 0*), mimo tego możliwa jest transmisja dupleksowa. Programowanie przez bezpośrednie wpisy do rejestrów mikrokontrolera (tab. 1) jest uniwersalne we wszystkich środowiskach akceptujących kontrolery ATMEL (np. Atmel Studio, Arduino IDE itp.) [10]. Przesyłane bajty muszą być analizowane pod kątem ich zawartości bitowej, aby zapewnić kontrolę czujników w układzie mikrokontrolerowym i sterowanie urządzeniami na liniach wyjściowych.

Tablica 1. Kod źródłowy programu transmisji szeregowej z komputerem PC przez wbudowany port UART

```
char f=48;

ISR(USART_RX_vect)
//alternatywna nazwa przerwania
//ISR(SIG_UART0_RECV)
{
//odbiór bajtu z portu UART
  int s=UDR0;
//przykład selekcji bitu sterującego nr 4 z odebranego bajtu
 int ster= s & 0b000010000;
if (ster == 0)
//wyłącz diode
else
//włącz diodę
}

ISR(USART_TX_vect)
//alternatywna nazwa przerwania
//ISR(SIG_UART0_RECV)
{
//przykład tworzenia bajtu do wysłania ze stanów (0 lub 1) 8-miu czujników
f = (czuj7<<7)| (czuj6<<6)| (czuj5<<5)| (czuj4<<4)| (czuj3<<3)| (czuj2<<2)| (czuj1<<1)| (czuj0);
//kolejne nadania – pierwsze jest w main()
  UDR0 = f;
}

int main (void)
{
#define FOSC 16000000 //częstotliwość kwarcu taktującego
#define BAUD 9600 //bitów na sekundę
#define ubrr FOSC/16/BAUD-1
//rejestr prędkości transmisji jest 16-bitowy
  UBRR0H = (ubrr >> 8);
  UBRR0L = ubrr;
//włączenie nadajnika i odbiornika UART oraz przerwań nadawczego i odbiorczego
  UCSR0B = (1 << RXEN0) | (1 << TXEN0)|(1 << RXCIE0) | (1 << TXCIE0);
//8n1 bez parzystości
  UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
//nadanie bajtu f – pierwsze nadanie po starcie programu
  UDR0 = f;
  sei();
  while(1){};
}
```

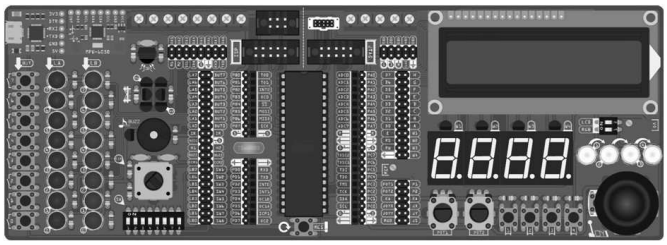
3. ZAPROJEKTOWANE PRZYRZĄDY WIRTUALNE W LabVIEW

Środowisko programistyczne LabVIEW wykorzystano do zaprojektowania aplikacji, umożliwiającej wizualizację stanów na liniach GPIO mikrokontrolerowego układu kontrolno-pomiarowego, stanowiącego testowo-dydaktyczną platformę laboratoryjną (rys. 1) [11]. Układ ten został wyposażony w czujniki wejściowe cyfrowe i analogowe oraz urządzenia wyjściowe sterowane analogowo i cyfrowo.

Dobór urządzeń umożliwia pokazanie różnych sposobów interpretowania danych w postaci bitów i bajtów, w szczególności:

- ✓ oddzielna interpretacja każdego bitu na przykładzie pojedynczych diod i przełączników na liniach GPIO (każdy bit reprezentuje jedno urządzenie), enkodera obrotowego (jego stan jest określany przez odczyt dwóch bitów), wyświetlaczy siedmiosegmentowych (bit wyboru wyświetlacza oraz osiem bitów sterujących segmentami wyświetlacza),
- ✓ interpretacja bajtu rozumianego jako wartość analogowa podana w rozdzielczości 10 lub 8-bitowej na przykładzie odczytu potencjometru, joysticka (połączenie dwóch potencjometrów podające wychylenia w płaszczyźnie XY), klawiatury analogowej, termometru, żyroskopu i akcelerometru (położenie i prędkość w przestrzeni są podawane w trzech bajtach w odniesieniu do trzech osi XYZ),
- ✓ sterowanie silnikiem krokowym za pomocą przebiegu cyfrowego o regulowanej częstotliwości, generowanego na wyprowadzeniu cyfrowym mikrokontrolera,
- ✓ sterowanie diodami RGB za pomocą bajtów przekazujących wartości analogowe w rozdzielczości 10 lub 8-bitowej.

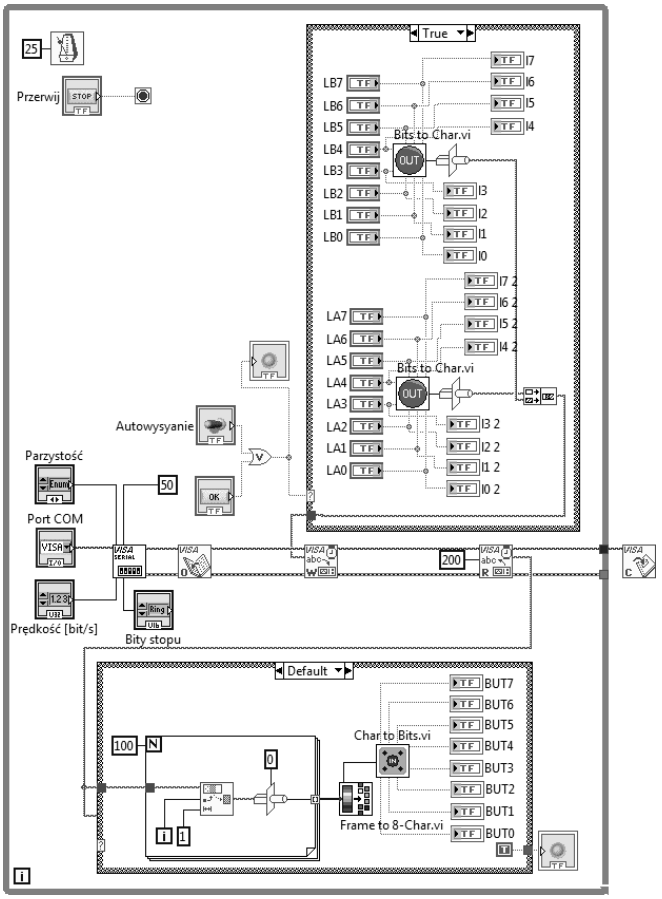
W graficznym środowisku programistycznym LabVIEW podstawowymi programami, realizującymi komunikację komputera z mikrokontrolerowym układem kontrolno-pomiarowym za pomocą standardowych interfejsów są przyrządy wirtualne VI (*ang. Virtual Instrument*) ze standardu bibliotecznego VISA (*ang. Virtual Instrument Software Architecture*), dostępne z poziomu diagramu projektowanej aplikacji w LabVIEW [12]. Służą one do konfigurowania portów komputera oraz komunikowania się z zewnętrznymi urządzeniami i stanowią szkielet prezentowanej aplikacji. Biblioteka VISA zapewnia niezależność interfejsową. Większość jej funkcji wykonuje swoje specyficzne zadania niezależnie od typu interfejsu.



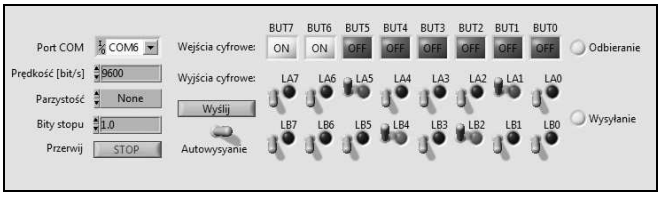
Rys. 1. Model wektorowy mikrokontrolerowego układu kontrolno-pomiarowego [11]

3.1. Wizualizacja stanu wejść i sterowanie wyjściami cyfrowymi

Na bazie biblioteki VISA zaprojektowano przyrząd *InOut.vi* do nadawania i odbioru bajtów służący do komunikacji z układem kontrolno-pomiarowym (rys. 2). Dane przychodzące w bajtach z czujników cyfrowych wymagają w programie rozdzielenia na bity, natomiast rozkazy sterujące dla urządzeń wyjściowych wydawane binarnie w aplikacji monitorującej wymagają przetworzenia na bajty przed przesłaniem do układu kontrolno-pomiarowego. Funkcje te realizują w przyrządzie wirtualnym *InOut.vi* podprogramy: *Bits to char.vi*, *Char to Bits.vi* oraz *Frame to 8-char.vi.* Na diagramie przyrządu wirtualnego można wyróżnić dwie struktury. Struktura wejściowa, wizualizująca położenie przycisków, dołączona jest do przyrządu *VISA Read Function.vi.* Natomiast tunel wyjściowy struktury sterującej zapalaniem diod połączony jest z terminalem wejściowym przyrządu *VISA Write Function.vi.* Panel frontowy przyrządu wirtualnego *InOut.vi* (rys. 3) powiązany przez komponenty graficzne języka tworzy aplikację wygodną dla użytkownika do kontroli i sterowania urządzeniami fizycznymi układu.



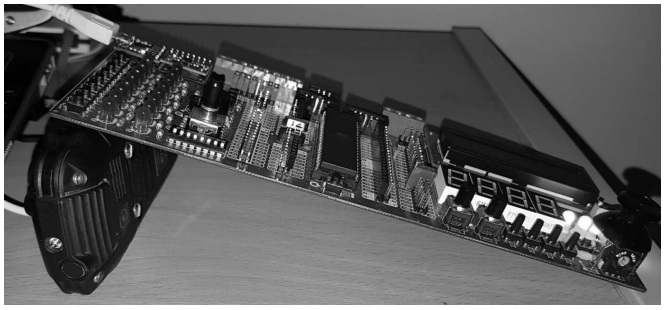
Rys. 2. Diagram terminalu *InOut.vi* [13]



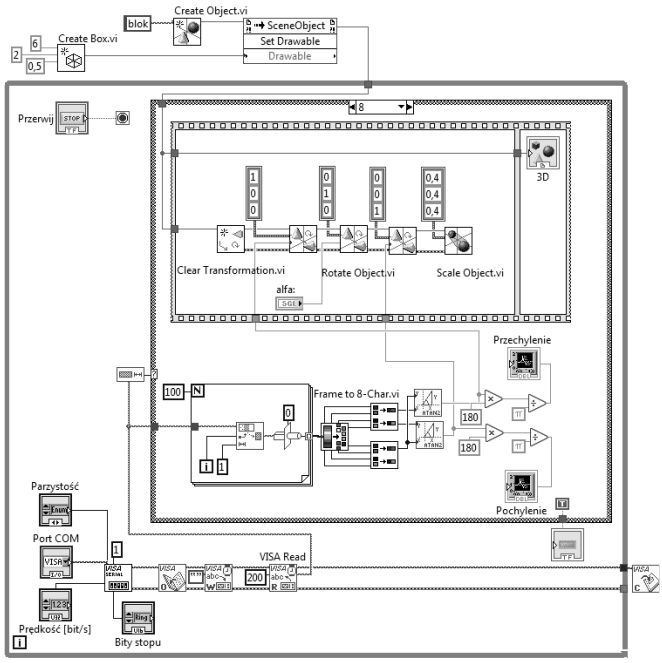
Rys. 3. Panel użytkownika terminalu *InOut.vi* [13]

3.2. Wizualizacja stanu wejść analogowych

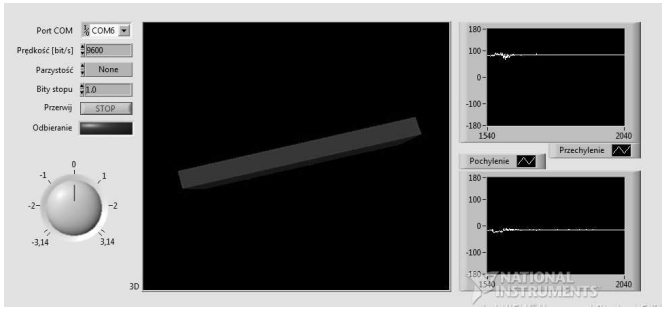
Przyrząd wirtualny *Mpu.vi* wizualizujący położenie płytki ewaluacyjnej w przestrzeni (rys. 4) jest przykładem aplikacji do monitorowania stanu wejść analogowych. Dane, które określają położenia płytki w przestrzeni mierzone są przez żyroskop umieszczony w przestrzeni zewnętrznej mikrokontrolera na magistrali I^2C. W programie *Mpu.vi* obsługa danych znajduje się w strukturze dołączonej do przyrządu *VISA Read Function.vi* (rys. 5).



Rys. 4. Pozycja płytki ewaluacyjnej w trakcie testowania przyrządu do wizualizacji wejść analogowych [13]



Rys. 5. Diagram terminalu *Mpu.vi* [13]



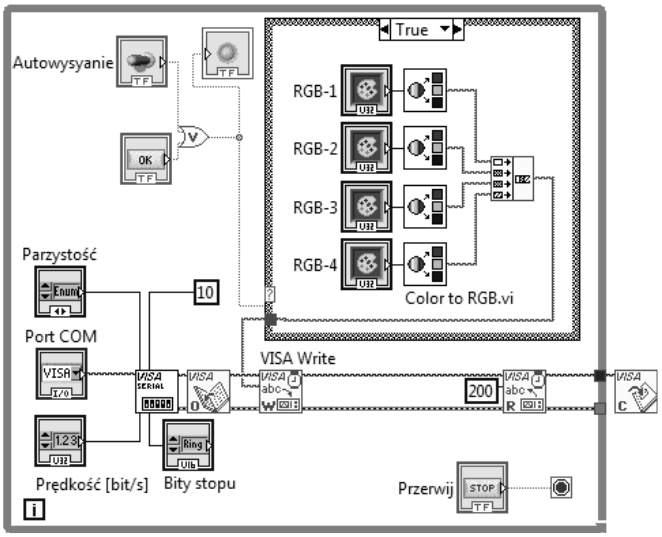
Rys. 6. Panel użytkownika terminalu *Mpu.vi* [13]

Poza strukturą *While Loop* programu znajdują się przyrządy *Create Object.vi*, które tworzą model obiektu w przestrzeni, przypominający płytkę ewaluacyjną. Podprogramy *Rotate Object.vi* odtwarzają obrót wirtualnego obiektu na podstawie danych odebranych z mikrokontrolera. Następnie, obiekt jest wyświetlany w przestrzeni trójwymiarowej. Na panelu frontowym przyrządu wirtualnego znajdują się kontrolki, które umożliwiają ustawienie parametrów komunikacji, pokrętło obrotu obiektu względem osi Y oraz indykatory graficzne w postaci wykresu trójwymiarowego, na którym wizualizowany jest obiekt i dwóch wykresów dwuwymiarowych przedstawiających kąty pochylenia i przechylenia obiektu (rys. 6).

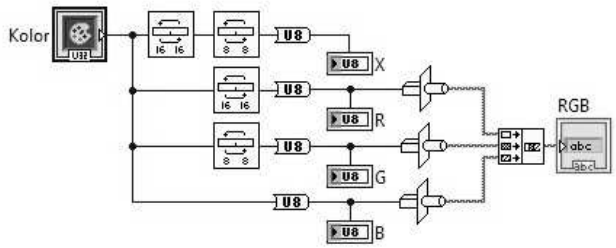
3.3. Sterowanie wyjściami analogowymi

Przyrząd wirtualny *RgbApa.vi* steruje kolorem czterech diod RGB, które są umieszczone w układzie kontrolno-pomiarowym. W kodzie źródłowym programu istotną rolę odgrywa przyrząd *VISA Write Function.vi*, za pośrednictwem, którego dane, określające intensywność barw każdej z diod są wysyłane do mikrokontrolera (rys. 7). Podprogram *Color to RGB.vi* (rys. 8) zamienia kolor, wprowadzony na kontrolce panelu frontowego przyrządu na ciąg trzech zmiennych odpowiadających intensywności barw czerwonej, zielonej i niebieskiej. Zmienne wyjściowe czterech podprogramów, reprezentujące kolor każdej z diod zgrupowane wysyłane są przez port szeregowy. W ramach graficznej aplikacji użytkownik oprócz parametrów

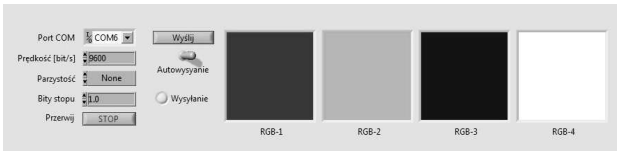
transmisji ma możliwość ustawienia automatycznej opcji wysyłania danych, co pozwala zmieniać kolory diod w czasie rzeczywistym (rys. 9).



Rys. 7. Diagram terminalu *RgbApa.vi* [13]



Rys. 8. Diagram podprogramu *Color to RGB.vi* [13]



Rys. 9. Panel użytkownika terminalu *RgbApa.vi* [13]

4. UWAGI KOŃCOWE

Zrealizowany projekt stanowi w pełni funkcjonalne stanowisko laboratoryjne w laboratorium systemów kontrolno-pomiarowych opartym na środowisku LabVIEW.

Skalowalność projektu daje możliwość dołączania dalszych urządzeń analogowych i cyfrowych do linii mikrokontrolera i powiększenie bazy przyrządów wirtualnych.

W bardziej zaawansowanej wersji można przewidzieć rozbudowę układu z mikrokontrolerem o sprzętowy stos TCP/IP i transmisję danych między aplikacją monitorującą a układem po łączu Ethernet.

5. BIBLIOGRAFIA

1. Realterm serial terminal, https://realterm.sourceforge.io, (dostęp 28.06.2017).
2. Arduino – Serial, https://www.arduino.cc/en/, (dostęp 28.06.2017).
3. Avallone E., Cunha DG., Padilha A., Scalon V. L.: Electronic multiplex system using the Arduino platform to control and record the data of the temperatures profiles in heat storage tank for solar collector, International Journal of Energy and Environmental Engineering, vol. 7 (4), 2016, pp. 391-398.
4. Davitadze Z., Partenadze G., Djincharadze E.: Graphical Visualization of Data Measurement of Programmable Microcontroller According to ARDUINO-project Example, IEEE East-West Design and Test Symposium (EWDTS), Yerevan 2016, pp. 1 – 5.
5. Yakimov P. I.: Open-source Platforms Application in Introductory Embedded Systems Teaching, 25th International Scientific Conference on Electronics (ET), Sofia 2016, pp. 1-4.
6. Torroja Y., Lopez A., Portilla J., Riesgo T.: A Serial Port Based Debugging Tool to Improve Learning with Arduino, Conference on Design of Circuits and Integrated Systems (DCIS), Estoril 2015, pp. 1 - 4.
7. Chęciński J., Filus Z.: Mikrokontroler w roli sterownika PWM przetwornicy impulsowej, Przegląd Elektrotechniczny, tom 86, nr 11a, 2010, ss.169-172.
8. Visual Studio IDE, https://www.visualstudio.com/pl/vs/, (dostęp 28.06.2017).
9. LabVIEW National Instruments, www.ni.com/labview, (dostęp 28.06.2017).
10. Rabczuk D.: Praktyczne nauczanie systemów wbudowanych z wykorzystaniem platformy Arduino, Zeszyty Naukowe Akademii Morskiej w Gdyni, nr 90, Gdynia 2015, s. 122-129.
11. Świtalski E.: Wykonanie testowo-dydaktycznej platformy laboratoryjnej z mikrokontrolerem 8-bitowym, Praca dyplomowa inżynierska, Akademia Morska Gdynia, Wydział Elektryczny, Gdynia 2016 r.
12. NI VISA Universal I/O Interface Software, http://sine.ni.com/nips/cds/view/p/lang/pl/nid/12145, (dostęp 28.06.2017).
13. Fornalski J.: Wizualizacja stanów linii GPIO mikroprocesorowego układu uruchomieniowego, Praca dyplomowa inżynierska, Akademia Morska Gdynia, Wydział Elektryczny, Gdynia 2017 r.

THE APPLICATION FOR MONITORING AND CONTROLLING A MICROCONTROLLER SYSTEM

In the paper, the application designed in LabVIEW for monitoring of analog and digital inputs and for controlling analog and digital outputs of an 8-bit microcontroller system is presented. The application consists of a series of virtual instruments aimed at processing and interpreting binary and byte data. The created virtual instruments allow to monitor various representative input devices (switches, rotary encoder, potentiometer, joystick, analog keyboard, gyroscope) and control digital and analog output devices of the microcontroller system (diodes, stepper motor, seven segment display).

Keywords: monitoring and control system, LabVIEW, virtual instruments, microcontroller system.