Wstęp

Książkę napisano jako podręcznik do przedmiotu SMS *Systemy Mikroprocesorowe w Sterowaniu*. Autor zakłada, że Czytelnik zna podstawy automatyki i posiada umiejętność programowania w języku C na poziomie średnio zaawansowanym. W książce opisane są układy regulacji obiektów liniowych, niezmiennych w czasie (*Linear Time-Invariant*) LTI z jednym wejściem i jednym wyjściem (*Single Input Single Output*) SISO.

Zalety regulacji cyfrowej

Praktycznie, wszystkie współczesne regulatory to systemy cyfrowe. Można je budować jako cyfrowe przybliżenie regulatorów analogowych, ale nie pozwala to w pełni wykorzystać możliwości jakie daje technika cyfrowa. Uzyskamy co najwyżej regulator prawie tak dobry, jak najlepszy analogowy. Dlatego warto poznać tajniki regulacji cyfrowej. Regulacja cyfrowa ma wiele wyraźnych zalet w stosunku do analogowej, co tłumaczy jej niezwykłą popularność:

- **Dokładność**. Sygnały są reprezentowane cyfrowo. Zastosowanie liczb zmiennopozycyjnych umożliwia przetwarzanie sygnałów z bardzo małymi błędami w porównaniu do sygnałów analogowych obciążonych szumem, dryftem temperaturowym i dryftem wynikającym ze starzenia elementów. Reprezentacja cyfrowa umożliwia także rejestrację (archiwizację) przebiegu regulacji i zdarzeń (alarmów) wewnątrz regulatora, co nie było możliwe w przypadku regulatorów analogowych.

- **Implementacja prawa regulacji**. Przetwarzanie cyfrowe umożliwia wykonywanie obliczeń (mnożenie, dzielenie, całkowanie, różniczkowanie, ...) z dokładnością nieporównywalnie większą niż w przypadku sygnałów analogowych. Regulatory cyfrowe umożliwiają realizację zaawansowanych, nieliniowych algorytmów predykcyjnych (np. opartych na sieciach neuronowych), których realizacja analogowa byłaby niemożliwa.

- **Elastyczność**. Regulator analogowy wymaga gruntownej przebudowy w celu zmiany prawa regulacji. W regulatorze cyfrowym, najczęściej wystarczy zmienić oprogramowanie, bez zmiany sprzętu. Możliwe jest także pobieranie ze strony producenta aktualizacji oraz wgrywanie różnych algorytmów (predykcyjnych, adaptacyjnych), które można dobrać do obiektu już w miejscu instalacji.

- **Zdalny nadzór i diagnostyka**. Regulator cyfrowy może być połączony z internetem, co pozwala na zbieranie danych o procesie w rozległych układach regulacji takich jak krajowa sieć gazowa, czy sieci wodociągowe. Takie połączenie ułatwia także pracę serwisu, który może na przykład określić, czy konieczny jest wyjazd, czy wystarczy interwencja na miejscu, personelu o niższych kwalifikacjach.

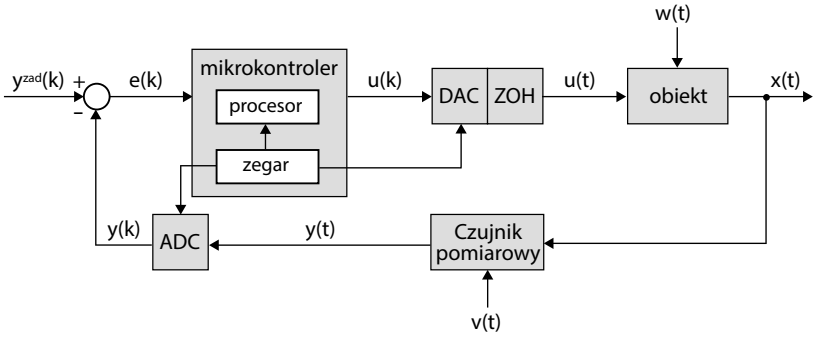
- **Szybkość działania**. Jeszcze w końcówce XX wieku poważną barierą we wprowadzaniu techniki mikroprocesorowej do regulacji przemysłowej była mała moc obliczeniowa mikroprocesorów w połączeniu z dużym poborem energii. Procesory o nieco większej mocy wymagały wymuszonego chłodzenia, co w warunkach przemysłowych często jest niemożliwe, ze względu na wymaganą niezawodność (awaryjność układów elektronicznych gwałtownie rośnie ze wzrostem ilości energii, którą trzeba rozproszyć). Zbyt mała moc obliczeniowa stanowiła barierę do stosowania zaawansowanych algorytmów obliczeniowych (np. wymagających rozwiązywania *on line* kwadratowego zadania optymalizacji) do szybkich procesów. Takie algorytmy praktycznie stosowane były wyłącznie do bardzo wolnych procesów chemicznych. W ostatnich latach nastąpił gwałtowny rozwój mikrokontrolerów o mocach wystarczających do realizacji zaawansowanych algorytmów regulacji większości procesów przemysłowych z tak małym poborem energii, że możliwe jest budowanie niezawodnych układów.

- **Niski koszt**. Ceny układów cyfrowych, także mikrokomputerów są obecnie tak niskie, że są one powszechnie stosowane nawet w najtańszych i najprostszych regulatorach.

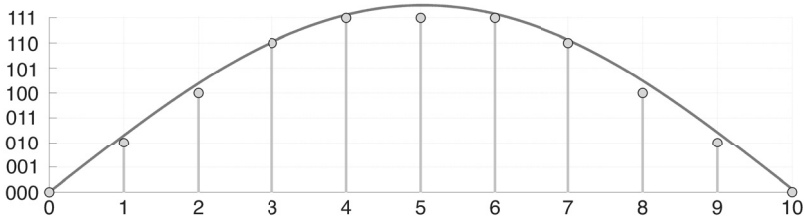
Podstawowy układ regulacji

Podstawowy układ regulacji cyfrowej ze sprzężeniem zwrotnym pokazany jest na **rysunku 1**. **Obiekt regulacji** (*plant*) jest to dowolny proces fizyczny, który do zadowalającej pracy wymaga sygnałów sterujących. Przez „zadowalającą pracę" rozumiemy to, że sygnał wyjściowy $y(t)$ różni się od sygnału odniesienia (wartości zadanej) $y^{zad}(t)$ dostatecznie mało, pomimo oddziaływania zakłóceń $w(t)$ na obiekt i pomimo szumu pomiarowego $v(t)$. Jeżeli sygnał $y^{zad}(t)$ jest stały w długich okresach (czasu), mówimy o zadaniu regulacji stałowartościowej. Jeżeli $y^{zad}(t)$ jest zmienne, to mamy regulację nadążną. Sygnały ciągłe będziemy oznaczać jako $x(t)$, gdzie t jest czasem ciągłym. Sygnały cyfrowe oznaczymy $x(k)$, gdzie k oznacza kolejne (dyskretne, czyli skwantowane) chwile czasowe. W układzie regulacji cyfrowej (rysunek 1) występują oba rodzaje sygnałów jednocześnie. Pomiędzy nimi są przetworniki: sygnał ciągły jest zamieniany na cyfrowy (dyskretny) w przetworniku analogowo-cyfrowym (*Analog Digital Converter*), oznaczonym na rysunku 1 jako ADC. Przetwarzanie w drugą stronę realizuje przetwornik cyfrowo-analogowy DAC z układem ZOH podtrzymującym napięcie do czasu następnego przetworzenia.

Różnica pomiędzy wartością zadaną, a zmierzonym wyjściem obiektu to **uchyb regulacji** $e(k) = y^{zad}(k) - y(k)$. Kolejnym warunkiem, jaki musi spełnić układ regulacji jest wymaganie, aby uchyb regulacji pozostawał dostatecznie mały także przy niewielkich zmianach dynamiki obiektu. Proces utrzymywania małego uchybu regulacji to **proces regulacji**. O procesie regulacji, który zapewnia mały uchyb w warunkch



Rys. 1. Układ regulacji cyfrowej ze sprzężeniem zwrotnym

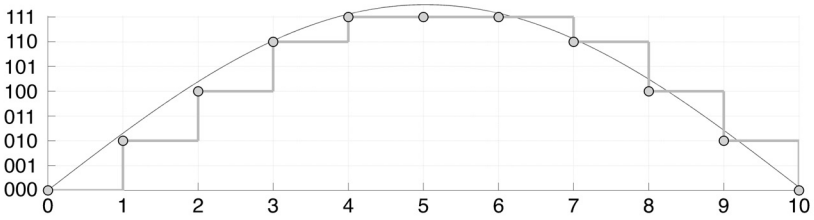


Rys. 2. Przetwornik ADC – dyskretyzacja sygnału analogowego

występowania zakłóceń obiektu i szumów pomiarowych mówimy, że ma dobrą **kompensację zakłóceń** (*disturbance rejection*). O systemie, który zapewnia dobrą regulację pomimo zmian parametrów dynamiki obiektu, mówimy, że ma małą wrażliwość na zmiany tych parametrów. System, który łączy obie te cechy, to znaczy że zapewnia dobrą kompensację zakłóceń przy jednoczesnej małej wrażliwości na zmiany dynamiki obiektu, to **system odporny**.

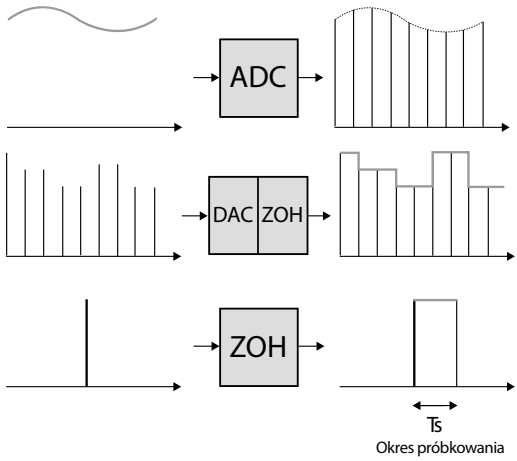
Regulację odporną (*robust*) można uzyskać w zamkniętym układzie regulacji: zmierzone wyjście obiektu $y(t)$, po przetworzeniu na $y(k)$ jest porównywane z wartością zadaną $y^{zad}(k)$. Zgodnie z zastosowanym prawem regulacji, na wyjściu regulatora jest generowany sygnał sterujący $u(k)$, który po przetworzeniu na $u(t)$, stanowi wymuszenie obiektu. Idea sprzężenia zwrotnego jest realizowana od początków automatyki. Regulację cyfrową wyróżnia stosowanie cyfrowego przetwarzania sygnałów pomiędzy wyjściem obiektu, a jego wejściem. To oznacza, że sygnał sterujący powstaje w wyniku realizacji odpowiedniego programu w procesorze (mikrokontrolerze). Regulowany proces jest ciągły, to znaczy opisują go sygnały analogowe, czyli ciągłe w czasie i ciągłe w zakresie swoich wartości. Procesory działają w dziedzinie czasu dyskretnego i na dyskretnych wartościach. Musi więc nastąpić dyskretyzacja sygnału wyjściowego $y(t)$ zarówno w czasie jak i dyskretyzacja jego wartości (**rysunek 2**). Wykonuje to **przetwornik analogowo-cyfrowy** ADC (*analog-to-digital converter*), który zamienia sygnał analogowy w ciąg liczb.

Typowy regulator cyfrowy ma stały **okres próbkowania** T_s (*sample period*). Jest to najczęściej realizowane za pomocą przerwań systemowych. W procesorach Cortex-M



Rys. 3. *Zero Order Hold* – sygnał analogowy odtworzony z cyfrowego

można do tego celu wykorzystać wewnętrzny licznik SysTick. W równych odstępach czasu T_s wykonywane jest przetwarzanie analogowo-cyfrowe, czyli mierzona wartość wyjścia obiektu $y(t)$ jest przetwarzana na liczbę $y(k)$, gdzie k symbolizuje bieżącą chwilę w dziedzinie czasu dyskretnego. Na rysunku 2 pokazano przykład działania przetwornika ADC 3-bitowego. Możliwe wartości liczbowe na wyjściu tego przetwornika to (binarnie) `000, 001, 010, 011, 100, 101, 110, 111`. Przetwornik w chwilach czasowych oznaczonych numerami od $k = 0$ do $k = 10$ zamienia wartość chwilową sygnału na najbliższą liczbę. Mikrokontroler otrzymuje liczbę $y(k)$, może wykonać odejmowanie $e(k) = y^{zad}(k) - y(k)$ i dalsze przetwarzanie danych. W wyniku działania algorytmu regulacji wyliczana jest wartość $u(k)$ która, przez przetwornik cyfrowo-analogowy, zostaje zamieniona na odpowiadające jej napięcie $u(t)$. To napięcie będzie podtrzymywane na stałej wartości przez układ ZOH *Zero Order Hold*, przez cały następny okres próbkowania T_s, czyli aż do wyliczenia nowej wartości sterowania $u(k+1)$ w następnym kroku działania regulatora. Zastosowanie ZOH powoduje, że **układ regulacji z rysunku 1 pracuje w otwartej pętli regulacji w czasie pomiędzy kolejnymi próbkami, gdyż sygnał sterujący $u(t)$ pozostaje stały bez względu na wartość wyjścia obiektu $y(t)$.**



Rys. 4. Przetwarzanie ADC i DAC

Na **rysunku 3** przedstawiono działanie przetwornika cyfrowo-analogowego i połączonego z nim układu ZOH. Jeżeli założymy, że $u(k) = y(k)$, to możemy porównać oryginalny sygnał analogowy z rysunku 2 z sygnałem, który powstał jako wynik przetwarzania analogowo-cyfrowego, a następnie cyfrowo-analogowego z podtrzymaniem ZOH. Dużą rozbieżność pomiędzy sygnałami można zmniejszyć stosując przetwornik o większej liczbie bitów, na przykład Cortex-M3 jest wyposażony w przetwornik 12-bitowy, który zapewnia podział zakresu przetwarzania na 4096 części (3-bitowy przetwornik z przykładu dzieli zakres na 8 części). Poprawę odwzorowania dynamiki sygnału uzyskamy zwiększając częstotliwość próbkowania, czyli zmniejszając okres T_s. Jak często trzeba próbkować określony sygnał analogowy wynika z Twierdzenia o Próbkowaniu (twierdzenie Whittakera-Nyquista-Kotielnikova-Shannona).

Zagadnienia próbkowania sygnałów analogowych, przetwarzania cyfrowych i zamiany wyniku na sygnały analogowe (rysunek 3) są przedmiotem dziedziny wiedzy zwanej DSP *Digital Signal Processing*, czyli cyfrowego przetwarzania sygnałów.

Płyty rozwojowe używane w książce

Projekty opisane w książce wykorzystują zestawy uruchomieniowe (*evaluation boards*):

- ZL27ARM, procesor STM32F103VB, Cortex-M3
- STM3210E-EVAL, procesor STM32F103ZE, Cortex-M3

Przykładowe programy dołączone do książki podzielone są na katalogi: każda płyta ma swój katalog. Wykaz projektów podany jest w **tabeli 1**.

PROGRAMY DOŁĄCZONE DO KSIĄŻKI PRZEZNACZONE SĄ WYŁĄCZNIE DO CELÓW SZKOLENIOWYCH.
AUTOR NIE PONOSI ODPOWIEDZIALNOŚCI ZA ŻADNE EWENTUALNE, BEZPOŚREDNIE I POŚREDNIE SZKODY WYNIKŁE Z ICH WYKORZYSTANIA.

THE SOFTWARE INCLUDED IS FOR GUIDANCE ONLY.
AUTHOR SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM USE OF THIS SOFTWARE.

Programy są pisane w taki sposób, aby podkreślić omawiane zagadnienia. Nie mają wstawionych zabezpieczeń i nie realizują wymaganych algorytmów. Na przykład programy transmisji szeregowej USART nie są zabezpieczone na wypadek zerwania transmisji w trakcie wymiany danych – program nie wykryje takiej sytuacji i będzie czekał na kolejne bajty. Drugi przykład: nie ma pełnej obsługi protokołu MODBUS: rozpoznawania funkcji, numerów rejestrów itd. Program jest tak napisany, żeby doszło do wymiany danych w jednym, określonym przypadku.

Ta uwaga dotyczy **wszystkich** programów dołączonych do książki.

Tabela 1. Wykaz projektów dołączonych do książki

ZL27ARM	Strona	STM3210E-EVAL	Strona
Projekt01_LED	106	Projekt01_LED	237
Projekt02_LCD	163		
Projekt03_asm	180	Projekt03_asm	
Projekt04_SysTick	192		
Projekt05_keypad	199		
Projekt06_scanf	202		
Projekt07_RTC	206		
Projekt08_TIM4-LED	211		
Projekt09_TIM2switch	216		
Projekt10_RadioControl	220		
Projekt11_6WheelDrive	226		
Projekt12_DistanceSensor	231		
Projekt13_PDC			
		Projekt14_TFTLCD	238
Projekt15_StepperMotor	250	Projekt15_StepperMotor	250
Projekt16_StepperMotor	261	Projekt16_StepperMotor	261
		Projekt17_FixedPoint_sin	282
		Projekt18_cube3D	284
		Projekt19_cube3D_FixedPoint	290
Projekt20_DMA_M2M	297	Projekt20_DMA_M2M	297
Projekt21_ADC1	306	Projekt21_ADC1	306
Projekt22_Pt100	312	Projekt22_Pt100	312
		Projekt23_DAC_sin	322
		Projekt24_1_Modbus_Master	333
		Projekt24_2_Modbus_Master	340
Projekt25_Modbus_Slave	336	Projekt25_Modbus_Slave	336
		Projekt26_I2C	346
Projekt27_SPI	348		
		Projekt28_obiekt_xxxx	353
		Projekt29_PI	361
		Projekt30_PID	367
		Projekt31_1_pole_placement	384
		Projekt31_2_pole_placement	393
		Projekt31_3_pole_placement	393
Projekt32_MultiTasking	400	Projekt32_MultiTasking	400

Programy można pobrać ze strony wydawnictwa: *www.wydawnictwo.btc.pl* lub ze strony autora: *www.edukacja.plum.pl*.

Nazewnictwo (terminologia)

W latach siedemdziesiątych i osiemdziesiątych ubiegłego wieku, czyli w czasach gwałtownego rozwoju techniki mikroprocesorowej, w Polsce wydawano stosunkowo dużo książek o układach logicznych (TTL i ECL) oraz o mikroprocesorach. Wielu autorów starało się tłumaczyć na język polski takie terminy angielskie jak *NAND gate*, *NOR gate*, czy *XOR gate*. Były to najczęściej „brama NIEI", „brama NIELUB", „brama ALBO". Jednocześnie młody inżynier (autor) korzystał z oryginalnych amerykańskich katalogów. Próby tłumaczenia wszystkiego na język polski sprawiały, że często książki polskich autorów stawały się nieczytelne. Po latach ustaliło się polskie nazewnictwo, mówimy teraz bramka NAND, bramka NOR i XOR. Nawet prosta zamiana oznaczenia zmiennej zespolonej w transmitancji operatorowej członu oscylacyjnego z s na p

$$G_o(p) = \frac{\omega_n^2}{p^2 + 2\zeta\omega_n p + \omega_n^2}$$

powoduje, że tekst jest trudny w odbiorze. A zamiana z s na z prowadzi wręcz do nieporozumień (z to zmienna transformaty dyskretnej $\mathcal{Z}$).

Dla czytelności przekazu, ważne jest stosowanie zwyczajowych oznaczeń i powszechnie przyjętego nazewnictwa. Materiały źródłowe producentów mikrokontrolerów są wydawane w języku angielskim. Czytelnik tej książki będzie musiał z nich korzystać. Dlatego, dla jasności przekazu warto jest zachować oryginalną (czyli angielską) terminologię. Ponadto większość nazw angielskich jest powszechnie używana w polskim świecie techniki wbudowanej (*embedded*). Dlatego, na przykład (szczególnie na rysunkach) zamiast oznaczenia **PoDS** („Pamięć o Dostępie Swobodnym"), będzie stosowany opis *Random-Access Memory* lub samo RAM. Nazwy angielskie są wyróżnione *kursywą* (ang. *italic*).

Kolejna konwencja dotyczy adresowania pamięci na rysunkach: z dołu do góry, czy odwrotnie. Przyjęte jest, że mapa pamięci jest rysowana z adresami rosnącymi do góry, czyli najmniejsze są na dole, jak na rysunku 3.8. Jednak listing programu będziemy drukować odwrotnie, to znaczy od góry na dół, jak w tabeli 3.1. Pozwala to czytać program naturalnie, jak tekst w książce.

Przedrostki dwójkowe i przedrostki SI

W całej dziedzinie komputerowej występuje przemieszanie przedrostków dwójkowych (G = 2^{30}) i dziesiętnych stosowanych w systemie SI (G = 10^9). Tak więc producenci dysków twardych podają pojemność w wartościach dziesiętnych, a systemy operacyjne w dwójkowych, przez co dysk 10 GB w systemie operacyjnym pokazuje jedynie 9,3 GB.

W związku z problemami związanymi z wymieszaniem przedrostków dwójkowych i tych z układu SI, w roku 1998 Międzynarodowa Komisja Elektrotechniczna (IEC) zaproponowała używanie alternatywnego nazewnictwa (**tabela 2**):

Tabela 2. Przedrostki dwójkowe IEC

Nazwa	Prefix	Rozmiar
kibi	Ki	2^{10} =1024
mebi	Mi	2^{20} =1 048 576
gibi	Gi	2^{30} =1 073 741 824
tebi	Ti	2^{40} =1 099 511 627 776

Niestety ten format nazw nie przyjmuje się zbyt szybko, dlatego występują problemy, gdy jedna strona stosuje przedrostki układu SI, a druga spodziewa się przedrostków dwójkowych. Jednym ze skutków jest fakt, że początkujący programista uważa, iż jeden kilobajt to tysiąc bajtów, a zaawansowany, że kilometr to 1024 metry.

W książce, podobnie jak to jest w systemie Windows, będziemy stosować system binarny, ale oznaczać, jakby to był SI, (**tabela 3**):

Tabela 3. Przedrostki stosowane w książce

Nazwa	Prefix	Rozmiar
kilo	k	2^{10} =1024
Mega	M	2^{20} =1 048 576
Giga	G	2^{30} =1 073 741 824
Tera	T	2^{40} =1 099 511 627 776