AKADEMIA MORSKA W SZCZECINIE

JEDNOSTKA ORGANIZACYJNA:
WYDZIAŁ NAWIGACYJNY, CENTRUM INŻYNIERII RUCHU MORSKIEGO

PRZEWODNIK METODYCZNY

Morskie Systemy Zintegrowane
Protokoły Transmisji Danych
Laboratorium

Opracował:	mgr inż. Bilewski Mateusz
Zatwierdził:	dr inż. Jankowski Stefan
Obowiązuje od: 2013/2014	

RAMOWY SPIS TREŚCI

1. CEL I ZAKRES PRZEDMIOTU

Celem kształcenia jest wykształcenie umiejętności efektywnego wykorzystania informacji pochodzących z urządzeń i systemów nawigacyjnych zainstalowanych w zintegrowanym mostku nawigacyjnym ze zwróceniem uwagi na metody wymiany danych.

2. PROGRAM PRZEDMIOTU

2.1. Program zajęć, regulaminy: BHP, P.POŻ., laboratorium, przykładowy program w języku C++; [MSZ/PTD]
2.2. Nadajniki systemów teletransmisji danych; [MSZ]
2.3. Odbiorniki systemów teletransmisji danych; [MSZ]
2.4. Przerwania sprzętowe; [MSZ]
2.5. Programowe konwertery danych cz. 1; [MSZ]
2.6. Programowe konwertery danych cz. 2; [MSZ]
2.7. Obsługa wyświetlaczy LCD; [MSZ]
2.8. Analiza danych cyfrowych; [PTD]
2.9. Konwersja danych liczbowych cz. 1; [PTD]
2.10. Konwersja danych liczbowych cz. 2; [PTD]
2.11. Transmisja szeregowa; [PTD]
2.12. Parser NMEA0183 cz. 1; [PTD]
2.13. Parser NMEA0183 cz. 2; [PTD]
2.14. Zaliczenie praktyczne, komunikacja UDP; [MSZ/PTD]
2.15. Poprawa zaliczenia praktycznego, komunikacja TCP/IP. [MSZ/PTD]

3. PRZEBIEG ĆWICZEŃ

3.1. Zajęcia nr 1

WSTĘP TEORETYCZNY:

Do programowania mikrokontrolerów w języku C potrzebne jest kilka elementów:

- mikrokontroler,
- programator,
- edytor tekstu,
- kompilator.

Zamiast dwóch ostatnich można użyć środowiska programistycznego IDE (Integrated Development Environment) takiego jak pakiet Visual Studio, narzędzia firmy Borland, Eclipse, Dev-C++ czy NetBeans.

Kompilator języka C jest programem, który tłumaczy kod źródłowy napisany przez nas do języka asembler, a następnie do postaci zrozumiałej dla komputera (maszyny cyfrowej), czyli do postaci ciągu zer i jedynek, które sterują pracą poszczególnych elementów komputera.

Programator jest to elektroniczne urządzenie przeznaczone do programowania układów elektronicznych. Dane, które mają być zaprogramowane (program lub inne informacje) są dostarczane do programatora z innego urządzenia, np. komputera PC za pośrednictwem interfejsu. Programator następnie przekształca te dane (poszczególne bity) na odpowiednie wielkości elektryczne, np. napięcie elektryczne o określonej wartości, które trwa przez określony czas. Układ programowany ma też zazwyczaj wejście służące do uaktywnienia programowania. Jeśli to wejście jest aktywne i na wejściu programowalnym jest ustawione odpowiednie napięcie przez odpowiednio długi czas, to wtedy zostaje zaprogramowany 1 bit. Sytuację powtarza się kolejno, aż wszystkie bity zostaną przesłane do programatora, a z niego dalej do układu programowanego.

Spośród mikrokontrolerów wyodrębnia się grupę mikrokontrolerów, które nie wymagają wyjęcia z urządzenia docelowego w celu zaprogramowania – są to mikrokontrolery ISP (In System Programmable). Rozwiązanie takie ułatwia pracę, ale nieznacznie komplikuje docelowe urządzenie.

W języku C każdą instrukcje zakańczamy średnikiem ';', a komentarze umieszczamy po podwójnym znaku backslash „//” lub pomiędzy zestawem znaków „/*” i „*/” jak na przykładzie:

```
a=5;    //komentarz jednolinijkowy
/*komentarz
wielo-
linijkowy*/
```

Zmienna jest to konstrukcja programistyczna posiadająca trzy podstawowe atrybuty:

- symboliczną nazwę,
- miejsce przechowywania,
- wartość,
- typ.

W kodzie źródłowym można odwoływać się przy pomocy nazwy do wartości lub miejsca przechowywania. Nazwa służy do identyfikowania zmiennej w związku z tym często nazywana jest identyfikatorem. Miejsce przechowywania przeważnie znajduje się w pamięci komputera i określane jest przez adres i długość danych. Wartość to zawartość miejsca przechowywania. Typ określa rodzaj danych przechowywanych w zmiennej i co za tym idzie sposób reprezentacji wartości w miejscu przechowywania. W programie wartość zmiennej może być odczytywana lub zastępowana. W zależności od rodzaju języka typ zmiennej

może być stały lub zmienny. Konstrukcją podobną lecz nie pozwalającą na modyfikowanie wartości jest stała.

W języku C wyróżnia się zmienne:

- char - typ znakowy lub liczba jednobajtowa,
- int - typ całkowity,
- float - typ zmiennopozycyjny/zmiennoprzecinkowy 4B,
- double - typ zmiennopozycyjny o podwójnej precyzji 8B,
- bool - typ logiczny.

Modyfikatory:

- volatile - wyłącza optymalizację odczytu – szczególnie użyteczne przy programowaniu mikrokontrolerów,
- register - tworzy zmienną w rejestrze – w miejscu, które mikrokontroler może najszybciej odczytać,
- static - używane w funkcjach - zachowanie wartości do następnego użycia funkcji,
- extern - zmienna globalna zadeklarowana w innym pliku.

Specyfikatory:

- signed - zmienna ze znakiem (wartość domyślna),
- unsigned - zmienna bez znaku (podwojenie zakresu),
- short - zmniejszenie (jeśli to możliwe) dokładności dla int,
- long - zwiększenie (jeśli to możliwe) dokładności dla int.

Zmienne deklaruje się poprzez wpisanie w kolejności modyfikatorów, specyfikatorów, typu i nazwy oddzielonych spacją, np.

```
modyfikatory specyfikatory typ nazwa_zmiennej;
register long unsigned int a;
```

Podczas deklaracji zmiennej można przypisać jej wartość początkową, ale nie jest to konieczne.

Stałe deklaruje się poprzez instrukcję const, np.:

```
const typ nazwa=wartość;
const float pi=3.14159265;
```

Podczas deklaracji stałej trzeba przypisać jej wartość początkową.

Operatory są to znaki, które wykonują zadaną funkcję na jednym, dwóch lub trzech operandach – zmiennych. Operatory dzielimy na:

- przypisania,
- arytmetyczne,
- inkrementacji i dekrementacji,
- bitowe,
- porównania,
- logiczne,
- specjalne.

Operator przypisania '=' przypisuje wartość prawego argumentu lewemu, np.:

```
b=5;
a=b+4;
```

Z operatorem przypisania wiąże się jeszcze pojęcie rzutowania. Rzutowanie jest to konwersja danej jednego typu na daną innego typu. Konwersja taka może być jawna lub nie jawna, np.:

```
zmienna_typu_a=zmienna_typu_b;          //rzutowanie niejawne
zmienna_typu_a=(typ_a)zmienna_typu_b;  //rzutowanie jawne
float a=3.453;          //zadeklarowanie i przypisanie wartości do a
```

int b=a; *//przypisanie do b z rzutowaniem niejawnym a do int*
long int c=(long int)a; *//przypisanie do c z rzutowaniem jawnym a do long int*

Operatory arytmetyczne to:

- dodawania '+',
- odejmowania '-',
- mnożenia '*',
- dzielenia '/',
- modulo (reszta z dzielenia) '%'.

Przy programowaniu często zmienne całkowite zmieniane są o 1, aby skrócić zapis wprowadzono dodatkowe operatory inkrementacji i dekrementacji:

- pre-inkrementacja '++zmienna',
- post-inkrementacja 'zmienna++',
- pre-dekrementacja '--zmienna',
- post-dekrementacja 'zmienna--'.

Przykłady:

```
a=0;b=0;c=0;d=0;    //a=0, b=0, c=0, d=0
b=a++;              //a=1, b=0, c=0, d=0
c=++a;              //a=2, b=0, c=2, d=0
d=--a;              //a=1, b=0, c=0, d=2
```

PŁYTKA TESTOWA (KAMAMI ZL15AVR):

PROGRAM KOMPUTEROWY:

```
//Połączenia:
//PA0-LD0

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
        DDRA = 0xFF;
        while(1)
        {
                PORTA = 0x00;
                _delay_ms(500);
                PORTA = 0x01;
                _delay_ms(500);
        }
        return 0;
}
```

ZADANIA:

Zapoznaj się z instrukcją do płytki ZL15AVR.
Przeanalizuj program linia po linii.
Przepisz program oraz zaprogramuj płytkę.
Przeanalizuj ponownie program linia po linii i porównaj z poprzednią analizą.

3.2. Zajęcia nr 2

PROGRAMY KOMPUTEROWE:

a)

```
//Połączenia:
//PA0-LD0
//PA1-LD1
//PA2-LD2
//PA3-LD3
//PA4-LD4
//PA5-LD5
//PA6-LD6
//PA7-LD7

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
	DDRA = 0xFF;

	while(1)
	{
		PORTA = 0x00;
		_delay_ms(500);
		PORTA = 0x01;
		_delay_ms(500);
		PORTA = 0x02;
		_delay_ms(500);
		PORTA = 0x04;
		_delay_ms(500);
		PORTA = 0x08;
		_delay_ms(500);
		PORTA = 0x10;
		_delay_ms(500);
		PORTA = 0x20;
		_delay_ms(500);
		PORTA = 0x40;
		_delay_ms(500);
		PORTA = 0x80;
		_delay_ms(500);
	}
	return 0;
}
```

b)

```
//Połączenia:
//PA0-LD0
//PA1-LD1
//PA2-LD2
//PA3-LD3
//PA4-LD4
//PA5-LD5
//PA6-LD6
//PA7-LD7

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
	DDRA = 0b01110100;

	while(1)
	{
		PORTA = 0x00;
		_delay_ms(500);
		PORTA = 0x01;
```

```
                _delay_ms(500);
                PORTA = 0x02;
                _delay_ms(500);
                PORTA = 0x04;
                _delay_ms(500);
                PORTA = 0x08;
                _delay_ms(500);
                PORTA = 0x10;
                _delay_ms(500);
                PORTA = 0x20;
                _delay_ms(500);
                PORTA = 0x40;
                _delay_ms(500);
                PORTA = 0x80;
                _delay_ms(500);
        }
        return 0;
}
```

c)

```
//Połączenia:
//PA0-LD0
//PA1-LD1
//PA2-LD2
//PA3-LD3
//PA4-LD4
//PA5-LD5
//PA6-LD6
//PA7-LD7

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
        DDRA = 0xFF;

        while(1)
        {
                PORTA = 0x00;
                _delay_ms(500);
                PORTA |= 0x01;
                _delay_ms(500);
                PORTA |= 0x02;
                _delay_ms(500);
                PORTA |= 0x04;
                _delay_ms(500);
                PORTA |= 0x08;
                _delay_ms(500);
                PORTA |= 0x10;
                _delay_ms(500);
                PORTA |= 0x20;
                _delay_ms(500);
                PORTA |= 0x40;
                _delay_ms(500);
                PORTA |= 0x80;
                _delay_ms(500);
        }
        return 0;
}
```

d)

```
//Połączenia:
//PA0-LD0
//PA1-LD1
//PA2-LD2
//PA3-LD3
//PA4-LD4
//PA5-LD5
//PA6-LD6
//PA7-LD7

#define F_CPU 16000000
```

```
#include <avr/io.h>
#include <util/delay.h>

int main()
{
	DDRA = 0xFF;

	while(1)
	{
		PORTA = 0b11111111;
		_delay_ms(500);
		PORTA &= 0b11111110;
		_delay_ms(500);
		PORTA &= 0b11111101;
		_delay_ms(500);
		PORTA &= 0b11111011;
		_delay_ms(500);
		PORTA &= 0b11110111;
		_delay_ms(500);
		PORTA &= 0b11101111;
		_delay_ms(500);
		PORTA &= 0b11011111;
		_delay_ms(500);
		PORTA &= 0b10111111;
		_delay_ms(500);
		PORTA &= 0b01111111;
		_delay_ms(500);
	}
	return 0;
}
```

e)

```
//Połączenia:
//PA0-LD0
//PA1-LD1
//PA2-LD2
//PA3-LD3
//PA4-LD4
//PA5-LD5
//PA6-LD6
//PA7-LD7

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
	DDRA = 0xFF;
	PORTA = 0x00;
	int a;
	while(1)
	{
		a=0x01;
		for(int i=0;i<8;i++)
		{
			_delay_ms(500);
			PORTA = a;
			a=a<<1;    //zamienić na a|=a<<1;
		}
	}
	return 0;
}
```

f)

```
//Połączenia:
//PA0-LD?
//PA1-LD?
//PA2-LD?
//PA3-LD?
//PA4-LD?
//PA5-LD?
//PA6-LD?
```

```c
//PA7-LD?

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
	DDRA = 0xFF;

	while(1)
	{
		PORTA = 0x00;
		_delay_ms(500);
		PORTA = 0x04;
		_delay_ms(500);
		PORTA = 0x40;
		_delay_ms(500);
		PORTA = 0x02;
		_delay_ms(500);
		PORTA = 0x08;
		_delay_ms(500);
		PORTA = 0x10;
		_delay_ms(500);
		PORTA = 0x80;
		_delay_ms(500);
		PORTA = 0x01;
		_delay_ms(500);
		PORTA = 0x20;
		_delay_ms(500);
	}
	return 0;
}
```

ZADANIA:

Przeanalizuj program a) i sprawdź jak działa.

Sprawdź czemu program b) nie działa prawidłowo.

Przeanalizuj programy c) i d) i sprawdź co daje zmiana operatorów logicznych.

Zmodyfikuj program e) tak, aby diody kolejno się zapalały i gasły.

Zmień tak połączenia dla programu f), aby diody zapalały się i gasiły w kolejności od lewej do prawej.

3.3. Zajęcia nr 3

PROGRAMY KOMPUTEROWE:

a)

```
//Połączenia:
//PA0-LD0
//PB0-SW0

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
	DDRA = 0xFF;
	DDRB = 0xFE;
	PORTB = 0x01;

	while(1)
	{
		if(PINB==0x01) PORTA=0x01;
		else PORTA=0x00;
	}
	return 0;
}
```

b)

```
//Połączenia:
//PA0-LD0
//PA1-LD1
//PB0-SW0
//PB1-SW1

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
	DDRA = 0xFF;
	DDRB = 0b11111100;
	PORTB = 0x03;

	while(1)
	{
		if((PINB&0x01)==0x01) PORTA|=0x01;
		else PORTA&=0xfe;
		if((PINB&0x02)==0x02) PORTA|=0x02;
		else PORTA&=0xfd;
	}
	return 0;
}
```

c)

```
//Połączenia:
//PA0-LD0
//PB0-SW0


#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
	DDRA = 0xFF;
	DDRB = 0b11111110;
	PORTB = 0x01;

	while(1)
	{
```

```
                if((PINB&0x01)==0x00)
                {
                        if((PORTA&0x01)==0x01) PORTA&=0xfe;
                        else PORTA|=0x01;
                        _delay_ms(100);
                }

        }
        return 0;
}
```

d)

```
//Połączenia:
//PA0-LD0
//PB0-SW0

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
        DDRA = 0xFF;
        DDRB = 0b11111110;
        PORTB = 0x01;

        while(1)
        {
                if((PINB&0x01)==0x00)
                {
                        while((PINB&0x01)==0x00) _delay_ms(1);
                        if((PORTA&0x01)==0x01) PORTA&=0xfe;
                        else PORTA|=0x01;
                }

        }
        return 0;
}
```

e)

```
//Połączenia:
//PA0-LD0
//PA1-LD1
//PB0-SW0
//PB1-SW1

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
        DDRA = 0xFF;
        DDRB = 0b11111100;
        PORTB = 0x03;

        while(1)
        {
                if((PINB&0x01)==0x00)
                {
                        while((PINB&0x01)==0x00) _delay_ms(1);
                        if((PORTA&0x01)==0x01) PORTA&=0xfe;
                        else PORTA|=0x01;
                }
                if((PINB&0x02)==0x00)
                {
                        while((PINB&0x02)==0x00) _delay_ms(1);
                        if((PORTA&0x02)==0x02) PORTA&=0xfd;
                        else PORTA|=0x02;
                }
        }
        return 0;
}
```

f)

```
//Połączenia:
//PA0-LD0
//PA1-LD1
//PA2-LD2
//PA3-LD3
//PB0-SW0
//PB1-SW1
//PB2-SW2
//PB3-SW3

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
	DDRA = 0xFF;
	DDRB = 0xF0;
	PORTB = 0x0F;

	while(1)
	{
		if(!(PINB&0x01))
		{
			while(!(PINB&0x01)) _delay_ms(1);
			if(PORTA&0x01) PORTA&=0xfe;
			else PORTA|=0x01;
		}
		if(!(PINB&0x02))
		{
			while(!(PINB&0x02)) _delay_ms(1);
			if(PORTA&0x02) PORTA&=0xfd;
			else PORTA|=0x02;
		}
		if(!(PINB&0x04))
		{
			while(!(PINB&0x04)) _delay_ms(1);
			if(PORTA&0x04) PORTA&=0xfb;
			else PORTA|=0x04;
		}
		if(!(PINB&0x08))
		{
			while(!(PINB&0x08)) _delay_ms(1);
			if(PORTA&0x08) PORTA&=0xf7;
			else PORTA|=0x08;
		}
	}
	return 0;
}
```

g)

```
//Połączenia:
//PA0-LD0
//PA1-LD1
//PA2-LD2
//PA3-LD3
//PA4-SW0
//PA5-SW1
//PA6-SW2
//PA7-SW3

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main()
{
	DDRA = 0x0F;
	PORTA = 0xF0;

	while(1)
```

```
        {
                if(!(PINA&0x10))
                {
                        while(!(PINA&0x10)) _delay_ms(1);
                        if(PORTA&0x01) PORTA&=0xfe;
                        else PORTA|=0x01;
                }
                if(!(PINA&0x20))
                {
                        while(!(PINA&0x20)) _delay_ms(1);
                        if(PORTA&0x02) PORTA&=0xfd;
                        else PORTA|=0x02;
                }
                if(!(PINA&0x40))
                {
                        while(!(PINA&0x40)) _delay_ms(1);
                        if(PORTA&0x04) PORTA&=0xfb;
                        else PORTA|=0x04;
                }
                if(!(PINA&0x80))
                {
                        while(!(PINA&0x80)) _delay_ms(1);
                        if(PORTA&0x08) PORTA&=0xf7;
                        else PORTA|=0x08;
                }
        }
        return 0;
}
```

ZADANIA:

Przeanalizuj program a) i spróbuj rozszerzyć go o drugi przycisk [program b)].

Sprawdź dlaczego w programie c) dołożono dodatkową pętlę – w czym ona pomaga?

Przeanalizuj program z redukcją szumów przełączania [program d)].

Rozszerz program d) na dwa lub cztery przyciski [programy e) i f)].

Spróbuj zmodyfikować program f) oraz przełącz odpowiednie przewody tak, aby wszystkie połączenia były na jednym porcie [program g)].

3.4. Zajęcia nr 4

PROGRAM KOMPUTEROWY:

```
//Połączenia:
//PA0-LD0
//PA1-LD1

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
volatile unsigned int Counter=0;

int main()
{

        DDRA = 0xFF;
        PORTA = 0x00;
        TCCR0 = (1 << CS00);
        TIMSK = (1 << TOIE0);
        sei();
        while(1)
        {
                _delay_ms(1000);
                if(PORTA&0x01) PORTA&=0xfe;
                else PORTA|=0x01;
        }
        return 0;
}

ISR(TIMER0_OVF_vect)
{
        Counter++;
        if(Counter==0)
        {
                if(PORTA&0x02) PORTA&=0xfd;
                else PORTA|=0x02;
        }
}
```

ZADANIA:

Przeanalizuj program z przerwaniem sprzętowym.

Które polecenia w programie odpowiadają za zdefiniowanie poleceń wykonywanych podczas przerwania, a które uruchamiają licznik przerwań?

Zmodyfikuj program tak, aby obie diody (jedna sterowania w programie głównym, druga w przerwaniu) zapalały się i gasły w tej samym momencie.

3.5. Zajęcia nr 5

PROGRAM KOMPUTEROWY:

```
//Połączenia:
//PA0-7SEG(A)
//PA1-7SEG(B)
//PA2-7SEG(C)
//PA3-7SEG(D)
//PA4-7SEG(E)
//PA5-7SEG(F)
//PA6-7SEG(G)
//PA7-7SEG(.)
//PB0-7SEG(0)
//PB1-7SEG(1)

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

volatile unsigned int Counter=0;
unsigned char Digits[]= {192, 249, 164};
volatile unsigned char DigitA=0;
volatile unsigned char DigitB=0;

int main()
{
	DDRA = 0xFF;
	DDRB = 0x03;
	TCCR0 = (1 << CS00);
	TIMSK = (1 << TOIE0);
	sei();
	while(1)
	{
		_delay_ms(1000);
		if(DigitA==0)
		{
			DigitA=1;
			DigitB=0;
		}
		else
		{
			DigitA=0;
			DigitB=2;
		}
	}
	return 0;
}

ISR(TIMER0_OVF_vect)
{
	Counter++;
	if(Counter==100) Counter=0;
	if(Counter==0)
	{
		PORTB=0x01;
		PORTA=Digits[DigitA];
	}
	if(Counter==50)
	{
		PORTB=0x02;
		PORTA=Digits[DigitB];
	}
}
```

ZADANIA:

Rozszerz wyświetlane znaki na wszystkie cyfry.

Zmodyfikuj tak program, aby stworzyć licznik.

3.6. Zajęcia nr 6

PROGRAM KOMPUTEROWY:

```
//Połączenia:
//PA0-7SEG(A)
//PA1-7SEG(B)
//PA2-7SEG(C)
//PA3-7SEG(D)
//PA4-7SEG(E)
//PA5-7SEG(F)
//PA6-7SEG(G)
//PA7-7SEG(.)
//PB0-7SEG(0)
//PB1-7SEG(1)

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

volatile unsigned int Counter=0;
unsigned char Digits[]= {192, 249, 164};
volatile unsigned char DigitA=0;
volatile unsigned char DigitB=0;

int main()
{
        DDRA = 0xFF;
        DDRB = 0x03;
        TCCR0 = (1 << CS00);
        TIMSK = (1 << TOIE0);
        sei();
        while(1)
        {
                _delay_ms(1000);
                DigitA=2;
                DigitB=1;
        }
        return 0;
}

ISR(TIMER0_OVF_vect)
{
        Counter++;
        if(Counter==100) Counter=0;
        if(Counter==0)
        {
                PORTB=0x01;
                PORTA=Digits[DigitA];
        }
        if(Counter==50)
        {
                PORTB=0x02;
                PORTA=Digits[DigitB];
        }
}
```

ZADANIA:

Rozszerz program o wszystkie cyfry oraz dwa następne wyświetlacze.

Napisz program, który obsłuży dwa przyciski: START i STOP.

Napisz stoper, który będzie liczył do 999 sekund z dokładnością do 100ms.

3.7. Zajęcia nr 7

PROGRAM KOMPUTEROWY:

```
//Połączenia:
//zworka JP4 (GRAPH DISPLAY)
//
//LCD_GRAPH(D0) -PA0
//LCD_GRAPH(D1) -PA1
//LCD_GRAPH(D2) -PA2
//LCD_GRAPH(D3) -PA3
//LCD_GRAPH(D4) -PA4
//LCD_GRAPH(D5) -PA5
//LCD_GRAPH(D6) -PA6
//LCD_GRAPH(D7) -PA7
//LCD_GRAPH(RS) -PD0
//LCD_GRAPH(RW) -PD1
//LCD_GRAPH(E)  -PD2
//LCD_GRAPH(CS1)-PD4
//LCD_GRAPH(CS2)-PD3

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#include "KS0108.h"

int main()
{
	GLCD_Initialize();
	while(1)
	{

		GLCD_ClearScreen();

		GLCD_WriteStringXY("To",0,0);
		GLCD_WriteStringXY("Jest",0,1);
		GLCD_WriteStringXY("Przykladowy",0,3);
		GLCD_WriteStringXY("Tekst",0,4);
		_delay_ms(1000);
	}
	return 0;
}
```

ZADANIA:

Uruchom program obsługujący wyświetlacz LCD.

Zmodyfikuj program tak, aby wyświetlić dowolny znak w każdym z narożników wyświetlacza.

3.8. Zajęcia nr 8

PROGRAM KOMPUTEROWY:

```
//Połączenia:
//zworka JP4 (GRAPH DISPLAY)
//
//LCD_GRAPH(D0) -PA0
//LCD_GRAPH(D1) -PA1
//LCD_GRAPH(D2) -PA2
//LCD_GRAPH(D3) -PA3
//LCD_GRAPH(D4) -PA4
//LCD_GRAPH(D5) -PA5
//LCD_GRAPH(D6) -PA6
//LCD_GRAPH(D7) -PA7
//LCD_GRAPH(RS) -PD0
//LCD_GRAPH(RW) -PD1
//LCD_GRAPH(E)  -PD2
//LCD_GRAPH(CS1)-PD4
//LCD_GRAPH(CS2)-PD3

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#include "KS0108.h"

int main()
{
	GLCD_Initialize();
	char tekst[] = "dlugi napis";
	char temp[] = " ";
	while(1)
	{
		GLCD_ClearScreen();
		for(int i=0;i<sizeof(tekst)-1;i++)
		{
			temp[0]=tekst[i];
			GLCD_WriteStringXY(temp,6*i,0);
			_delay_ms(1000);
		}
	}
	return 0;
}
```

ZADANIA:

Zmodyfikuj program tak, aby wyświetlał długi napis, znak po znaku.

3.9. Zajęcia nr 9

PROGRAM KOMPUTEROWY:

```
//Połączenia:
//zworka JP4 (GRAPH DISPLAY)
//
//LCD_GRAPH(D0) -PA0
//LCD_GRAPH(D1) -PA1
//LCD_GRAPH(D2) -PA2
//LCD_GRAPH(D3) -PA3
//LCD_GRAPH(D4) -PA4
//LCD_GRAPH(D5) -PA5
//LCD_GRAPH(D6) -PA6
//LCD_GRAPH(D7) -PA7
//LCD_GRAPH(RS) -PD0
//LCD_GRAPH(RW) -PD1
//LCD_GRAPH(E)  -PD2
//LCD_GRAPH(CS1)-PD4
//LCD_GRAPH(CS2)-PD3

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#include "KS0108.h"
#include "lcdkamami.h"

int main()
{

	GLCD_Initialize();
	while(1)
	{
		GLCD_ClearScreen();

		GLCD_Bitmap(kamami,0,0,128,64);
		_delay_ms(1000);
	}
	return 0;
}
```

ZADANIA:

Zmodyfikuj plik kamami.h oraz program, aby wyświetlić dowolnie wybraną grafikę.

3.10. Zajęcia nr 10

PROGRAMY KOMPUTEROWE:

a)

```
//Połączenia:
//zworka JP4 (GRAPH DISPLAY)
//
//LCD_GRAPH(D0) -PA0
//LCD_GRAPH(D1) -PA1
//LCD_GRAPH(D2) -PA2
//LCD_GRAPH(D3) -PA3
//LCD_GRAPH(D4) -PA4
//LCD_GRAPH(D5) -PA5
//LCD_GRAPH(D6) -PA6
//LCD_GRAPH(D7) -PA7
//LCD_GRAPH(RS) -PD0
//LCD_GRAPH(RW) -PD1
//LCD_GRAPH(E)  -PD2
//LCD_GRAPH(CS1)-PD4
//LCD_GRAPH(CS2)-PD3
//PB0-JOY(S)
//PB1-JOY(N)

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#include "KS0108.h"
#include "lcdkamami.h"

int main(void)
{
        int y=0,t=2;

        PORTB = 0x03;
        DDRB = 0x00;
        GLCD_Initialize();                              // LCD initialization

        do{
        if (!(PINB & 0x01))
                {
                        if(y==7) y=7;
                        else y++;
                        t=1;
                }

                if (!(PINB & 0x02))
                {
                        if(y==0) y=0;
                        else y--;
                        t=0;
                }

                GLCD_ClearScreen();

                switch(t)
                {
                        case 0: GLCD_Bitmap(kamami1,0,y,8,8);
//GLCD_Bitmap(nazwa_tablicy_graficznej,pozycja_x,pozycja_y,rozmiar_x,rozmiar_y)
                        break;
                        default: GLCD_Bitmap(kamami2,0,y,8,8);
                        break;
                }
                _delay_ms(500);

        }while(1);
        return 0;
}
```

b)

```
//Połączenia:
//zworka JP4 (GRAPH DISPLAY)
//
//LCD_GRAPH(D0) -PA0
//LCD_GRAPH(D1) -PA1
//LCD_GRAPH(D2) -PA2
//LCD_GRAPH(D3) -PA3
//LCD_GRAPH(D4) -PA4
//LCD_GRAPH(D5) -PA5
//LCD_GRAPH(D6) -PA6
//LCD_GRAPH(D7) -PA7
//LCD_GRAPH(RS) -PD0
//LCD_GRAPH(RW) -PD1
//LCD_GRAPH(E)  -PD2
//LCD_GRAPH(CS1)-PD4
//LCD_GRAPH(CS2)-PD3
//PB0-JOY(S)
//PB1-JOY(N)
//PB2-JOY(E)
//PB3-JOY(W)

#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#include "KS0108.h"
#include "lcdkamami.h"

int main(void)
{
        int x=0,y=0,t=2;

        PORTB = 0x0F;
        DDRB = 0x00;

        GLCD_Initialize();                              // LCD initialization
        do
        {
                if (!(PINB & 0x01))
                {
                        if(y==7) y=7;
                        else y++;
                        t=1;
                }

                if (!(PINB & 0x02))
                {
                        if(y==0) y=0;
                        else y--;
                        t=0;
                }
                if (!(PINB & 0x04))
                {
                        if(y==15) x=15;
                        else x++;
                        t=3;
                }

                if (!(PINB & 0x08))
                {
                        if(y==0) x=0;
                        else x--;
                        t=2;
                }

                GLCD_ClearScreen();
                switch(t)
                {
                        case 0: GLCD_Bitmap(kamami1,8*x,y,8,8);
//GLCD_Bitmap(nazwa_tablicy_graficznej,pozycja_x,pozycja_y,rozmiar_x,rozmiar_y)
                        break;
```

```
                        case 1: GLCD_Bitmap(kamami3,8*x,y,8,8);
                        break;
                        case 2: GLCD_Bitmap(kamami4,8*x,y,8,8);
                        break;
                        default: GLCD_Bitmap(kamami2,8*x,y,8,8);
                        break;
                }

                _delay_ms(500);

        }while(1);
        return 0;
}
```

ZADANIA:

Rozszerz program a) o przesuwanie obiektu w poziomie.

Następnie rozszerz ten program o zmianę grafiki w zależności z której strony nastąpiło przesunięcie [program b)].

3.11. Zajęcia nr 11

PROGRAM KOMPUTEROWY:

```
//Połączenia:
//        RxD (con7) - PD0 (con18)
//        TxD (con7) - PD1 (con18)
//
//Na komputerze:
//Uruchomić HyperTerminal
//Dowolna nazwa połączenia
//COM1
//bitrate 4800b/s
//8bitów
//brak bitów parzystości
//1 bit stopu
//brak sterowania przepływem


#define F_CPU 16000000
#define BAUD(x) (((F_CPU/16)/x))

#include <avr/io.h>
#include <util/delay.h>
#include "usart.h"

int main()
{
        DDRD = 0xff;
        char x;
        USART_Init(BAUD(4800));
        while(1)
        {
                x = USART_GetChar();
                USART_PutChar('[');
                USART_PutChar(x);
                USART_PutChar(']');
                USART_PutChar('\n');
                USART_PutChar('\r');
        }
        return 0;
}
```

ZADANIA:

Zaimplementuj połączenie RS232 z komputerem wg informacji zawartych w programie.
Zmodyfikuj tak program, żeby echo odsyłane do komputera z płytki zmieniało wielkość liter.

3.12. Zajęcia nr 12

PROGRAM KOMPUTEROWY:

```
//Połączenia:
//zworka JP4 (GRAPH DISPLAY)
//
//LCD_GRAPH(D0) -PA0
//LCD_GRAPH(D1) -PA1
//LCD_GRAPH(D2) -PA2
//LCD_GRAPH(D3) -PA3
//LCD_GRAPH(D4) -PA4
//LCD_GRAPH(D5) -PA5
//LCD_GRAPH(D6) -PA6
//LCD_GRAPH(D7) -PA7
//LCD_GRAPH(RS) -PB0
//LCD_GRAPH(RW) -PB1
//LCD_GRAPH(E)  -PB2
//LCD_GRAPH(CS1)-PB4
//LCD_GRAPH(CS2)-PB3
//RxD - PD0
//TxD - PD1
//
//Na komputerze:
//Uruchomić HyperTerminal
//Dowolna nazwa połączenia
//COM1
//bitrate 4800b/s
//8bitów
//brak bitów parzystości
//1 bit stopu
//brak sterowania przepływem


#define F_CPU 16000000
#define BAUD(x) (((F_CPU/16)/x))

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "KS0108.h"
#include "usart.h"

int main()
{
	DDRD = 0xff;
	char x[2];//=" ";
	x[1]='\0';
	GLCD_Initialize();
	USART_Init(BAUD(4800));
	while(1)
	{
		GLCD_ClearScreen();
		int i=0;
		int j=0;
		while(j<8)
		{
			while(i<21)
			{
				x[0] = USART_GetChar();
				GLCD_WriteStringXY(x,6*i++,j);
			}
			j++;
			i=0;
		}
	}
	return 0;
}
```

ZADANIA:

Na podstawie programu stwórz parser NMEA0183, który odczyta szerokość i długość geograficzną.

3.13. Zajęcia nr 13

PROGRAM KOMPUTEROWY:

```
//Połączenia:
//zworka JP4 (GRAPH DISPLAY)
//
//LCD_GRAPH(D0) -PA0
//LCD_GRAPH(D1) -PA1
//LCD_GRAPH(D2) -PA2
//LCD_GRAPH(D3) -PA3
//LCD_GRAPH(D4) -PA4
//LCD_GRAPH(D5) -PA5
//LCD_GRAPH(D6) -PA6
//LCD_GRAPH(D7) -PA7
//LCD_GRAPH(RS) -PB0
//LCD_GRAPH(RW) -PB1
//LCD_GRAPH(E)  -PB2
//LCD_GRAPH(CS1)-PB4
//LCD_GRAPH(CS2)-PB3
//RxD - PD0
//TxD - PD1
//
//Na komputerze:
//Uruchomić HyperTerminal
//Dowolna nazwa połączenia
//COM1
//bitrate 4800b/s
//8bitów
//brak bitów parzystości
//1 bit stopu
//brak sterowania przepływem


#define F_CPU 16000000
#define BAUD(x) (((F_CPU/16)/x))

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "KS0108.h"
#include "usart.h"

int main()
{
	DDRD = 0xff;
	char x[2];//=" ";
	x[1]='\0';
	GLCD_Initialize();
	USART_Init(BAUD(4800));
	while(1)
	{
		GLCD_ClearScreen();
		int i=0;
		int j=0;
		while(j<8)
		{
			while(i<21)
			{
				x[0] = USART_GetChar();
				GLCD_WriteStringXY(x,6*i++,j);
			}
			j++;
			i=0;
		}
	}
	return 0;
}
```

ZADANIA:

Na podstawie programu stwórz parser NMEA0183, który odczyta 10 wybranych informacji.

3.14. Zajęcia nr 14

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Net;

namespace udp_komunikator
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            CheckForIllegalCrossThreadCalls = false;
            bw = new BackgroundWorker();
            bw.DoWork += new DoWorkEventHandler(bw_DoWork);
            bw.RunWorkerAsync();
        }
        private BackgroundWorker bw;
        private void button1_Click(object sender, EventArgs e)
        {
            byte[] buffer = System.Text.Encoding.ASCII.GetBytes(String.Format("{0}:{1}", textBox1.Text,
textBox2.Text));
            UdpClient udpClient = new UdpClient();
            udpClient.Send(buffer, buffer.Length, new IPEndPoint(IPAddress.Broadcast, 23456));
            udpClient.Close();
            textBox2.Text = "";
        }
        private void bw_DoWork(object sender, DoWorkEventArgs ea)
        {
            UdpClient udpClient = new UdpClient(23456);
            while (true)
            {
                IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);
                byte[] buffer = udpClient.Receive(ref RemoteIpEndPoint);
                if (buffer != null)
                {
                   // string[] read = System.Text.Encoding.ASCII.GetString(buffer).Split(':');
                    richTextBox1.Text += String.Format("{0}\n",
System.Text.Encoding.ASCII.GetString(buffer));
                }
            }
        }
    }
}
```

ZADANIA:

Uruchom program napisany w C# w MS Visual Studio. Przeanalizuj linia po linii.

3.15. Zajęcia nr 15

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;

namespace tcpip_client
{
    public partial class Form1 : Form
    {
        private int port;
        private IPAddress ip;
        private byte[] data = new byte[256];
        private string tekst=string.Empty;
        private TcpClient client;
        private NetworkStream stream;
        private BackgroundWorker bw;
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            button1.Enabled = false;
            try
            {
                ip = IPAddress.Parse(textBox1_IP.Text);
                port = int.Parse(textBox2_port.Text);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Błąd w IP");
            }
            try
            {
                bw = new BackgroundWorker();
                bw.DoWork += new DoWorkEventHandler(bw_DoWork);
                bw.RunWorkerAsync();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Błąd w połączeniu");
            }
        }
        private void bw_DoWork(object sender, DoWorkEventArgs e)
        {
            try
            {
                while (true)
                {
                    client = new TcpClient(ip.ToString(), port);
                    stream = client.GetStream();
                    while (true)
                    {
                        try
                        {
                            data = System.Text.Encoding.ASCII.GetBytes(tekst);
                            stream.Write(data, 0, data.Length);
                            System.Threading.Thread.Sleep(3000);
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show("Błąd");
                        }
                    }
```

```
                }
                }
            catch(Exception ex)
            {
                MessageBox.Show("Inny Błąd");
            }
        }

        private void richTextBox1_TextChanged(object sender, EventArgs e)
        {
            tekst = richTextBox1.Text;
        }
    }
}
```

ZADANIA:

Uruchom program napisany w C# w MS Visual Studio. Przeanalizuj linia po linii.

4. WARUNKI ZALICZENIA PRZEDMIOTU

Zaliczenie praktyczne

5. PROGRAM ZAJĘĆ

5.1. MSZ

1) Budowa mostka zintegrowanego IBS na przykładzie rzeczywistej instalacji na statku szkoleniowym AM Nawigator XXI,
2) Budowa sieci lokalnej z użyciem rzeczywistych interfejsów komputerowych i symulatorów urządzeń w laboratorium IBS,
3) Projektowanie w CAD mostka na rzeczywisty statek (gazowiec lub inny specjalistyczny), dobór urządzeń i ich połączeń
4) Elementy ergonomiczne MSZ - IBS na przykładzie projektu w programie ERGO na rzeczywisty statek
5) Podstawowe diagnostyki MSZ-IBS
6) Własny projekt IBS na zaliczenie z rzeczywistym interfejsem komputeorywm

5.2. PTD

1) Budowa systemu interfejsu szeregowego RS-232C, podstawowe podłączenia i układy sygnałowe
2) Integracja Systemu RS232
3) Budowa interfejsu równoległego IEEE-488(IEC-625), budowa rozproszonego system pomiarowy z interfejsem IEEE-488,
4) Budowa sieci CAN i Profibus – przykład mostka zintegrowanego
5) Bezprzewodowy system akwizycji danych – oparty o GSM, radiomodemy, bluetooth, oraz IEE802.11
6) Lokalna sieć komputerowa na przykładzie sterowników LAN i konwerterów LAN -> transmisja szeregowa – zintegrowana sieć statkowa.
7) Budowa własnego systemu pomiarowego na zalicznie

6. INFORMACJE DODATKOWE

7. LITERATURA

Literatura podstawowa

1. Specht C., *System GPS*, Biblioteka Nawigacji nr 1, Wydawnictwo "Bernardinum", Pelplin. 2007
2. Hofmann-Wellenhof B., Lichtenegger H., Collins J., (1997) GPS Theory and practice, Fourth edition, Springer, Wien New York.
3. Spilker J, Parkinson B., i in, *Global Positioning System: Theory and Applications*, AIAA, 1996.
4. Haykin S., *Systemy telekomunikacyjne cz. I i II*, WKŁ, Warszawa, 1998.
5. Wesołowski K., *Systemy radiokomunikacji ruchomej*, WKŁ, Warszawa, 1999.
6. Hołubowicz W., Płóciennik P., Różański A., *Systemy łączności bezprzewodowej*, Poznań, 1997.
7. Simmonds A., *Wprowadzenie do transmisji danych*, WKŁ, Warszawa, 1999.
8. *RTCM Recommended Standards for Differential GNSS (Global Navigation Satellite Systems) Service*, Version 2.3 (RTCM Paper 136-2001/SC104-STD), 2001
9. *SPS Global Positioning System (GPS), Standard Positioning Service, Signal Specification*, Department of Defence, Positioning/Navigation Executive Commitee, November 5, 1993
10. *GPS Global Positioning System Standard Positioning Service*, Performance Standard, Assistant Secretary of Defence, October, 2001
11. *ICD-GPS - 200 – Interface Control Document*, Washington DC.

Literatura uzupełniająca

1. Wells (red), *Guide to GPS Positioning*, Canadian GPS Associates, Fredericton, 1987.
2. Killen H. B., *Transmisja cyfrowa w systemach* światłowodowych i satelitarnych, WKŁ, Warszawa, 1983.
3. Wojnar A., *Systemy radiokomunikacji ruchomej lądowej*, WKŁ, Waszawa, 1989.

8. FORMULARZE, ZAŁĄCZNIKI