Rzeszów, czerwiec 2020

Materiały do przedmiotu "Sterowniki mikroprocesorowe"

Opracował: Zbigniew Świder

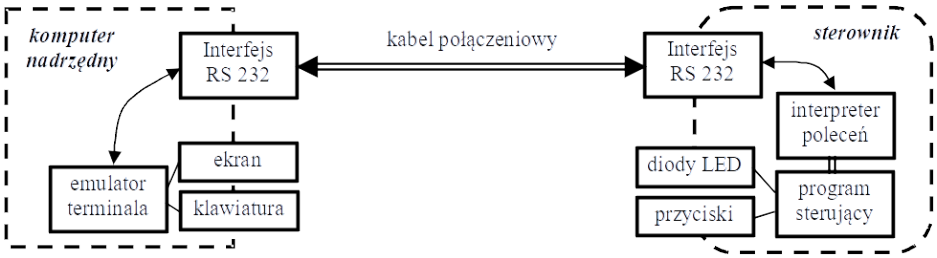
Komunikacja, cz. I

1. Pojęcia podstawowe

Mikroprocesorowe systemy wbudowane zwykle nie mają rozbudowanych interfejsów komunikacji z użytkownikiem. Najczęściej wyposażone są w pewną liczbę diod sygnalizacyjnych, przycisków czy przełączników, mogą także posiadać wyświetlacz LCD. Zwykle należy więc zapewnić możliwość komunikacji komputera nadrzędnego (uruchomieniowego) z docelowym systemem wbudowanym. Często jest to spowodowane koniecznością testowania prawidłowości działania sprzętu, potrzebą konfigurowania, testowanie czy też uruchamiania oprogramowania w sterowniku.

Transmisja szeregowa asynchroniczna - przesyłane są dane poprzedzone bitem startu (stan logiczny: 0), który jest sygnałem rozpoczęcia transmisji, następnie bity danych od najmłodszego do najstarszego (zwykle 5 do 8), potem bit parzystości (opcjonalnie), a na koniec odstęp (bity stopu) przed następnym znakiem (stan logiczny: 1). Każdy znak jest przesyłany i interpretowany niezależnie od innych.

Komunikacja poprzez interfejs RS232 - jest zwykle stosowana do wymiany danych z systemami wbudowanymi. Zaletą takiego rozwiązania jest jego prostota i łatwość oprogramowania. Układ UART (ang. *Universal Asynchronous Receiver Transmitter*) sterujący transmisją szeregową jest prosty i można go spotkać nawet w najprostszych mikrokontrolerach. Maksymalna szybkość transmisji uzależniona jest od sprzętu, a także od długości kabla połączeniowego, a przy krótkim kablu szybkość transmisji może osiągać nawet 115200 znaków na sekundę



Emulator terminala szeregowego - to program użyty po stronie komputera nadrzędnego, na przykład *Hyper Terminal* lub też *putty*. Jego zadaniem jest wysyłanie do sterownika (poprzez łącze RS 232) kolejnych znaków wprowadzanych z klawiatury oraz wyświetlanie na ekranie komputera (terminala) przychodzących znaków (wysyłanych przez sterownik).

Protokół komunikacyjny - to zbiór reguł i zasad, które są stosowane przez urządzenia komunikacyjne (komputer, sterownik) w celu wymiany danych. Protokoły zwykle składają się z trzech części:

- procedury powitalnej, czyli początkowe informacje, w tym np. rodzaj przesyłanych danych,
- właściwego przekazu danych (ciąg danych o stałej lub zmiennej długości),
- pól analizy poprawności przekazu (np. sum kontrolnych) połączonej z procedurą pożegnania.

W systemach sterowania przemysłowego stosowane są protokoły: Modbus, DNP 3.0, Profibus, OPC, IEC 60870-5 oraz IEC 61850. Jednak w systemach prototypowych (w laboratorium) można zaimplementować własny (uproszczony) protokół lub tez wysyłać pojedyncze znaki (bez protokołu).

Zadanie 1. Rozpatrzmy prosty układ z klawiszem i diodą (rozwiązany już wcześniej na wykładzie) połączony z terminalem szeregowym (komputerem z zainstalowanym programem *putty*). Naciśnięcie przycisku KL powoduje zaświecenie diody na 3 sekundy. Jeśli teraz w trakcie świecenia puścimy przycisk, to dioda gaśnie (świecenie jest przerywane). Zamiast fizycznego klawisza (przypisanego do *aK1*) wykorzystamy komunikację szeregową z przesyłaniem pojedynczych znaków (bez protokołu). Wysłanie z konsoli do sterownika znaku '1' oznacza wciśnięcie klawisza (*KL==1*) natomiast wysłanie znaku '0' oznacza puszczenie klawisza (*KL==0*).

Rozwiązanie.

Przebiegi czasowe, graf oraz opis stanów przedstawiono na wcześniejszym wykładzie.

W komunikacji będziemy wykorzystywać trzy funkcje (dostępne w dodatkowym module *serial.c*):

- *void* ***UART_init****(int baud)* - ustawia parametry łącza, w tym szybkość przesyłania danych,
- *void* ***COM_send****(char c)* - wysyła jeden znak ze sterownika do terminala (komputera),
- *char* ***COM_recv()*** - odczytuje jeden znak wysłany z komputera do sterownika lub wartość 0 (*null*), gdy od ostatniego odczytu nie wysłano żadnego nowego znaku

Funkcję *UART_init*() wywołujemy raz na początku (przed główną pętlą *while*) dla zainicjowania łącza komunikacyjnego i ustawienia parametrów transmisji (*baud*), funkcję *COM_send*() stosujemy do wysłania znaków ze sterownika do komputera (i wyświetlenia ich na ekranie terminala), natomiast za pomocą funkcji *COM_recv*() odczytujemy znaki wysyłane z klawiatury komputera i przesyłane do sterownika (płytki). Pamiętajmy, że **każdy odczyt** funkcją *COM_recv*() zeruje status odczytu (ustawia flagę "*odczytano znak, czekaj na następny*"), więc zwykle przypisujemy wartość tej funkcji do zmiennej (np. znak) i tylko tej zmiennej używany w dalszej części programu.

Program w C będzie więc wyglądał następująco (zwróćmy szczególną uwagę na różnicę pomiędzy zapisem **'1'** (w apostrofach, znak) a **1** (bez apostrofów, liczba):

```
// ---------------------------
char KL, LD, stan=1, tim, znak;  // definicje zmiennych roboczych
#include "serial.c"      // wczytanie pliku z funkcjami do komunikacji
// ---------------------------
UART_init(9600);         // inicjowanie łącza (9600 bit/s)
// ---------------------------
znak=COM_recv();         // odczyt odebranego znaku (komunikacja)
if (znak=='1')  KL=1;    // jeśli odebrano znak '1', to ustaw KL na 1
if (znak=='0')  KL=0;    // jeśli odebrano znak '0', to ustaw KL na 0

switch(stan) {           // ------ główny graf sterowania ---------
 case 1: LD=0;
         if (KL) { tim=30; stan=2; }    // zmienna KL==1
         break;
 case 2: LD=1;
         if (!tim) stan=3;       // skończył się czas -> 3
         else
         if (!KL) stan=1;        // zmienna KL==0     -> 1
         break;
 case 3: LD=0;
         if (!KL) stan=1;        // zmienna KL==0     -> 1
         break;
}
if (tim) --tim;                  // dekrementuj timer co cykl
L1=LD;                           // i ustaw wyjścia
```

Zadanie 2. Rozpatrzmy to samo zadanie, co w przykładzie 1, ale teraz do komunikacji zastosujemy własny protokół (a nie pojedyncze znaki '0' i '1' jak poprzednio). Protokoły niech będą następujące:

':'	KL	!KL	'#'						- ustawianie wartości KL (na 0 lub 1)
':'	T1	cs	'A'	'#'					- T1 - czas świecenia diody (1...9 s), warunek: *T1 + cs == 10*
':'	'0'	'#'	→		':'	LD	!LD	'#'	- odesłanie stanu diody LD (0 lub 1)

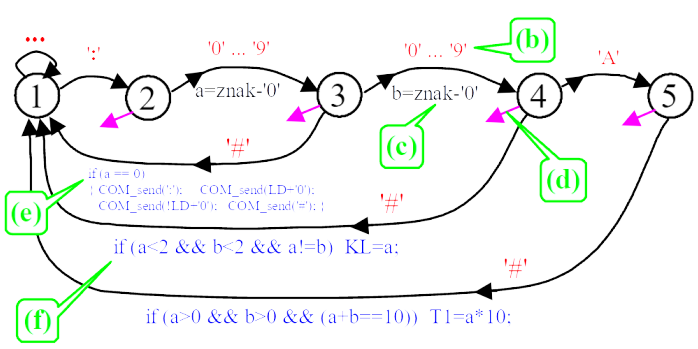
Tak więc wysłanie z komputera do sterownika komunikatu [: 1 0 #], czyli kolejno naciskamy na klawiaturze komputera znak dwukropka (':'), cyfrę jeden ('1'), cyfrę zero ('0') oraz znak *hash* ('#'), spowoduje w sterowniku ustawienie zmiennej KL na wartość 1. Wysłanie z komputera do sterownika komunikatu [: 5 5 A #], czyli kolejno naciskamy na klawiaturze komputera znak dwukropka (':'), cyfrę cztery ('4'), cyfrę sześć ('6'), literę A duże ('A') oraz na koniec znak *hash* ('#'), spowoduje w sterowniku ustawienie zmiennej T1 (czasy świecenia diody) na wartość 40 (czyli *4 sekundy*). Suma kontrolna (zmienna *cs*) jest obliczana tak, aby był spełniony warunek podany w zadaniu, czyli *T1 + cs == 10* (bo 4 + 6 == 10). Natomiast wysłanie komunikatu [: 0 #], czyli kolejno znaku dwukropka (':'), cyfry zero ('0'), oraz znaku *hash* ('#'), spowoduje odesłanie komunikatu zawierającego informację o stanie diody LD, na przykład komunikat [: 1 0 #] będzie oznaczał, że dioda świeci (LD==1).

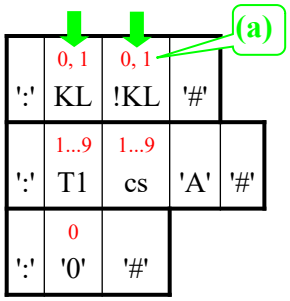
Odebranie przez sterownik błędnego (niezgodnego z protokołem) komunikatu nie powoduje żadnej akcji (jest w całości ignorowane) i sterownik oczekuje na kolejny komunikat. Na przykład wysłanie komunikatu [: 0 0 #] jest ignorowane, bo nie pasuje do żadnego komunikatu z zadania.

Analizę kolejno nadsyłanych do sterownika znaków (a więc i protokołów) zrealizujemy za pomocą grafu. W odróżnieniu od głównego grafu sterującego (wywoływanego co cykl, czyli co 0.1 sekundy), **graf komunikacyjny jest wywoływany dopiero, gdy przyjdzie (zostanie odebrany) nowy znak**.

Następuje wtedy jego zapamiętanie, analiza (zgodnie z założonym protokołem), a na końcu podjęta odpowiednia akcja (np. ustawienie KL lub czasu T1), lub też zignorowanie całego komunikatu (gdy wystąpił błąd, np. źle wprowadzona dana lub przekłamanie w transmisji).

Rozwiązanie. W tym zadaniu graf komunikacyjny będzie następujący:





(*zakresy od-do*)

(a) - Do tabeli komunikatów wpisujemy zakresy zmiennych (wartości *od-do*), na przykład dla zmiennej KL są to wartości 0 lub 1, natomiast dla T1 wartości od 1 do 9 (sekund).

(b) - Nad strzałką wpisujemy wartości zmiennej *znak*, np. '0'...'9' oznacza, że przejście ze stanu 3 do stanu 4 nastąpi, gdy zostanie odebrany znak pomiędzy znakiem '0' a znakiem '9' (włącznie).

(c) - Operacja *znak-'0'* oznacza **zamianę znaku na liczbę**, czyli do zmiennej *b* wpisujemy albo liczbę 0 (gdy *znak=='0'*) lub liczbę 1 (gdy *znak=='1'*).

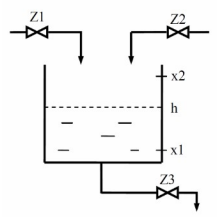
(d) - Krótka strzałka przy stanie oznacza powrót do stanu 1, gdy odebrano inny (błędny) znak.

(e) - Jeśli odebranym komunikatem był [: 0 #], czyli wartość a == 0, to odeślij komunikat do komputera [: LD !LD #] wykorzystując funkcję *COM_send*() oraz **zamianę liczby na znak**, czyli dodając do liczby (zmiennej) kod '0' (np. 1 + '0' daje znak '1', a 9 + '0' daje znak '9').

(f) - Tutaj sprawdzany jest zakres wartości *od-do* oraz dodatkowe warunki (np. suma kontrolna) i jeśli cały komunikat jest poprawny, to następuje akcja (tutaj ustawienie *KL* na wartość *a*).

Program w C będzie więc wyglądał następująco:

```
// ---------------------------
char KL, LD, stan=1, tim;           // definicje zmiennych roboczych
char znak, stank=1, a, b, T1=30;    // w tym czas T1 (początkowo 3 sekundy)
#include "serial.c"         // wczytanie pliku z funkcjami do komunikacji
// ---------------------------
UART_init(9600);            // inicjowanie łącza (9600 bit/s)
// ---------------------------
znak=COM_recv();            // odczyt odebranego znaku (komunikacja)

if (znak != 0)              // jeśli przyszedł nowy znak  (!!!)
 switch (stank) {           // to uruchom graf komunikacyjny --------
  case 1: if (znak==':')  stank=2;
          break;
  case 2: if (znak>='0' && znak<='9') { a=znak-'0'; stank=3; }
          else stank=1;           // błąd - krótka strzałka na grafie
          break;
  case 3: if (znak>='0' && znak<='9') { b=znak-'0'; stank=4; }
          else if (znak=='#') )           // odebrano znak '#'
           { if (a==0)
              { COM_send(':');  COM_send(LD+'0');  // wyślij komunikat
                COM_send(!LD+'0'); COM_send('#'); }
             stank=1; }                            // i stank na 1
          else stank=1;        // błąd - krótka strzałka na grafie
          break;
  case 4: if (znak=='A') stank=5;         // odebrano znak 'A'
          else if (znak=='#')             // odebrano znak '#'
           { if (a<2 && b<2 && a!=b)  KL=a;       // ustaw KL
             stank=1; }                           // i stank na 1
          else stank=1;        // błąd - krótka strzałka na grafie
          break;
  case 5: if (znak=='#')                  // odebrano znak '#'
           { if (a>0 && b>0 && (a+b==10))  T1=a*10;   // ustaw T1
             stank=1; }                           // i stank na 1
          else stank=1;        // błąd - krótka strzałka na grafie
          break;
 }

switch(stan) {              // ------ główny graf sterowania ---------
 case 1: LD=0;
         if (KL) { tim=T1; stan=2; }         // ustaw tim na T1
         break;
 case 2: LD=1;
         if (!tim) stan=3;          // skończył się czas -> 3
         else
         if (!KL) stan=1;           // zmienna KL==0     -> 1
         break;
 case 3: LD=0;
         if (!KL) stan=1;           // zmienna KL==0     -> 1
         break;
}
if (tim) --tim;                     // dekrementuj timer co cykl
L1=LD;                              // i ustaw wyjścia
```



Zadanie 3. Zaprojektuj układ sterowania według algorytmu:

h < x1 - nalewamy z obu Z1+Z2 do x1

h ≥ x1 - dalej nalewamy tylko z Z1 do x2 w cyklu 3+2 s (**T1**+**T2**)

h ≥ x2 - zamykamy, czekamy 5 s (**T3**)

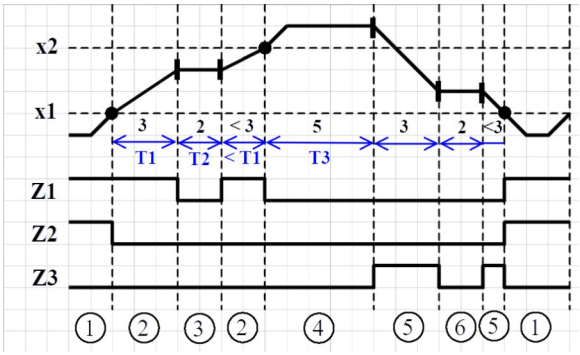
- wylewamy z Z3 do x1 w cyklu 3+2s.

Zakresy czasów: T1, T2 = 1...6 s, T3 = 3...9 s

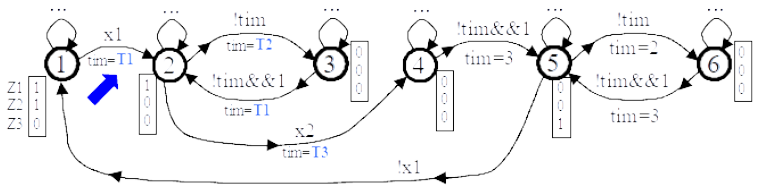
':'	x1	x2	cs	'#'									- ustawienie x1, x2, warunek: (x1+x2+cs) parzyste
':'	T1	T2	cs	'0'	'#'								- ustawienie T1, T2, warunek: (T1+T2+cs)%5 == 0
':'	T3	cs	'#'										- ustawienie T3, warunek: (T3+cs) == 9
':'	'0'	'#'	→	':'	Z1	Z2	Z3	cs	'#'				(Z1+Z2+Z3+cs) parzyste
':'	'1'	'#'	→	':'	T1	T2	T3	cs	'#'				(T1+T2+T3+cs)%5 == 0

Rozwiązanie:

Przebiegi czasowe, opis stanów, graf sterujący oraz zakresy *od-do* dla komunikacji:

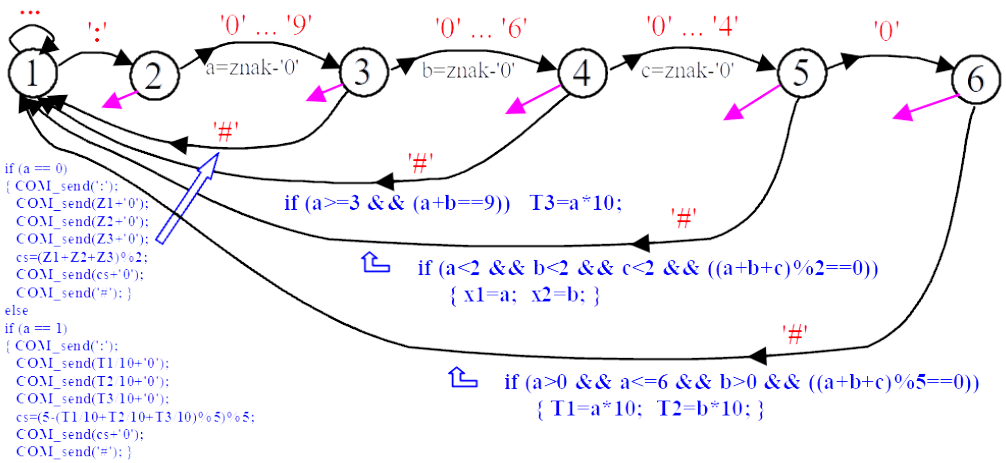


':'	0, 1 x1	0, 1 x2	0, 1 cs	'#'
':'	1...6 T1	1...6 T2	0...4 cs	0 '0'	'#'
':'	3...9 T3	0...6 cs	'#'	
':'	'0'	'#'		
':'	'1'	'#'		



1 - nalewanie z obu (Z1 – Z2)
2 - nalewanie z Z1 maksymalnie T1
3 - pauza w nalewaniu dokładnie T2
4 - oczekiwanie (pauza) dokładnie T3
5 - wylewanie z Z3 maksymalnie 3 s
6 - pauza w wylewaniu dokładnie 2 s

Graf komunikacyjny (zwróćmy uwagę na sprawdzanie ograniczeń i sum kontrolnych):

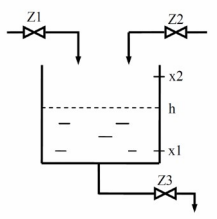


Program w C będzie więc wyglądał następująco:

```
// ------------------------------------------------------------------------
char x1, x2, Z1, Z2, Z3, stan=1, tim;   // definicje zmiennych dla sterowania
char znak, stank=1, a, b, c, cs;        // definicje zmiennych dla komunikacji
char T1=30, T2=20, T3=50;    // definicje i inicjowanie czasów T1, T2, T3
#include "serial.c"         // wczytanie pliku z funkcjami do komunikacji
// ------------------------------------------------------------------------
UART_init(9600);           // inicjowanie łącza (9600 bit/s)
// ------------------------------------------------------------------------
znak=COM_recv();           // odczyt odebranego znaku --------- (komunikacja)
if (znak != 0)             // -------- jeśli przyszedł nowy znak  (!!!)
 switch (stank) {          // -------- to uruchom graf komunikacyjny --------
  case 1: if (znak==':')  stank=2;
          break;
  case 2: if (znak>='0' && znak<='9') { a=znak-'0'; stank=3; }
          else stank=1;           // błąd - krótka strzałka na grafie
          break;
  case 3: if (znak>='0' && znak<='6') { b=znak-'0'; stank=4; }
          else if (znak=='#') )                          // odebrano znak '#'
           { if (a==0)
              { COM_send(':');    COM_send(Z1+'0');        // wyślij komunikat 1
                COM_send(Z2+'0'); COM_send(Z3+'0');
                cs=(Z1+Z2+Z3)%2;  COM_send(cs+'0');
                COM_send('#'); }
             else if (a==1)
              { COM_send(':');       COM_send(T1/10+'0');  // wyślij komunikat 2
                COM_send(T2/10+'0'); COM_send(T3/10+'0');
                cs=(5-(T1/10+T2/10+T3/10)%5)%5;
                COM_send(cs+'0');    COM_send('#'); }
            stank=1; }                            // i stank na 1
          else stank=1;        // błąd - krótka strzałka na grafie
          break;
  case 4: if (znak>='0' && znak<='4') { c=znak-'0'; stank=5; }
          else if (znak=='#')              // odebrano znak '#'
           { if (a>=3 && (a+b==9))  T3=a*10;      // ustaw T3
             stank=1; }                           // i stank na 1
          else stank=1;        // błąd - krótka strzałka na grafie
          break;
  case 5: if (znak=='0') stank=6;
          else if (znak=='#')              // odebrano znak '#'
           { if (a<2 && b<2 && c<2 && ((a+b+c)%2==0))
              { x1=a;  x2=b; }                    // ustaw x1, x2
             stank=1; }                           // i stank na 1
          else stank=1;        // błąd - krótka strzałka na grafie
          break;
  case 6: if (znak=='#')                   // odebrano znak '#'
           { if (a>0 && a<6 && b>0 && ((a+b+c)%5==0))
              { T1=a*10; T2=b*10; }               // ustaw T1, T2
             stank=1; }                           // i stank na 1
          else stank=1;        // błąd - krótka strzałka na grafie
          break;
 }
```

```
switch(stan) {     // ------ główny graf sterowania (co 0.1 s) ---------
 case 1: Z1=1; Z2=1; Z3=0;
         if (x1)    { tim=T1; stan=2; }        // ustaw timer na T1
         break;

 case 2: Z1=1; Z2=0; Z3=0;
         if (!tim) { tim=T2; stan=3; }        // skończył się czas   -> 3
         else                                 // albo
         if (x2)    { tim=T3; stan=4; }       // napełniono zbiornik -> 4
         break;

 case 3: Z1=0; Z2=0; Z3=0;
         if (!tim&&1) { tim=T1; stan=2; }     // koniec pauzy   -> 2
         break;

 case 4: Z1=0; Z2=0; Z3=0;
         if (!tim&&1) { tim=30; stan=5; }     // oczekiwanie dokładnie 5 s
         break;

 case 5: Z1=0; Z2=0; Z3=1;
         if (!tim) { tim=20; stan=6; }        // skończył się czas -> 6
         else                                 // albo
         if (!x1)  stan=1;                    // opróżniono zbiornik -> 1
         break;

 case 6: Z1=0; Z2=0; Z3=0;
         if (!tim&&1) { tim=30; stan=5; }     // koniec pauzy   -> 5
         break;

}  //-------------------------------------------------------------------

if (tim) --tim;                               // dekrementuj timer co cykl

L1=Z1; L2=Z2; L3=Z3;                          // zaświeć diody LED

// ===========================================================================

=============================================================================
```

Czasy dwucyfrowe (1...15 s)



Zadanie 4a. Zaprojektuj układ sterowania według algorytmu:

h < x1 - nalewamy z obu Z1+Z2 do x1

h ≥ x1 - dalej nalewamy tylko z Z1 do x2 w cyklu 3+2 s (**T1**+**T2**)

h ≥ x2 - zamykamy, czekamy 5 s (**T3**)

- wylewamy z Z3 do x1 w cyklu 3+2s.

Zakresy czasów: T1, T2 = 1...6 s, T3 = **1...15 s** ⇦

	':'	x1	x2	cs	'#'			- ustawienie x1, x2, warunek: (x1+x2+cs) parzyste				
	':'	T1	T2	cs	'0'	'#'		- ustawienie T1, T2, warunek: (T1+T2+cs)%5 == 0				
⇨	':'	T3H	T3L	cs	'1'	'#'		- ustawienie T3, warunek: (T3+cs) %5 == 0				
	':'	'0'	'#'	→	':'	Z1	Z2	Z3	cs	'#'		(Z1+Z2+Z3+cs) parzyste
	':'	'1'	'#'	→	':'	T1	T2	cs	'#'			(T1+T2+T3+cs)%5 == 0
⇨	':'	'2'	'#'	→	':'	T3H	T3L	cs	'#'			(T3_cs)%5 == 0

Czas T3 (1...15 s) jest zapisany na dwóch znakach: cyfra dziesiątek T3H (0,1) oraz cyfra jednostek T3L (0...9), stąd: T3 = 10 * T3H + T3L

Sprawdzenie zakresu wartości i sumy kontrolnej – pamiętamy, że zawsze **sprawdzamy całą liczbę**, a nie jej poszczególne cyfry:

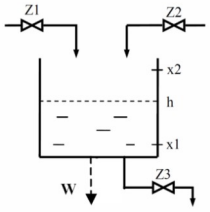
```
if (... && (a==0||a==1) && (10*a+b)>=1 && (10*a+b)<=15 && (10*a+b+c)%5==0)

    T3 = (10*a+b)*10;
```

Wypisanie czasu T3 (wypisanie cyfry dziesiątek oraz jednostek):

```
COM_send(T3/100+'0');     COM_send((T3/10)%10+'0');
```

Uwagi:
- definiujemy TYLKO zmienną T3 (nie wolno definiować T3H, T3L - one są obliczane 'w locie')
- sprawdzamy cały zakres liczby T3 (a więc liczbę, a nie oddzielnie jej poszczególne cyfry)
- wypisując funkcją COM_send() rozkładamy 'w locie' liczbę T3 na cyfrę dziesiątek i cyfrę jednostek, a następnie je osobno wypisujemy

Zadanie 4b. (do samodzielnego rozwiązania). Zaprojektuj układ sterowania + komunikacja:



h < x1 - nalewamy z obu Z1+Z2 do x1

h ≥ x1 - dalej **na przemian** Z1/Z2 do x2 **w cyklu** 3+3 s (T1+T2),

- wylewamy Z3 do x1 **ale nie krócej** niż 5 s (T3)

Zakresy czasów: T1, T2 = **1...6 s**, **T3 = 1...14 s** ⇐

':'	x1	x2	'0'	cs	'#'								(x1+x2+cs)==3
':'	T1	T2	'1'	cs	'#'								(T1+T2+cs)%4==2
':'	**T3H**	**T3L**	'A'	cs	'#'								(T3+cs)%4==1
':'	'0'	'#'		→		':'	Z1	Z2	Z3	cs	'#'		(Z1+Z2+Z3+cs) nieparzyste
':'	'1'	'#'		→		':'	T1	T2	'2'	cs	'#'		(T1+T2+T3+cs)%4 == 0
':'	'2'	'#'				':'	**T3H**	**T3L**	'3'	cs	'#'		(T3+cs)%4==1