Wojskowa Akademia Techniczna im. Jarosława Dąbrowskiego

Laboratorium z przedmiotu:

Systemy wbudowane

Sprawozdanie z ćwiczenia laboratoryjnego nr 5:

Sterowanie mocą, transmisja danych

Prowadzący:
mgr inż. Artur Miktus

Wykonał: Radosław Relidzyński
Nr albumu: 76836
Grupa: WCY20IY4S1
Data laboratoriów: 03.06.2022 r.
Deklarowana ocena: 3, 4, 5

Spis treści

A. Treść zadania

Zadanie na Laboratorium nr 5

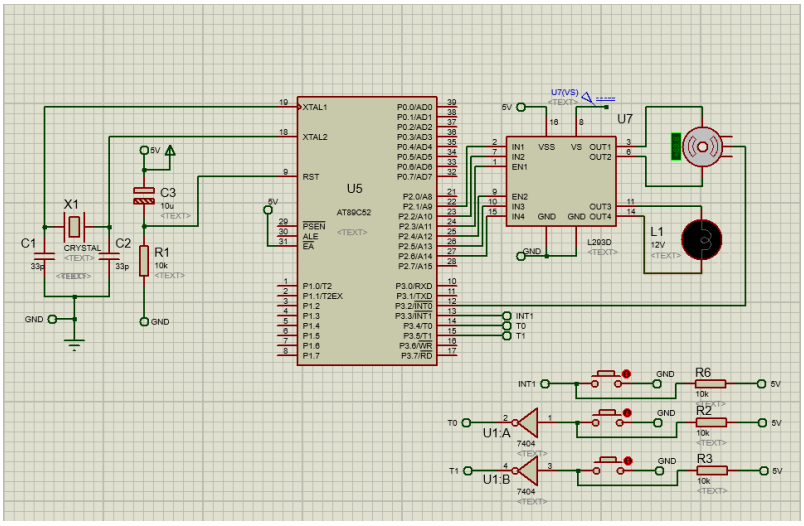
Zadanie na dostatecznie.

Na podstawie przykładowego programu ze strony o PWM:

napisać program **zad1.c** w języku C na układ z projektu PWM_1A.pdsprj (w razie potrzeby zbudować taki układ samodzielnie w starszej wersji Proteusa) tak, aby na wyjściu P2_0 uzyskać falę prostokątną : o **okresie** równym około (**10 ms x numer_w_dzienniku**) i współczynniku wypełnienia równym **((3 x numer_w_dzienniku) +10) %**.

W sprawozdaniu należy przedstawić stosowne obliczenia, niezbędne do wykonania zadania oraz zrzuty ekranowe z programu Proteus, wykorzystujące ekran oscyloskopu tak, aby graficznie za pomocą pomiarów wykorzystujących funkcję Cursors oscyloskopu (awaryjnie podstawę czasu oscyloskopu i liczbę kratek, zajmowanych przez wykres badanego przebiegu), udowodnić poprawną realizację zadania. Obliczenia, dotyczące **okresu należy wyróżnić w edytorze zielonym kolorem tła**, a obliczenia dotyczące **współczynnika wypełnienia żółtym kolorem tła**. W podpisie rysunku muszą być zawarte wyniki pomiaru, podane jawnie (nie za pomocą zwrotu - wyniki pomiaru przedstawiono na rysunku!).

Schemat układu na dobrze i bardzo dobrze.



Zadanie na dobrze.

To, co na dostatecznie i ponadto:

Dla układu z projektu, przedstawionego powyżej:

napisać program **zad2.c** w języku C, który spowoduje, że jednokrotne naciśnięcie przycisku, **podłączonego do pinu P3.3 (INT1):**

a) jeśli silnik nie pracował i lampa L1 nie świeciła
=== załączy silnik tak, aby wirnik obracał się z prędkością około

(20% + [70% / mniej znacząca cyfra numeru w dzienniku]) prędkości maksymalnej

---

czyli dla nr 1, 11 i 21 wartość ta ma być równa (20% + [70% / 1]) = 20% + 70 % = 90%,
dla numeru 2, 12 i 22 wartość ta ma być równa (20% + [70% / 2]) = 20% + 35% = 55%,

...

dla numeru 9, 19 wartość ta ma być równa (20% + [70% / 9]) = 20% + 7% = 27%.
Numery 10 i 20, jako wyjątkowe przy tej regule, mają przyjąć wartość 22%.

Po uruchomieniu symulacji w Proteusie **wirnik silnika ma się nie obracać** i lampa L1 **ma nie świecić**.

W sprawozdaniu należy pisemnie **określić i udokumentować sposób wyznaczenia prędkości maksymalnej wirnika silnika (opis eksperymentu, program użyty do wyznaczenia tej prędkości).**

Studenci o **numerach nieparzystych** "obracają" wirnik silnika **tak, aby na wyświetlaczu silnika w czasie jego pracy pokazywana była liczba dodatnia.**
Studenci o **numerach parzystych** - "obracają" wirnik silnika **tak, aby na wyświetlaczu silnika w czasie jego pracy pokazywana była liczba ujemna.**

W sprawozdaniu należy zamieścić niezbędne obliczenia, uzasadniające przyjęte wartości liczbowe i zrzut ekranu z symulacji, pokazujący osiągniętą prędkość obrotów.
Lampa L1 ma być zasilana dokładnie tak samo, jak silnik.

b) jeśli silnik pracował i lampa L1 świeciła,
=== wyłączy oba urządzenia.

Poza typowymi elementami sprawozdania w zadaniu na dobrze należy opisać
a) budowę,
b) przeznaczenie,
c) sposób działania

elementu L293D, zastosowanego w powyższym schemacie (przykład w linku). W sprawozdaniu trzeba też opisać, jak autor programu wykorzystuje piny: In1, In2, En1, Out1, Out2 w działaniu swojego programu.

Proszę zwrócić uwagę, że przeznaczenie pinów jest bardzo konkretne i **zamiana ich roli** spowoduje **niezaliczenie** tego zadania.

Należy podać, **odczytane z dokumentacji**, maksymalne wartości natężeń prądów i napięć zasilających odbiornik np. silnik.

W sprawozdaniu w osobnym punkcie, zatytułowanym "OPIS POMYSŁU NA REALIZACJĘ na db" koniecznie musi wystąpić opis pomysłu na realizację zadania i jego algorytm - bez nich nawet poprawnie działający program na ocenę dobrą nie zostanie zaliczony.

Zadanie na bardzo dobrze

To, co na dobrze i ponadto dla powyższego układu:
(uwaga - wartość parametrów zadania - dla studentów o numerach **nieparzystych N=2, K=3**, dla studentów o numerach **parzystych N=3, K=2**)

Stosując obsługę wejść T0 i T1 napisz program **zad3.c**, który spowoduje, że

1. wciśnięcie **N razy przycisku T0 (P3_4) (bez żadnego wciśnięcia w tym czasie T1)** włącza silnik,
2. wciśnięcie **K razy przycisku T1 (P3_5) (bez żadnego wciśnięcia w tym czasie T0)** silnik wyłącza.

   Uwaga - **wciśnięcie T0 kasuje dotychczasowy stan wciśnięć T1 i odwrotnie**.
3. Przycisk **/INT1 zmienia naprzemiennie** kierunek obrotu silnika.
4. Lampa L1 ma przez cały czas pracy programu **pozostać wyłączona**.

   Prędkość obrotowa silnika i początkowy kierunek obrotów mają być dla danego studenta takie, jak w zadaniu na dobrze.

W sprawozdaniu w osobnym punkcie, zatytułowanym "OPIS POMYSŁU NA REALIZACJĘ na bdb" koniecznie musi wystąpić opis pomysłu na realizację zadania i jego algorytm - bez nich nawet poprawnie działający program na ocenę bardzo dobrą nie zostanie zaliczony.

B. Zadanie na ocenę dostateczną

Opis mojego rozwiązania

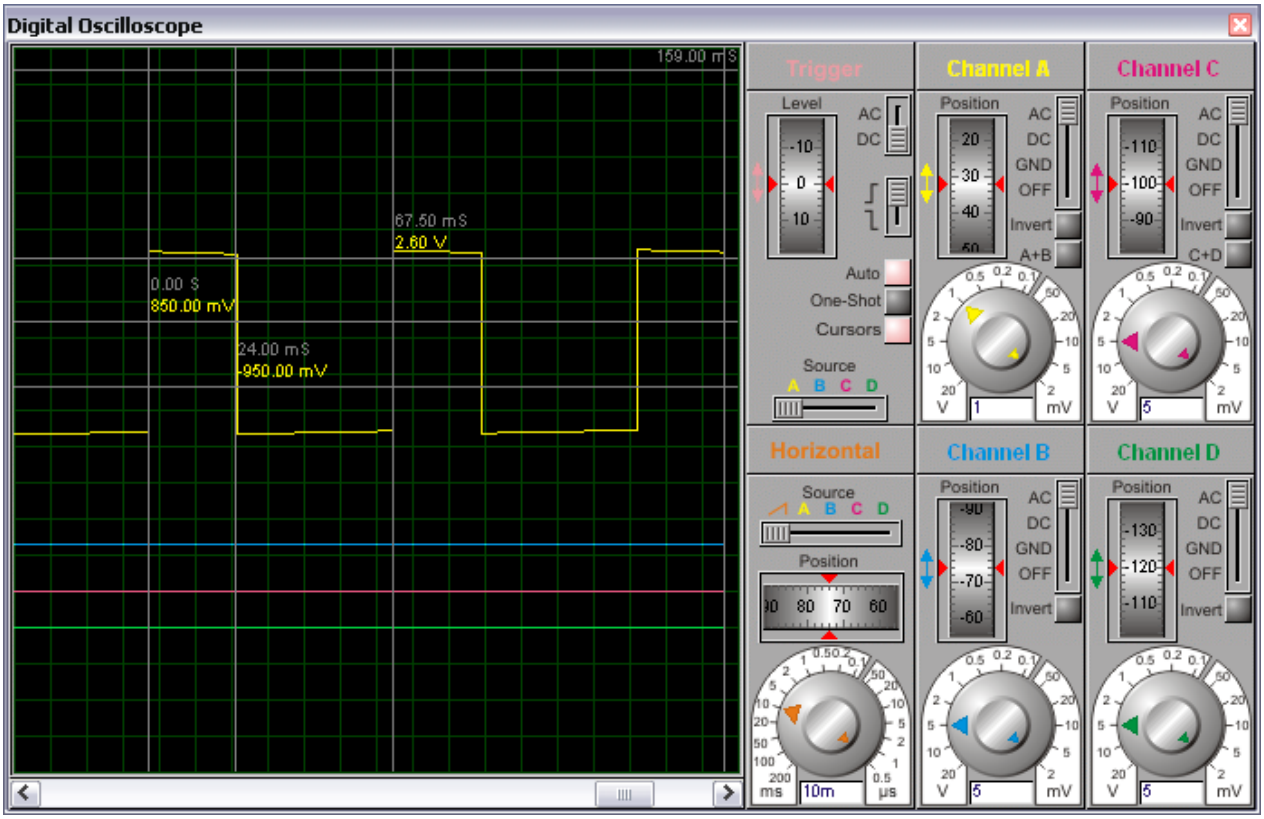
Oczekiwany okres: 10 ms x 9 = 90 ms

Oczekiwany współczynnik wypełnienia: 3% * 9 + 10% = 37%

Okres wyznaczam na podstawie wartości zmiennej „PWM_Freq_Num”

Współczynnik wypełnienia wyznaczam na podstawie wartości zmiennej „PWM” będącej iloczynem liczby 255 i współczynnika: PWM = 255 * 0,37 = 94,35 ≈ 94

Na początku sprawdzam działanie dla obliczonego PWM równego 94 i dla PWM_Freq_Num = 30:

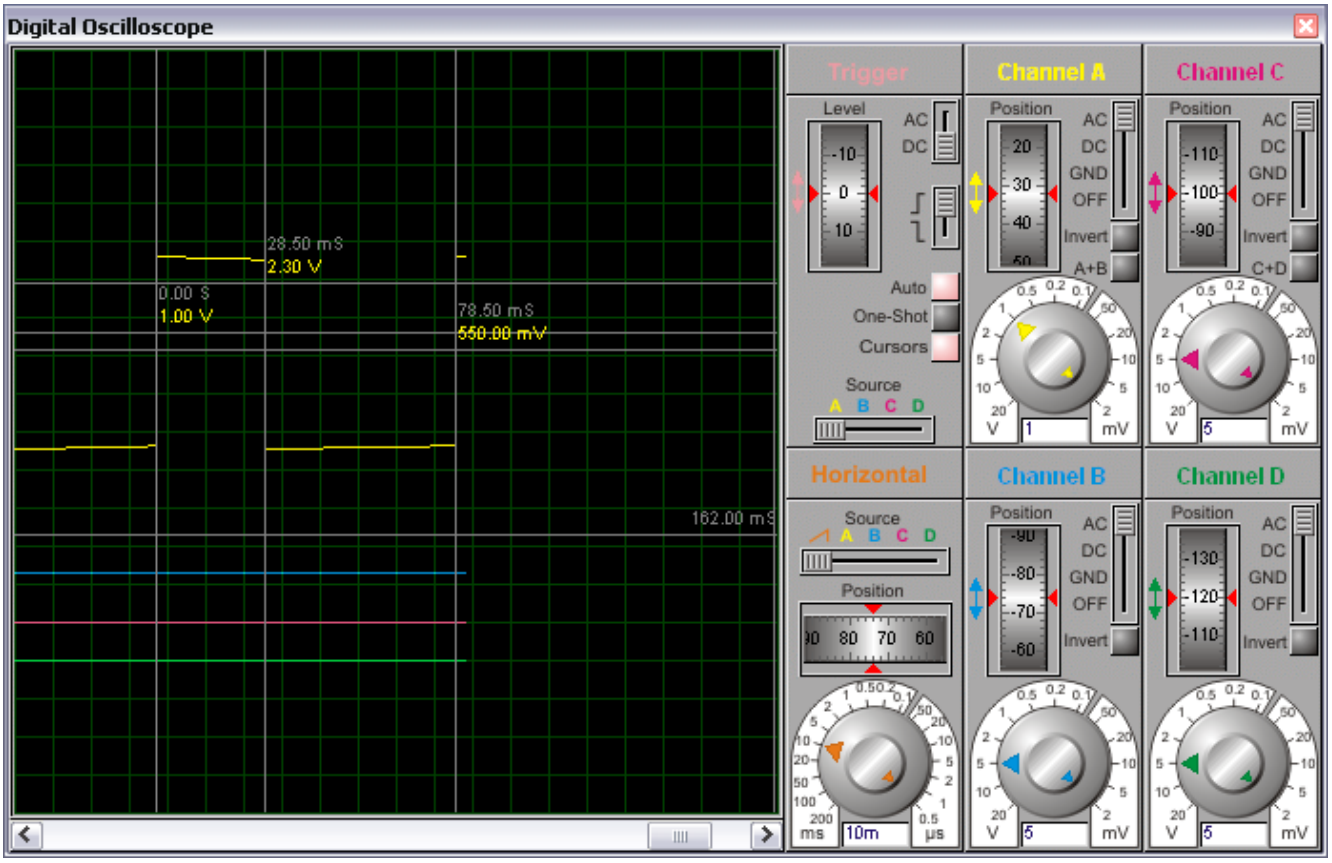


Okres = 67,5 – 0 = 67,5 ms (za mały)

Czas trwania jedynki: 24 - 0 = 24 ms

Współczynnik wypełnienia: 24 ms / 67,5 ms = 0,355556

Współczynnik się prawie zgadza, ale okres jest znacznie za mały. Zmieniam wartość PWM_Freq_Num na 35:



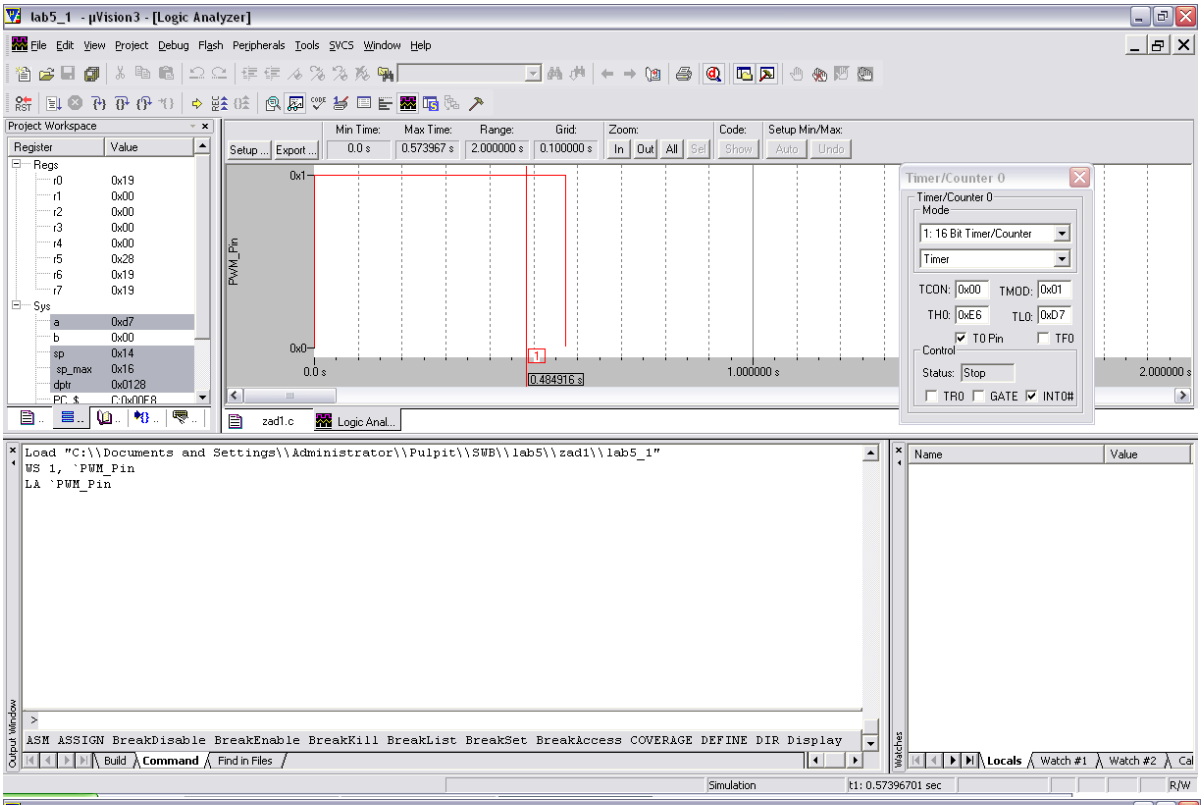
Okres = 78,5 – 0 = 78,5 ms (za mały)

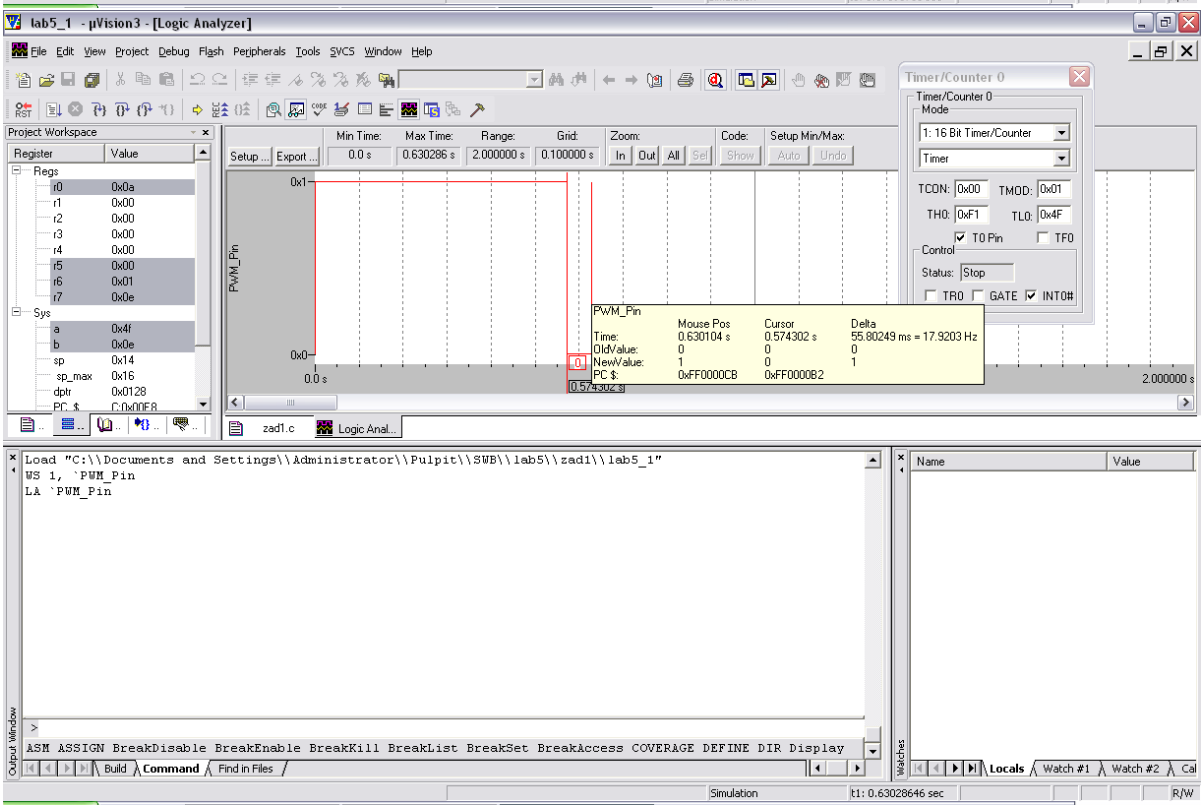
Mając wyniki dla 2 wartości, mogę obliczyć wartość oczekiwaną dla mojego okresu:

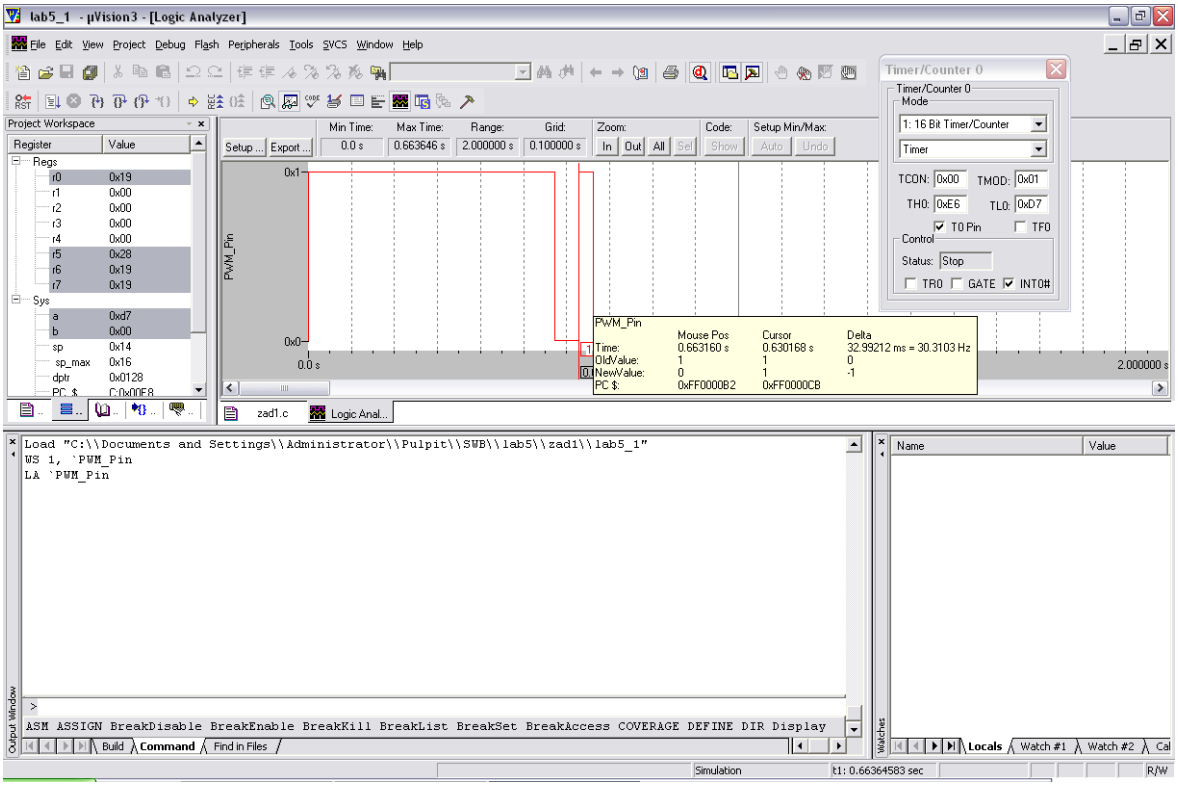
PWM_Freq_Num = oczekiwany okres * (PWM_Freq_Num1 – PWM_Freq_Num2) / (okres1 – okres2)

PWM_Freq_Num = 90 * (35-30) / (78,5 – 67,5) ≈ 40

Sprawdzam zatem dla wartości PWM = 94 i dla PWM_Freq_Num = 40:


lab5_1 - µVision3 - [Logic Analyzer]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register Value
Regs
r0 0x19
r1 0x00
r2 0x00
r3 0x00
r4 0x00
r5 0x28
r6 0x19
r7 0x19
Sys
a 0xd7
b 0x00
sp 0x14
sp_max 0x16
dptr 0x0128
Min Time: Max Time: Range: Grid: Zoom: Code: Setup Min/Max:
0.0 s 0.573967 s 2.000000 s 0.100000 s
PWM_Pin
0.0 s
0.484916 s
1.000000 s
2.000000 s
Timer/Counter 0
Timer/Counter 0
Mode
1: 16 Bit Timer/Counter
Timer
TCON: 0x00 TMOD: 0x01
TH0: 0xE6 TL0: 0xD7
T0 Pin TF0
Control
Status: Stop
TR0 GATE INT0#
zad1.c
Logic Anal...
Load "C:\\Documents and Settings\\Administrator\\Pulpit\\SWB\\lab5\\zad1\\lab5_1"
WS 1, `PWM_Pin
LA `PWM_Pin
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display
Build Command Find in Files
Name Value
Locals Watch #1 Watch #2
Simulation t1: 0.57396701 sec
R/W


lab5_1 - µVision3 - [Logic Analyzer]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register Value
Regs
r0 0x0a
r1 0x00
r2 0x00
r3 0x00
r4 0x00
r5 0x00
r6 0x01
r7 0x0e
Sys
a 0x4f
b 0x0e
sp 0x14
sp_max 0x16
dptr 0x0128
Min Time: Max Time: Range: Grid: Zoom: Code: Setup Min/Max:
0.0 s 0.630286 s 2.000000 s 0.100000 s
PWM_Pin
0.0 s
2.000000 s
PWM_Pin
Mouse Pos Cursor Delta
Time: 0.630104 s 0.574302 s 55.80249 ms = 17.9203 Hz
OldValue: 0 0 0
NewValue: 1 0 1
PC $: 0xFF0000CB 0xFF0000B2
Timer/Counter 0
Timer/Counter 0
Mode
1: 16 Bit Timer/Counter
Timer
TCON: 0x00 TMOD: 0x01
TH0: 0xF1 TL0: 0x4F
T0 Pin TF0
Control
Status: Stop
TR0 GATE INT0#
zad1.c
Logic Anal...
Load "C:\\Documents and Settings\\Administrator\\Pulpit\\SWB\\lab5\\zad1\\lab5_1"
WS 1, `PWM_Pin
LA `PWM_Pin
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display
Build Command Find in Files
Name Value
Locals Watch #1 Watch #2
Simulation t1: 0.63028646 sec
R/W



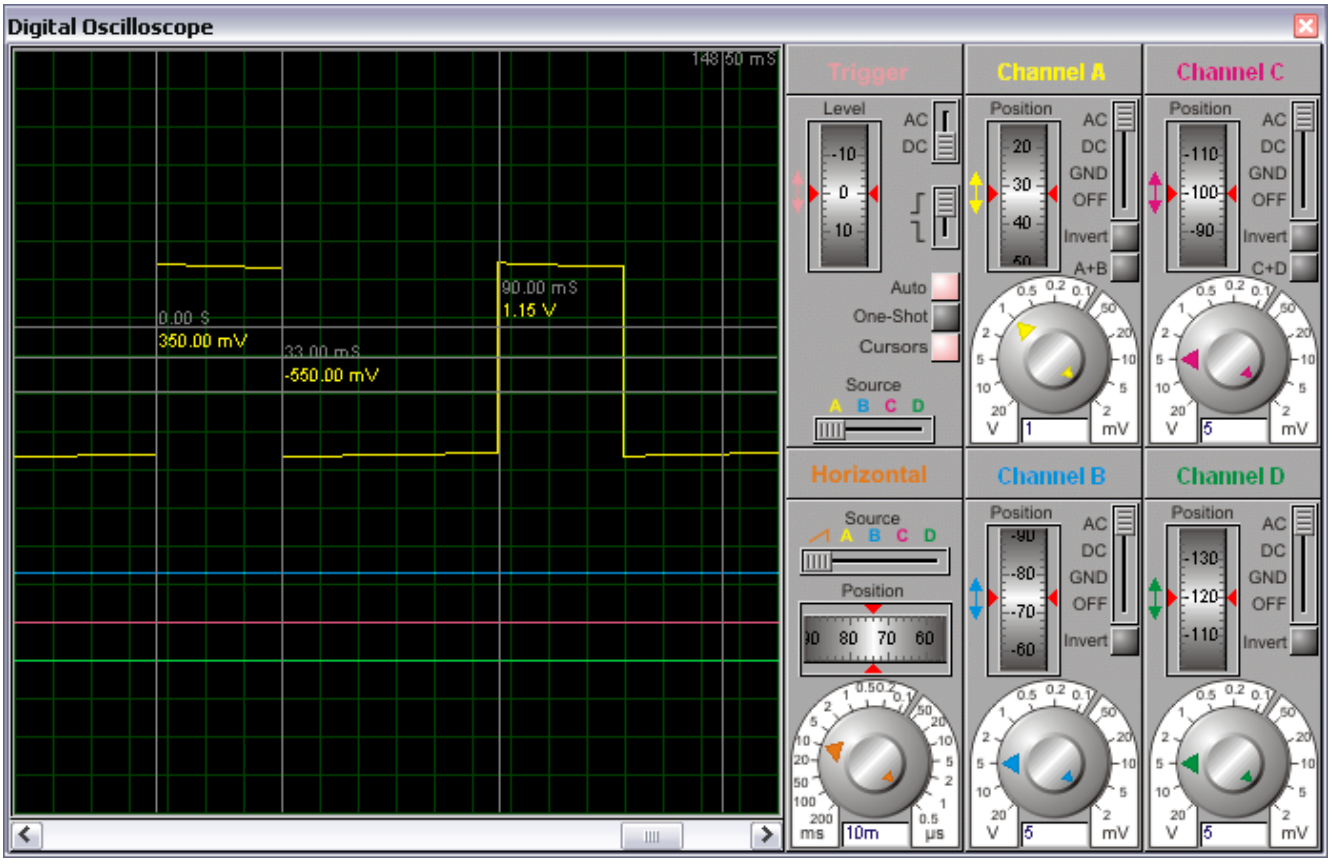
Okres = 56 ms + 33 ms = 89 ms

Czas trwania jedynki: 33 ms

Współczynnik wypełnienia: 33 ms / 98 ms = 0,37078 ≈ 0,37 = 37%

Wartości TH i TL w zależności od stanu:

Stan	TH0	TL0
0	0xF1	0x4F
1	0xE6	0xD7



Okres = 90 ms – 0 ms = 90 ms

Czas trwania jedynki: 33 ms – 0 ms = 33 ms

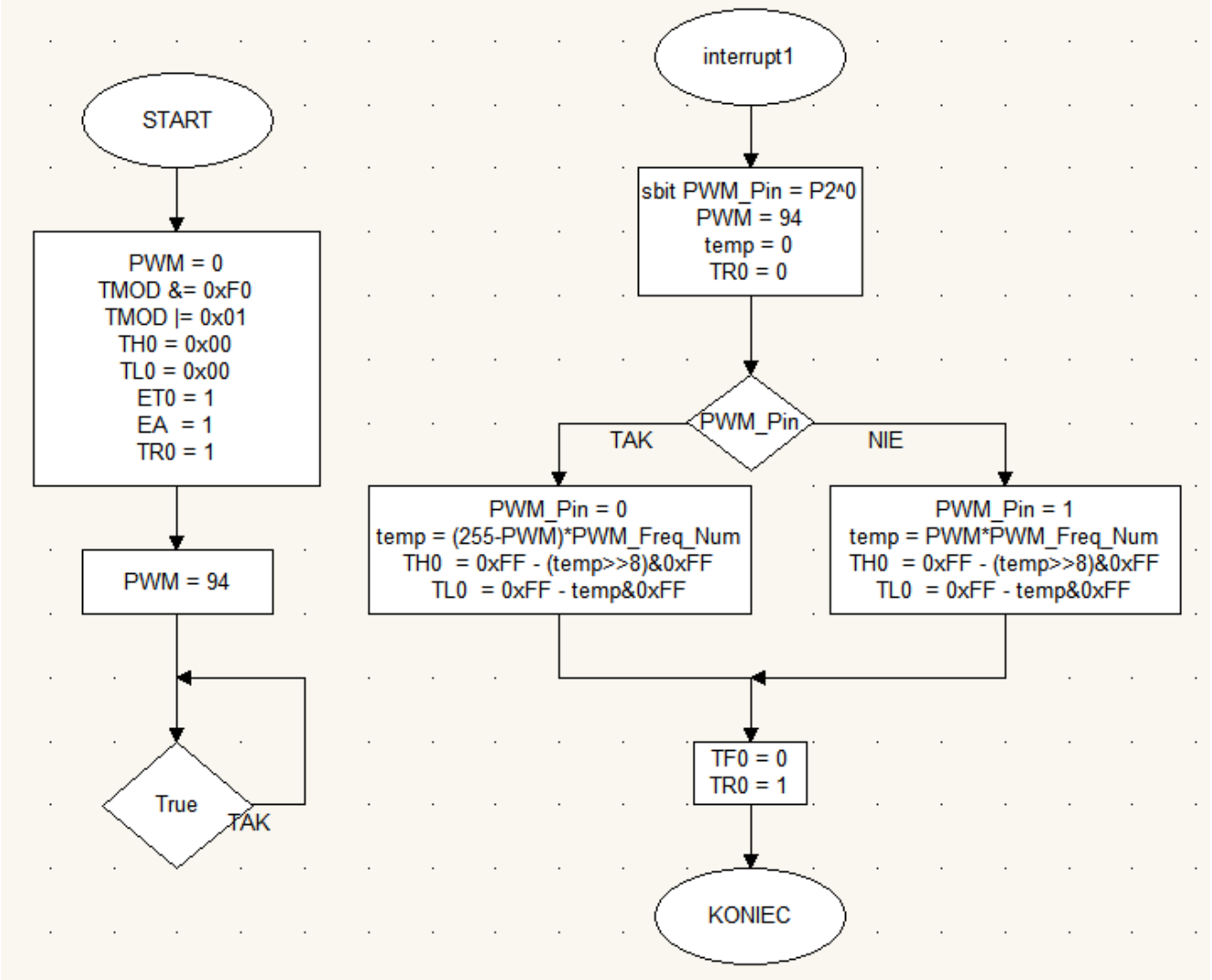
Współczynnik wypełnienia: 33 ms / 90 ms = 0,366667 ≈ 0,37 = 37%

Wyniki są zgodne z oczekiwaniami, więc są to właśnie te wartości (PWM = 94, PWM_Freq_Num = 40).

Działanie programu opiera się na nieskończonej pętli w funkcji main() podczas której obsługiwane będą przerwania (funkcją Timer0_ISR). Każde przerwanie będzie oparte na poniższej liście kroków:

1. Zatrzymanie zegara (żeby zapobiec wystąpienia kolejnego przerwania).
2. Cyklicznej zmianie sygnału w zależności od wartości zmiennej „PWM_Pin”:
    a. Jeśli 1, to wyzeruj sygnał (0 na wykresie oscylatora). Na podstawie okresu oblicza czas podawania stanu 0 (100% duty cycle - PWM)
    b. Jeśli 0, to ustaw pin na 1 (1 na wykresie oscylatora). Na podstawie okresu oblicza czas podawania stanu 1 (PWM).
3. Załadowuje odpowiedni licznik na podstawie wspomnianych w punkcie 2a lub 2b.
4. Kończy przerwanie.
5. Startuje timer.

Schemat blokowy rozwiązania



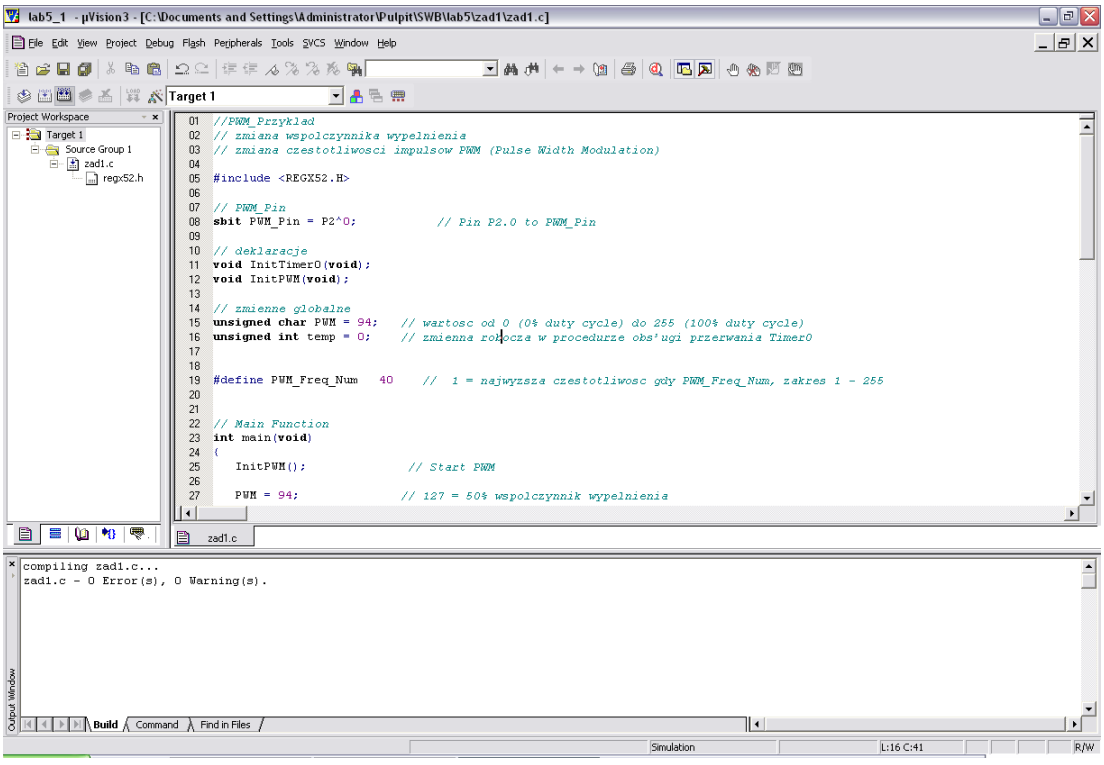
Listing programu

```
//PWM_Przyklad
// zmiana wspolczynnika wypelnienia
// zmiana czestotliwosci impulsow PWM (Pulse Width Modulation)

#include <REGX52.H>

// PWM_Pin
sbit PWM_Pin = P2^0;           // Pin P2.0 to PWM_Pin

// deklaracje
void InitTimer0(void);
void InitPWM(void);

// zmienne globalne
unsigned char PWM = 94;   // wartosc od 0 (0% duty cycle) do 255 (100% duty cycle)
unsigned int temp = 0;    // zmienna robocza w procedurze obsługi przerwania Timer0


#define PWM_Freq_Num   40    //  1 = najwyzsza czestotliwosc gdy PWM_Freq_Num, zakres
1 - 255


// Main Function
int main(void)
{
   InitPWM();              // Start PWM

   PWM = 94;              // 127 = 50% wspolczynnik wypelnienia

   while(1) {;}
}


// Timer0 init
void InitTimer0(void)
{
    TMOD &= 0xF0;    // wyzeruj bity dla Timer0
    TMOD |= 0x01;    // ustaw tryb mode 1 = 16bit mode

    TH0 = 0x00;      // Pierwsze
    TL0 = 0x00;      // ustawienie

    ET0 = 1;         // Enable Timer0 interrupts
    EA  = 1;         // Enable All

    TR0 = 1;         // Start Timer 0
}

// PWM init
void InitPWM(void)
{
    PWM = 0;         // poczatkowo zero
```

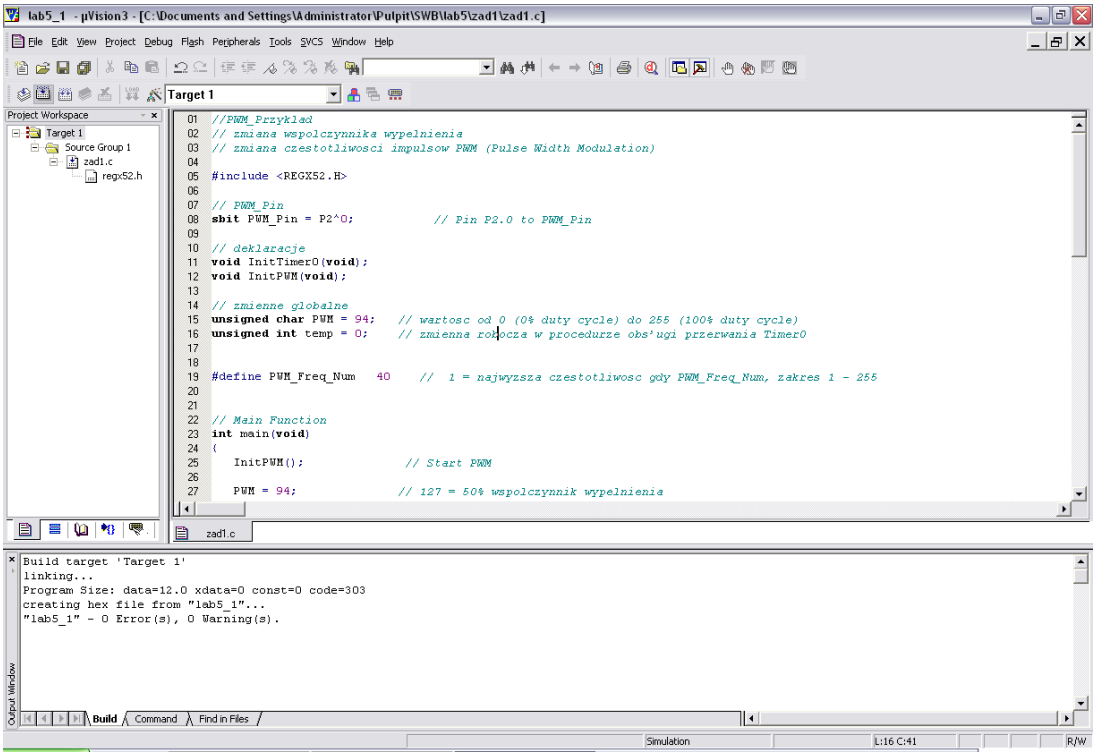
```
    InitTimer0();    // Init Timer0 dla rozpoczecia generacji przerwan
}

// Timer0 ISR
void Timer0_ISR (void) interrupt 1
{
    TR0 = 0;    // Stop Timer 0

    if(PWM_Pin) // if PWM_Pin =1 wyzeruj sygnal PWM i zaladuj licznik
    {
        PWM_Pin = 0;
        temp = (255-PWM)*PWM_Freq_Num;
        TH0  = 0xFF - (temp>>8)&0xFF;
        TL0  = 0xFF - temp&0xFF;
    }
    else         // if PWM_Pin =0 ustaw pin na 1 i zaladuj licznik
    {
        PWM_Pin = 1;
        temp = PWM*PWM_Freq_Num;
        TH0  = 0xFF - (temp>>8)&0xFF;
        TL0  = 0xFF - temp&0xFF;
    }

    TF0 = 0;     // wyczysc flage
    TR0 = 1;     // Start Timer 0
}
```

Sprawdzenie poprawności

Kompilowanie

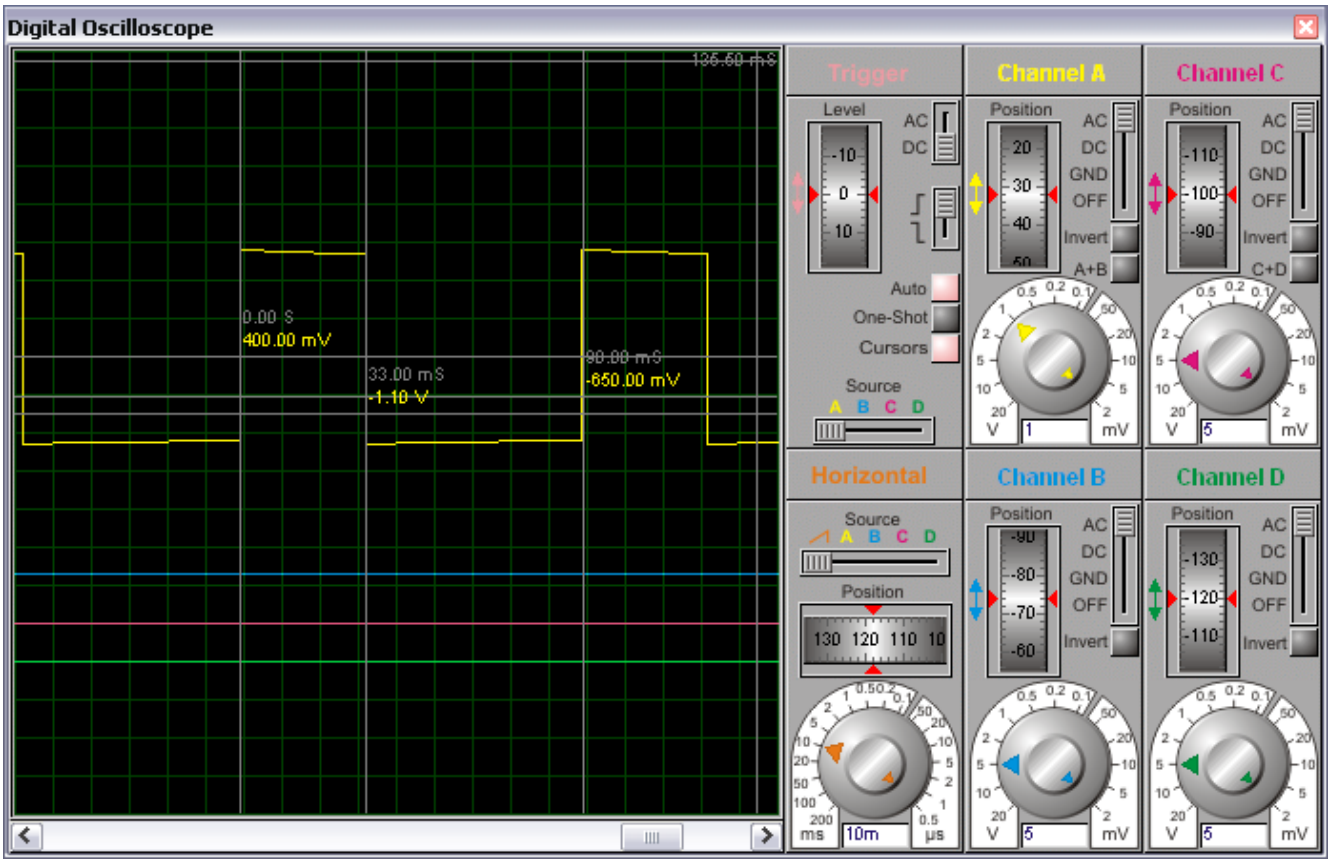


Linkowanie



Prezentacja realizacji zadania przez program

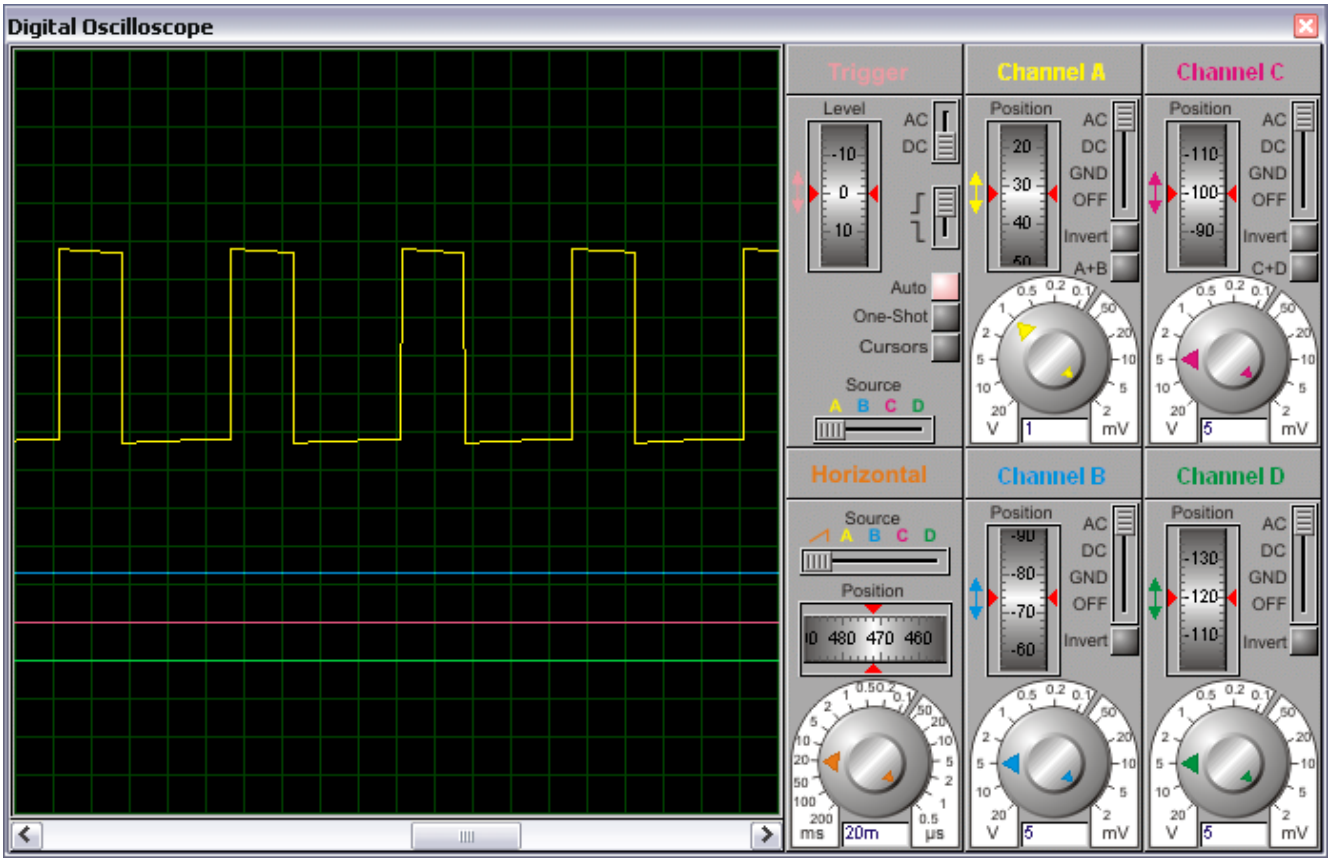
Ocena czasów:



Okres = 90 ms – 0 ms = 90 ms

Współczynnik wypełnienia: 33 ms / 90 ms = 0,366667 ≈ 0,37 = 37%

Widać, że stan 1 to około 1/3 całego okresu, ponieważ stan 0 trwa około 2 razy dłużej:



C. Zadanie na ocenę dobrą

Opis mojego rozwiązania

L293D to dwukanałowy układ scalony będący sterownikiem silników stałoprądowych. Stosuje się go przy napięciach do 36 V i natężeniach do 0,6 A.

W bieżącym układzie sterownik L293D będzie odpowiadał za odbiór sygnału wejściowego i przetworzenia go na wyjście tak, aby sterowało ono silnikiem. Używane będą następujące łącza:

- IN1, IN2 – wskazywać będą kierunek obrotu silnika

- OUT1, OUT2 – podawać będą odpowiednie napięcie do silnika, w zależności od wejścia EN1

- EN1 – odbiera on zmienny sygnał.

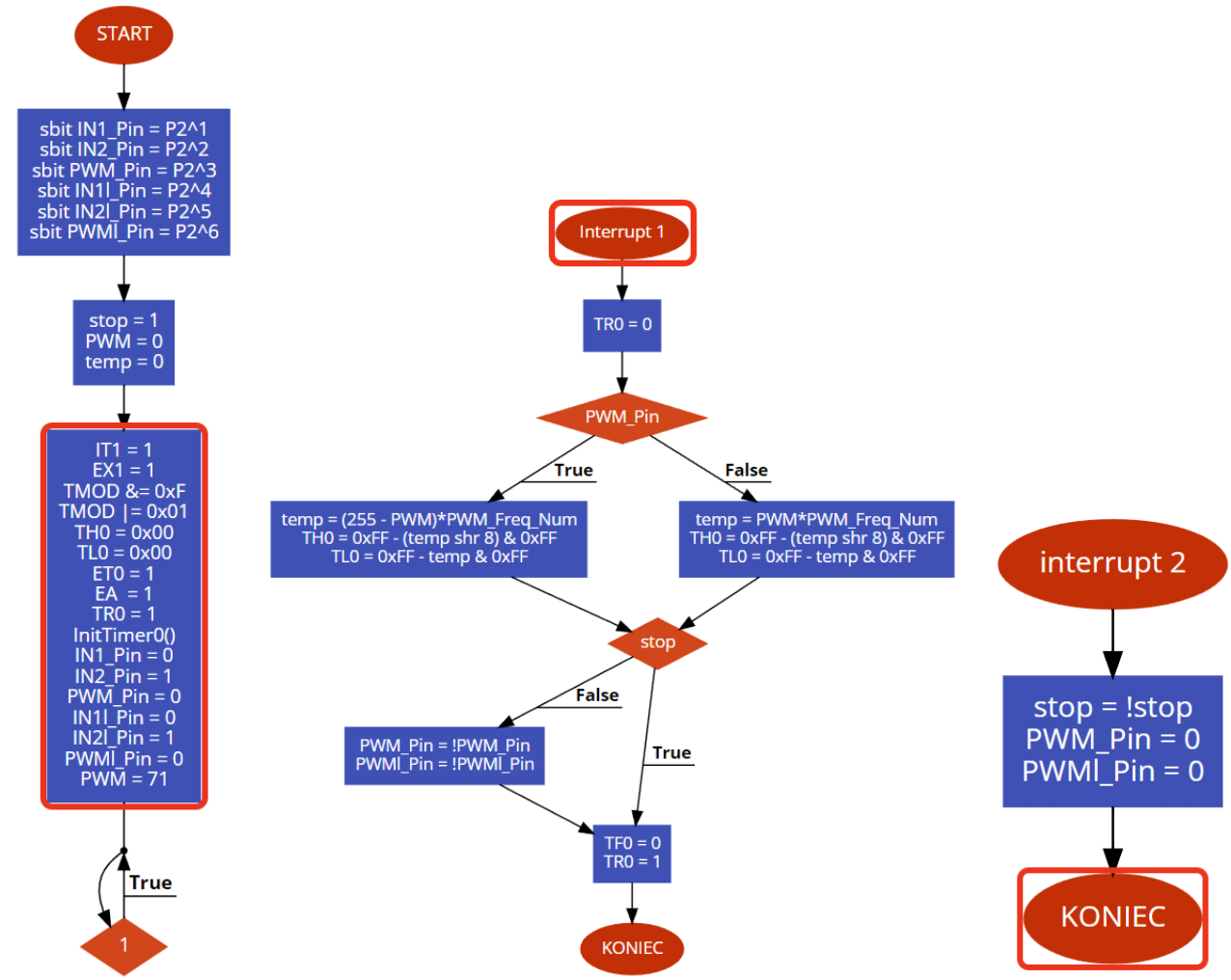
- VSS, GND – źródła zasilania / uziemienia układu

Układ odbiera na EN1 zmienny sygnał. Na podstawie współczynnika wypełnienia będzie wprost proporcjonalnie określać wysokość napięcia (maksymalne dla PWM=255, zerowe dla PWM=0). Wynika to z dużej częstotliwości układu. Taki sygnał będzie przekazywany na wyjścia OUT.

Prędkość silnika: 20% + 70% / 9 = 27%
Wartość PWM: 255 * 27% = 69 (wartość na silniku ma pokazywać wartość dodatnią)
Aby silnik pokazywał wartość dodatnią, należy na wartość IN1 wysłać sygnał 0, a na IN2 wysłać sygnał 1.

Timer 0 będzie służył do generowania syngału PWM, a timer1 do przechwytywania wciśnięć przycisku poprzez interrupt.

Schemat blokowy rozwiązania



Listing programu

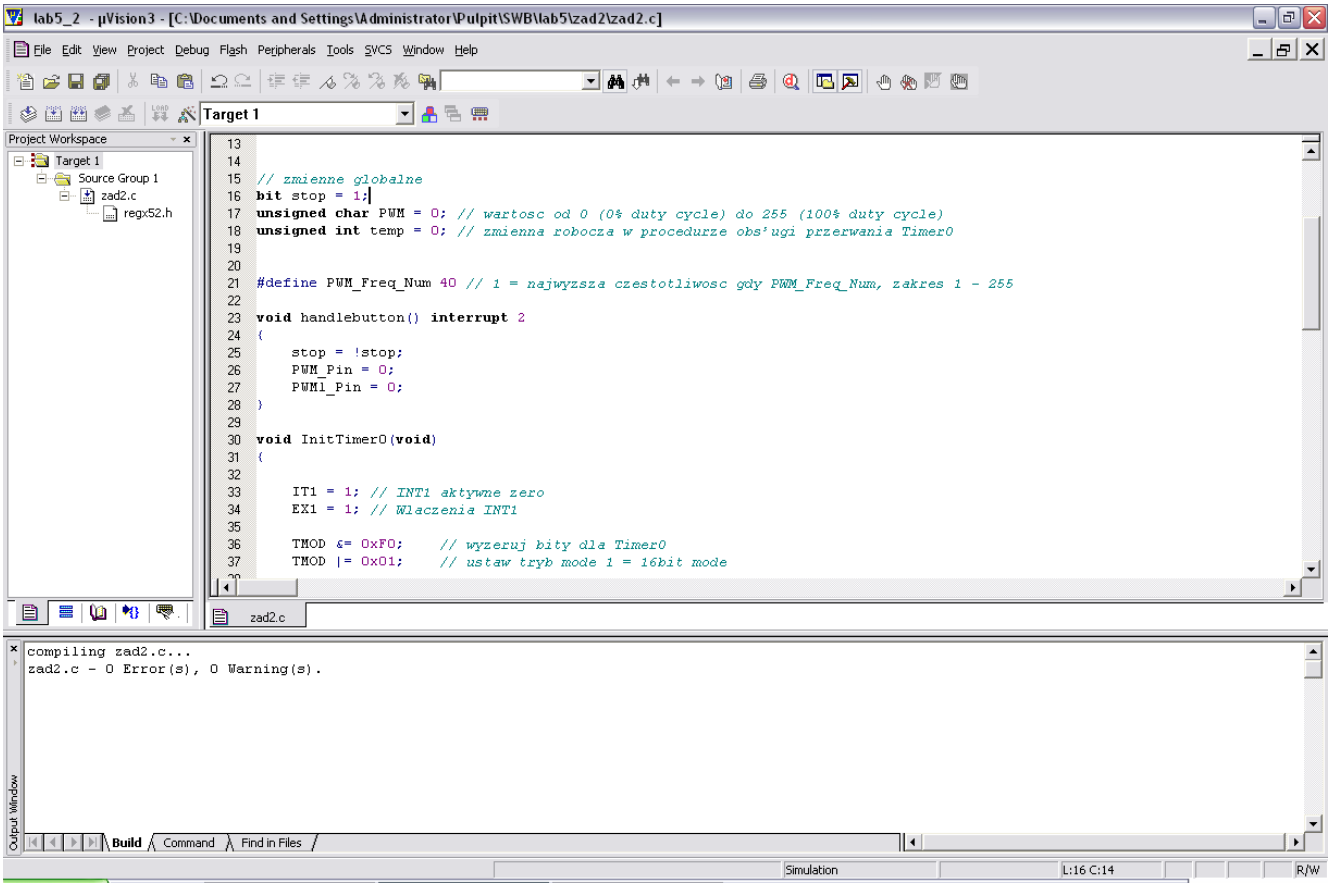
```
#include <REGX52.H>

sbit IN1_Pin = P2^1;
sbit IN2_Pin = P2^2;
sbit PWM_Pin = P2^3;
sbit PWMl_Pin = P2^4;
sbit IN3_Pin = P2^5;
sbit IN4_Pin = P2^6;



// zmienne globalne
bit stop = 1;
unsigned char PWM = 0; // wartosc od 0 (0% duty cycle) do 255 (100% duty cycle)
unsigned int temp = 0; // zmienna robocza w procedurze obsługi przerwania Timer0


#define PWM_Freq_Num 40 // 1 = najwyzsza czestotliwosc gdy PWM_Freq_Num, zakres 1 -
255

void handlebutton() interrupt 2
{
```

```
    stop = !stop;
    PWM_Pin = 0;
    PWMl_Pin = 0;
}

void InitTimer0(void)
{

    IT1 = 1; // INT1 aktywne zero
    EX1 = 1; // Wlaczenia INT1

    TMOD &= 0xF0;    // wyzeruj bity dla Timer0
    TMOD |= 0x01;    // ustaw tryb mode 1 = 16bit mode

    TH0 = 0x00;      // Pierwsze
    TL0 = 0x00;      // ustawienie

    ET0 = 1;         // Enable Timer0 interrupts
    EA  = 1;         // Enable All

    TR0 = 1;         // Start Timer 0
}

int main(void)
{
    InitTimer0(); // Init Timer0 dla rozpoczecia generacji przerwan
    IN1_Pin = 0;
    IN2_Pin = 1;
    PWM_Pin = 0;

    IN3_Pin = 0;
    IN4_Pin = 1;
    PWMl_Pin = 0;

    PWM = 69;
    while(1) {;}
}

void Timer0_ISR (void) interrupt 1
{
    TR0 = 0; // Stop Timer 0
    if(PWM_Pin) // if PWM_Pin =1 wyzeruj sygnal PWM i zaladuj licznik
    {
        temp = (255 - PWM) * PWM_Freq_Num;
        TH0 = 0xFF - (temp >> 8) & 0xFF;
        TL0 = 0xFF - temp & 0xFF;
    }
    else // if PWM_Pin =0 ustaw pin na 1 i zaladuj licznik
    {
        temp = PWM * PWM_Freq_Num;
        TH0 = 0xFF - (temp >> 8) & 0xFF;
        TL0 = 0xFF - temp & 0xFF;
    }
```

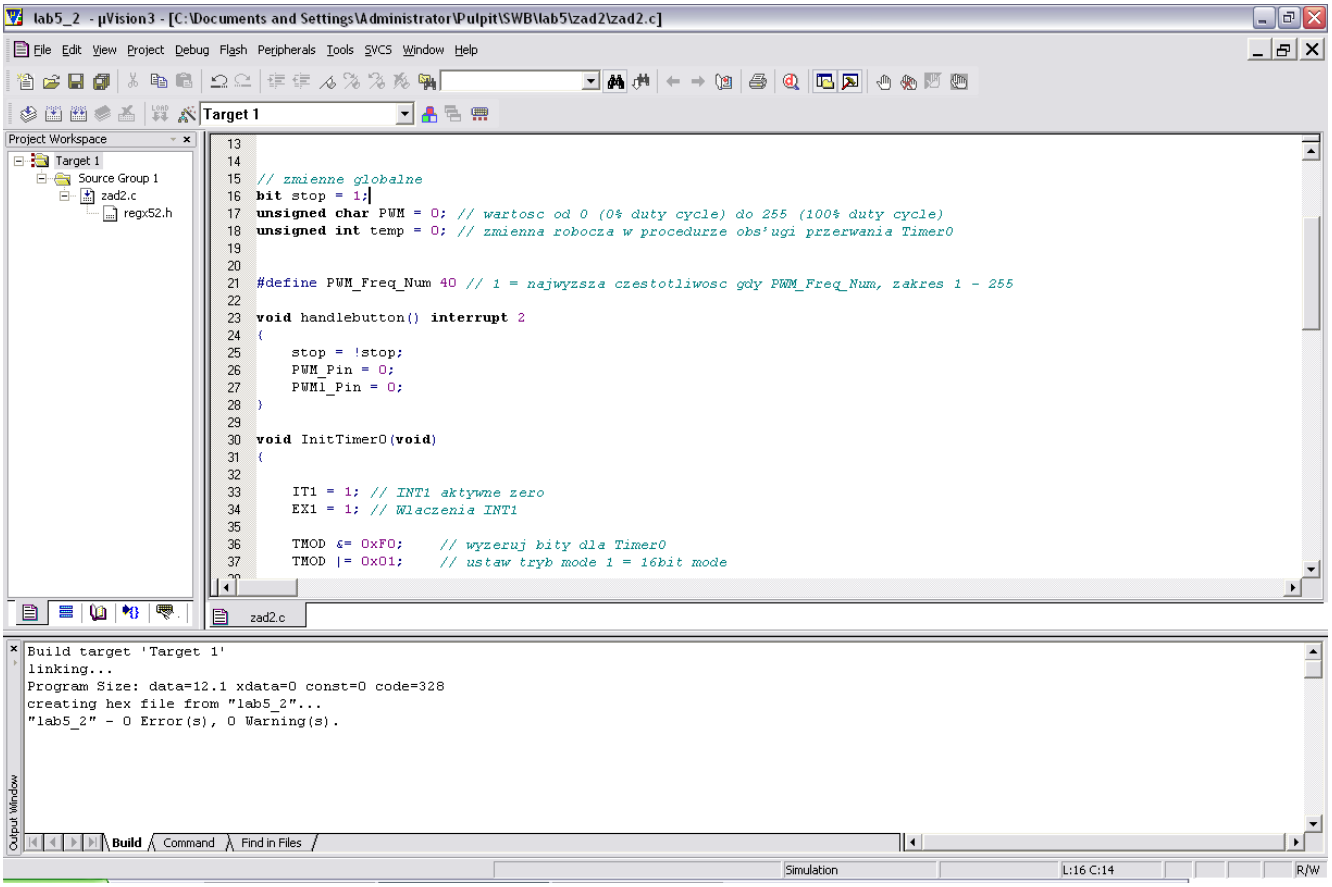
```
    if(!stop)
    {
        PWM_Pin = !PWM_Pin;
        PWMl_Pin = !PWMl_Pin;
    }

    TF0 = 0; // wyczysc flage
    TR0 = 1; // Start Timer 0
}
```

Sprawdzenie poprawności

Kompilowanie

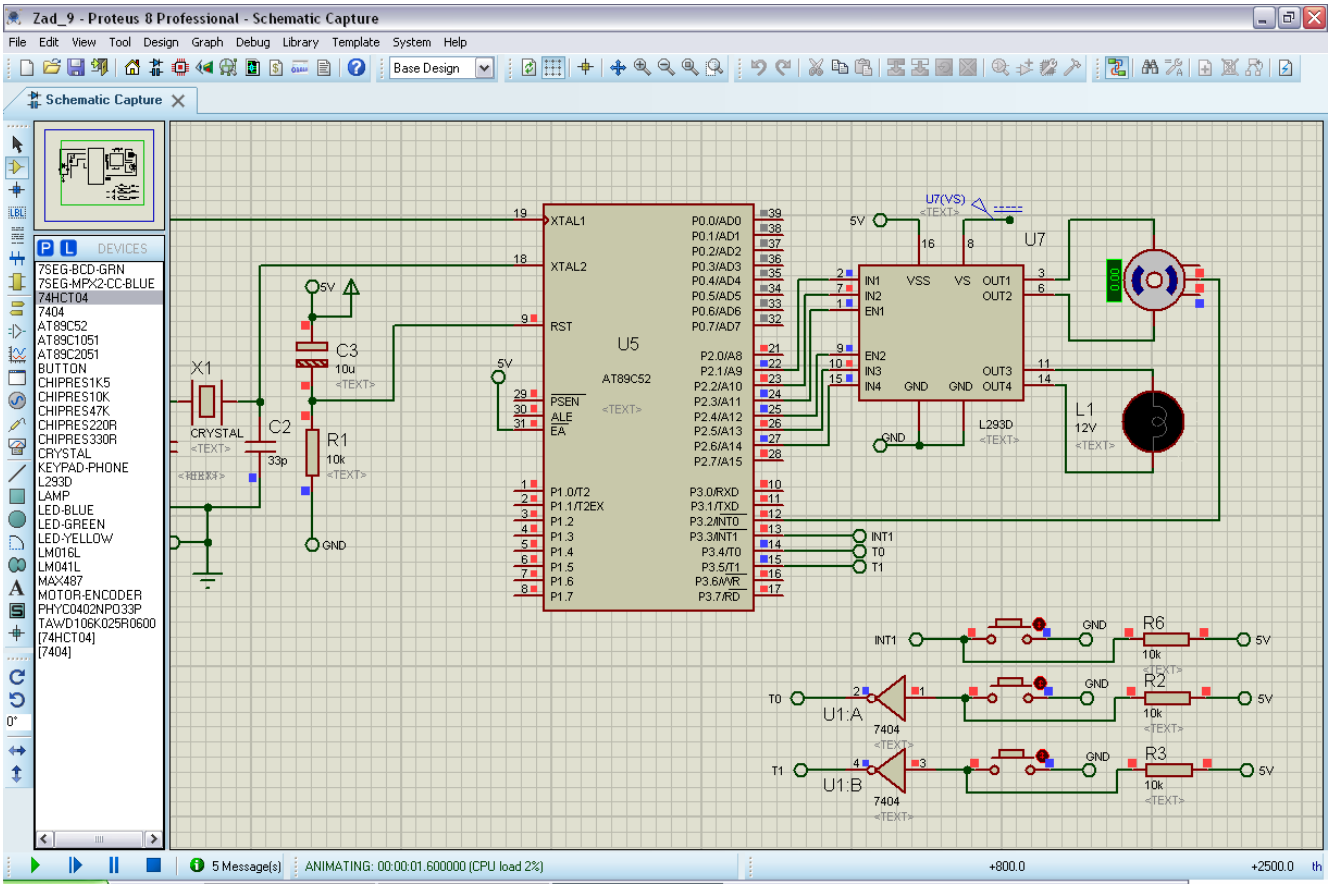


Linkowanie

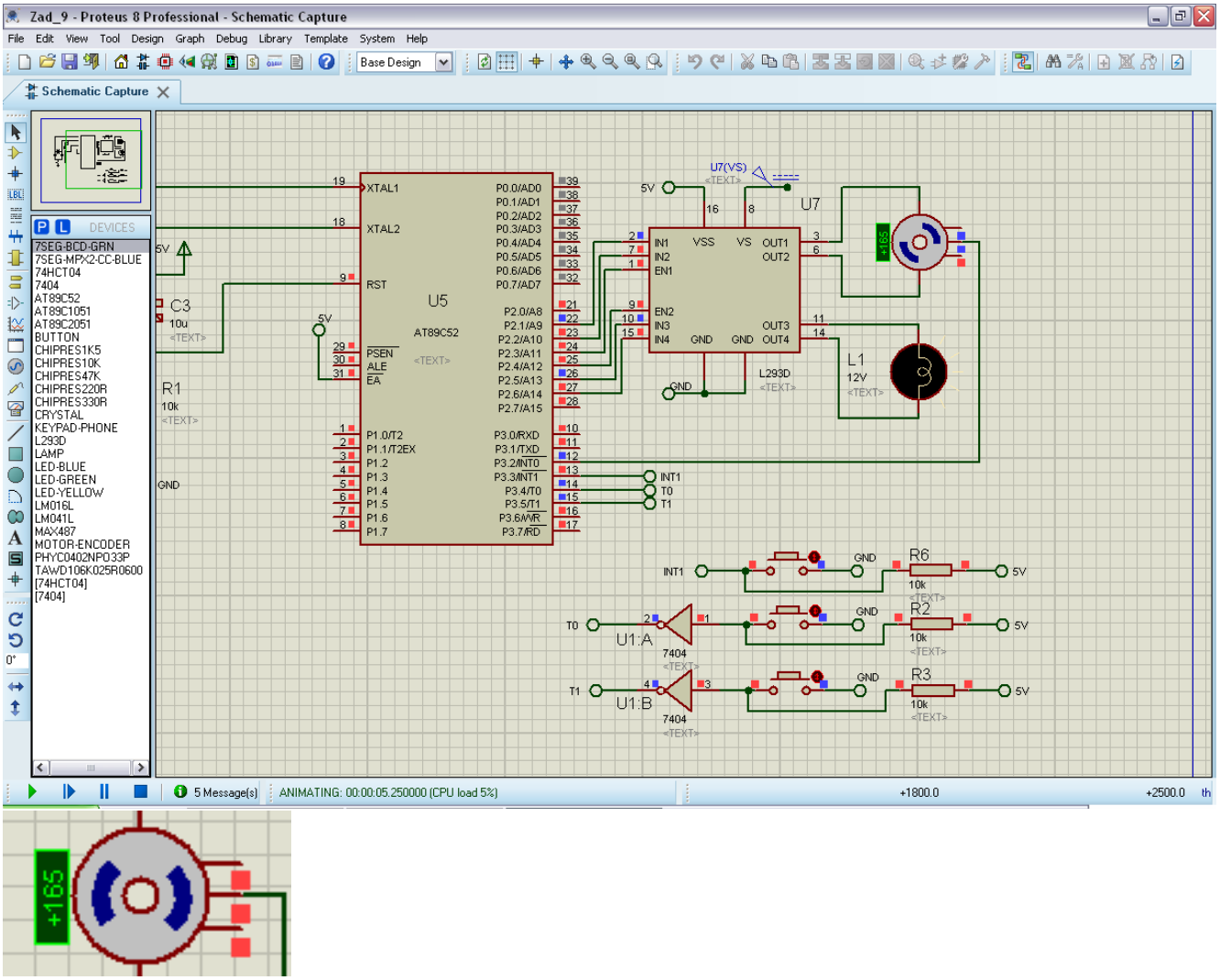


Prezentacja realizacji zadania przez program

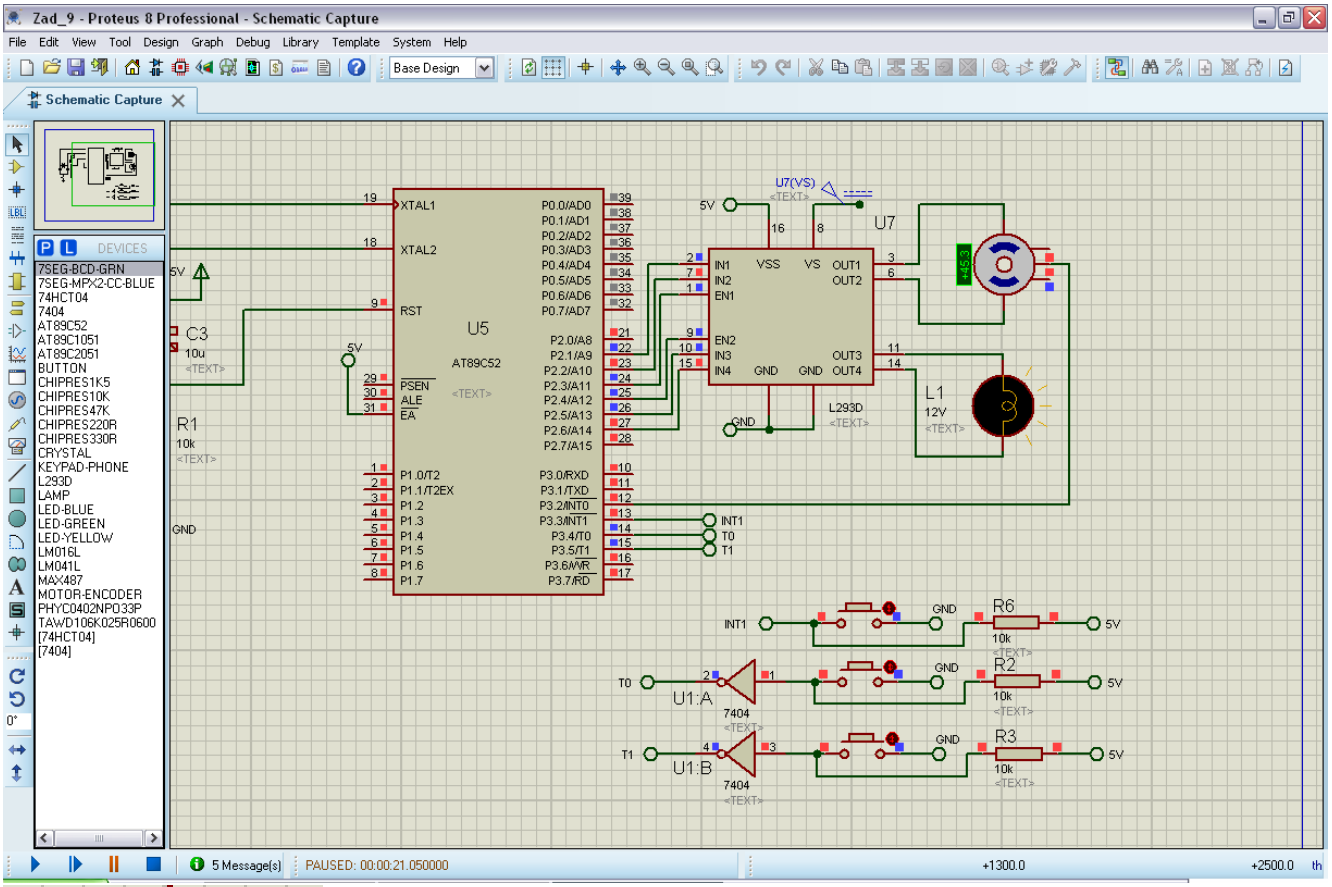
Stan początkowy, brak obrotów:

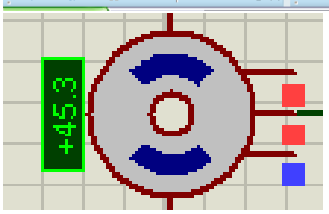


Włączenie programu dla 100% PWM (255). Stabilizuje się na wartości 165



Włączenie programu dla 27% PWM (69) . Oscyluje wokół wartości 45,5.





Sprawdzenie wyniku:
45.3 / 165 = 0,2729 ≈ 0,27 = 27%
45.5 / 165 = 0,2741 ≈ 0,27 = 27%

Zatem widać, że zależność między maksymalną prędkością a prędkością dla zadanego PWM jest zgodna z oczekiwaniami.

D. Zadanie na ocenę bardzo dobrą

Opis mojego rozwiązania

W układzie będą działały 3 mechanizmy:

1. Timer0 – licznik 16-bitowy dla wejścia T0 (TMOD = 0x55). Zliczać będzie wciśnięcia dla T0.
2. Timer1 – licznik 16-bitowy dla wejścia T1 (TMOD = 0x55). Zliczać będzie wciśnięcia dla T1.
3. Timer2 – timer 16-bitowy - generator sygnału (T2CON = 0x04) (to samo, co timer0 w zad 4).

Interrupt 5 – obsługa generatora PWM - timer2

Ponieważ zmienia się generator, należy przekształcić funkcję Timer0_ISR() na funkcję Timer2_ISR(). Zachowuje ona swój cały algorytm, zmianie ulegają jedynie flagi, zmieniają się z 0 na 2).

Interrupt 1 oraz **Interrupt 3** – obsługa przycisków do włączania i wyłączania silnika

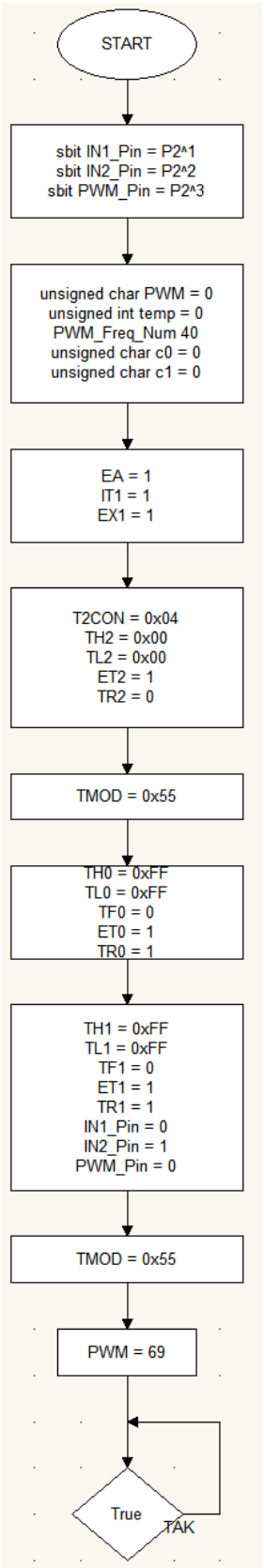
Liczniki będą posiadały swoje zmienne zliczające wciśnięcia. Każde wciśnięcie będzie działało zgodnie z poniższą listą kroków:

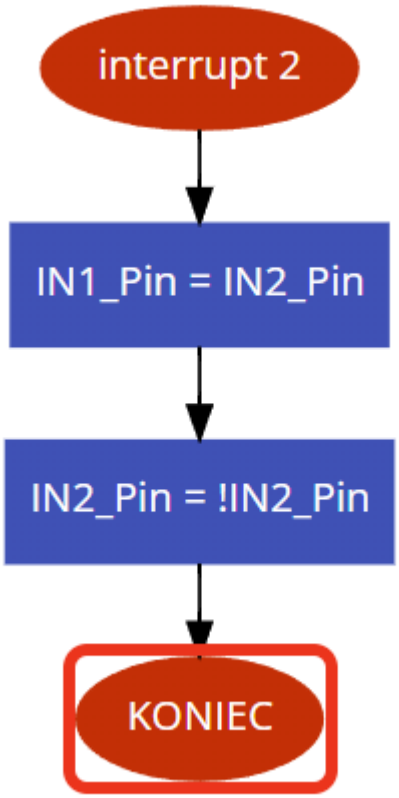
1. Zwiększ wartość zmiennej danego licznika.
2. Wyzeruj wartość zmiennej drugiego licznika (na wypadek, gdyby poprzednio został wciśnięty).
3. Jeśli naliczono odpowiednią ilość wciśnięć, włącz (T0 dla 2) lub wyłącz (T1 dla 3) silnik.

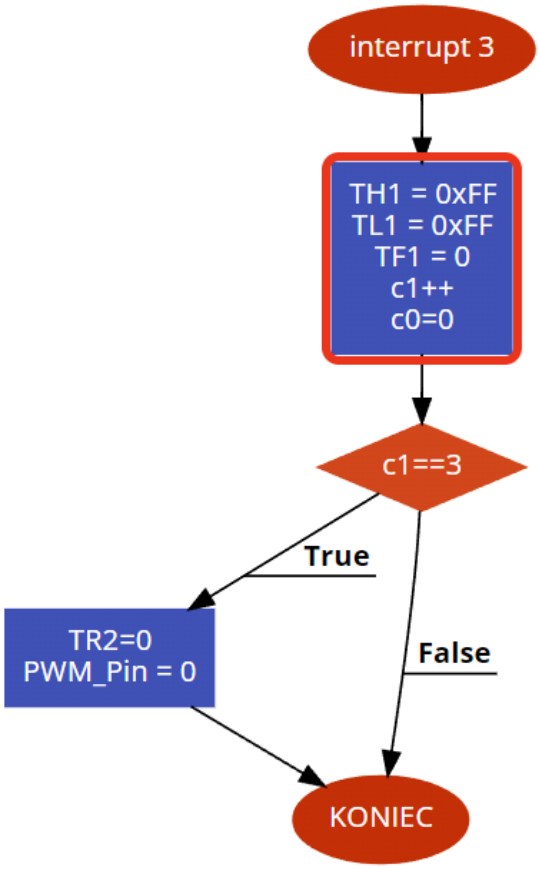
Interrupt 2 - obsługa zmiany kierunku obrotów

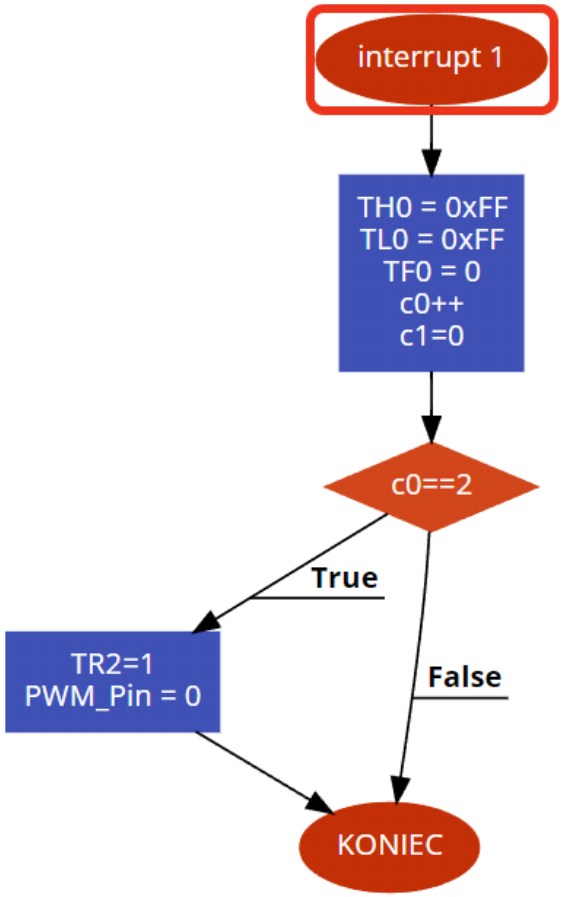
Jeśli zostanie wciśnięty przycisk dla int0, to odwraca on kierunek poruszania się silnika. Działa on nawet, jeśli silnik stoi w miejscu (po włączeniu silnik zacznie się kręcić w przeciwnym kierunku do ostatniego). Odwrócenie kierunku odbywa się na podstawie mechanizmu z zadania 4, czyli jeśli chcę wartość dodatnią, to IN=0, a IN=1, w przeciwnym wypadku, IN0=1, IN1=0.

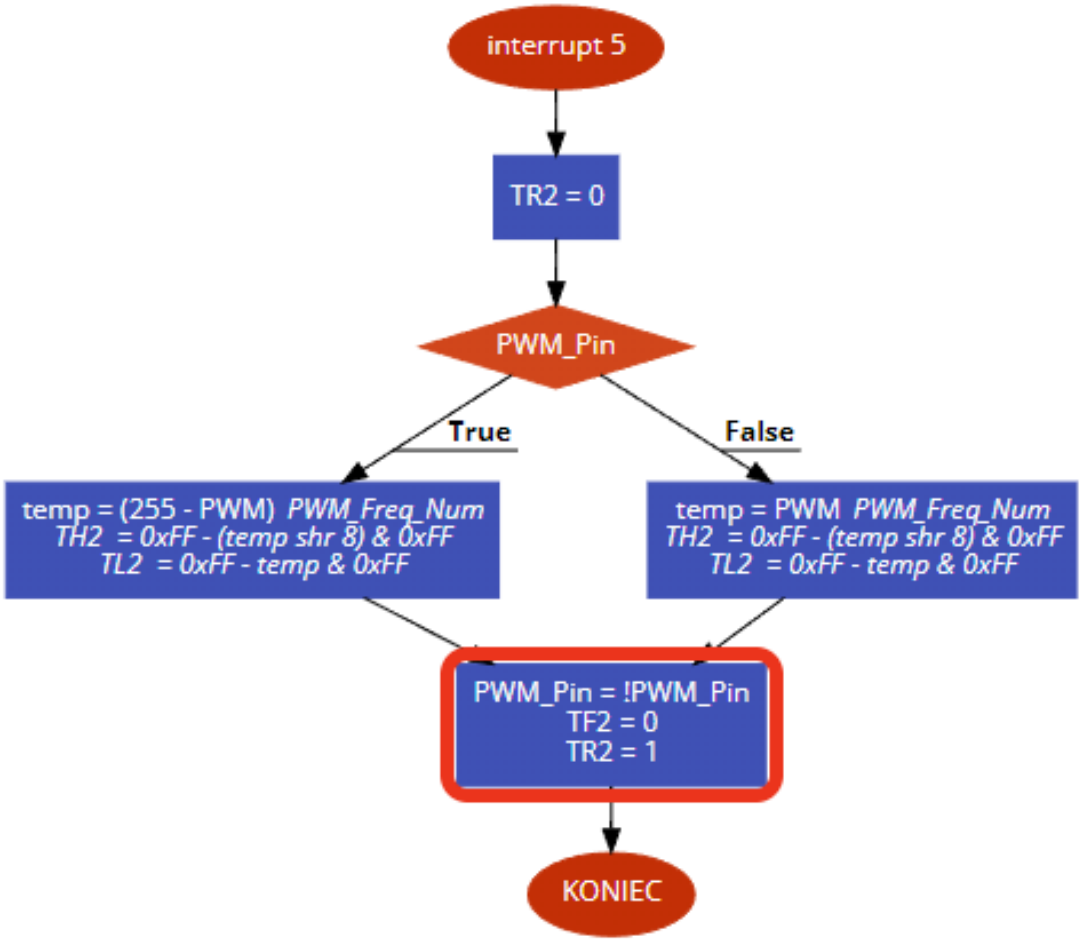
Schemat blokowy rozwiązania






interrupt 3
TH1 = 0xFF
TL1 = 0xFF
TF1 = 0
c1++
c0=0
c1==3
True
False
TR2=0
PWM_Pin = 0
KONIEC


interrupt 1
TH0 = 0xFF
TL0 = 0xFF
TF0 = 0
c0++
c1=0
c0==2
True
False
TR2=1
PWM_Pin = 0
KONIEC


interrupt 5
TR2 = 0
PWM_Pin
True
False
temp = (255 - PWM) PWM_Freq_Num
TH2 = 0xFF - (temp shr 8) & 0xFF
TL2 = 0xFF - temp & 0xFF
temp = PWM PWM_Freq_Num
TH2 = 0xFF - (temp shr 8) & 0xFF
TL2 = 0xFF - temp & 0xFF
PWM_Pin = !PWM_Pin
TF2 = 0
TR2 = 1
KONIEC

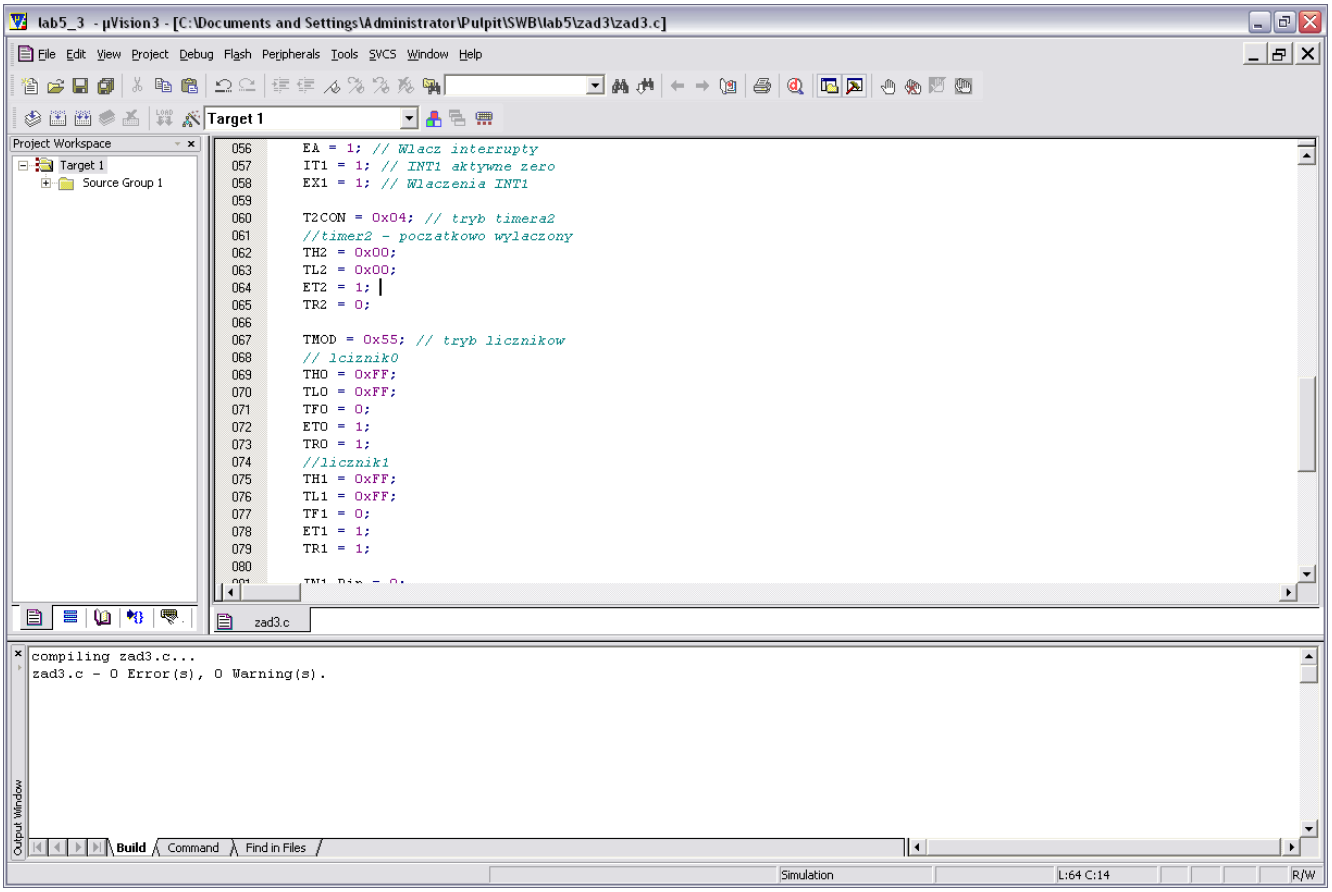
Listing programu

```
//PWM_Przyklad
// zmiana wspolczynnika wypelnienia
// zmiana czestotliwosci impulsow PWM (Pulse Width Modulation)

#include <REGX52.H>

sbit IN1_Pin = P2^1;
sbit IN2_Pin = P2^2;
sbit PWM_Pin = P2^3;

// zmienne globalne
unsigned char PWM = 0; // wartosc od 0 (0% duty cycle) do 255 (100% duty cycle)
unsigned int temp = 0; // zmienna robocza w procedurze obsługi przerwania Timer0

#define PWM_Freq_Num 40 // 1 = najwyzsza czestotliwosc gdy PWM_Freq_Num, zakres 1 -
255

unsigned char c0 = 0; // Counter0 - dla Timer0
unsigned char c1 = 0; // Counter1 - dla Timer1

void handlebuttonI0() interrupt 2
{
    IN1_Pin = IN2_Pin;
    IN2_Pin = !IN2_Pin;
}

void handlebuttonT0() interrupt 1
{
    TH0 = 0xFF;
    TL0 = 0xFF;
    TF0 = 0;

    c0++;
    c1=0;
    if(c0==2){
        TR2=1; // Wlacz silnik
        PWM_Pin = 0;
    }
}

void handlebuttonT1() interrupt 3
{
    TH1 = 0xFF;
    TL1 = 0xFF;
    TF1 = 0;

    c1++;
    c0=0;
    if(c1==3){
        TR2=0; // Wylacz silnik
        PWM_Pin = 0;
    }
```

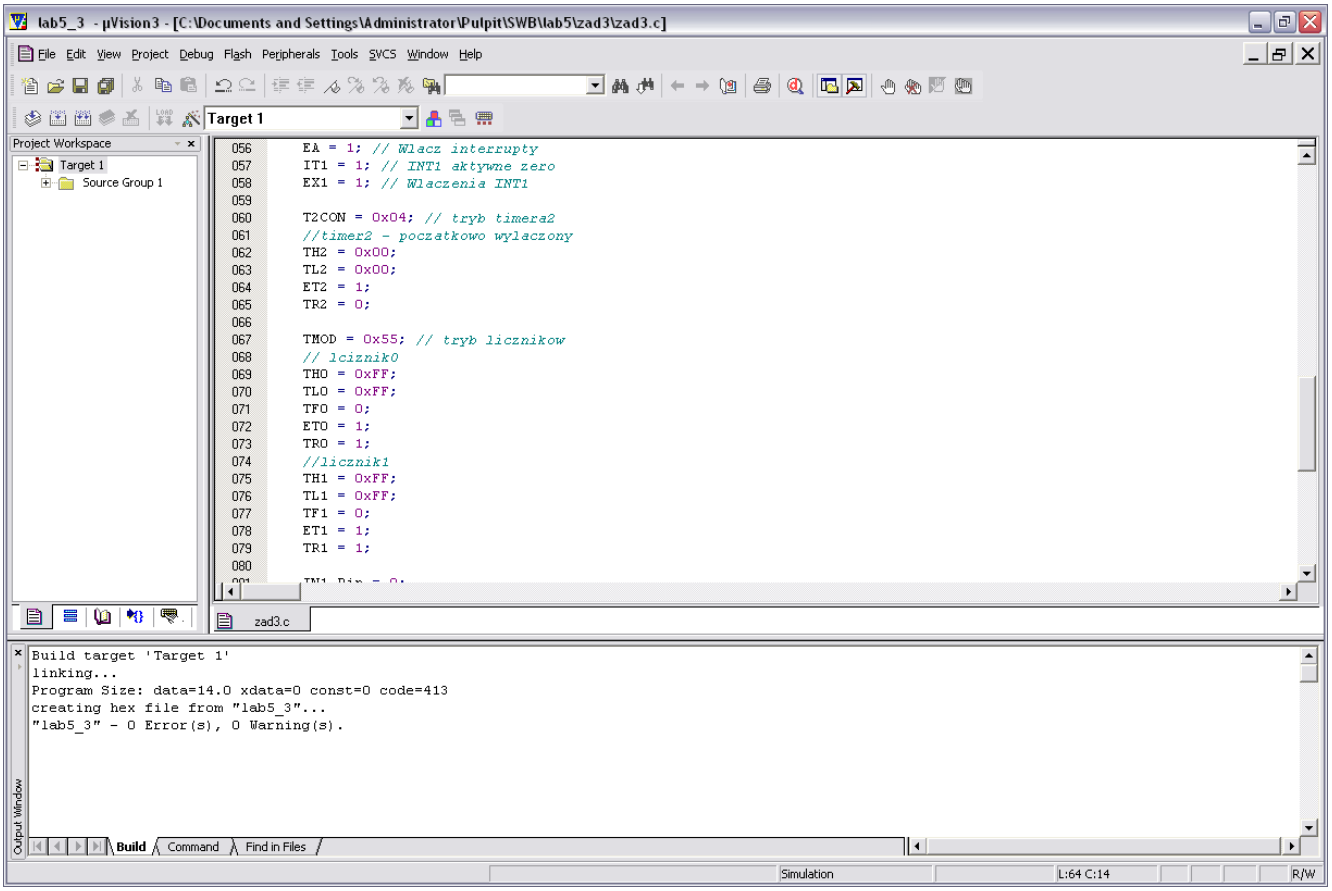
```
}

int main(void)
{
    EA = 1; // Wlacz interrupty
    IT1 = 1; // INT1 aktywne zero
    EX1 = 1; // Wlaczenia INT1

    T2CON = 0x04; // tryb timera2
    //timer2
    TH2 = 0x00;
    TL2 = 0x00;
    ET2 = 1;
    TR2 = 0;

    TMOD = 0x55; // tryb licznikow
    // lciznik0
    TH0 = 0xFF;
    TL0 = 0xFF;
    TF0 = 0;
    ET0 = 1;
    TR0 = 1;
    //licznik1
    TH1 = 0xFF;
    TL1 = 0xFF;
    TF1 = 0;
    ET1 = 1;
    TR1 = 1;

    IN1_Pin = 0;
    IN2_Pin = 1;
    PWM_Pin = 0;

    PWM = 69;
    while(1) {;}
}

void Timer2_ISR(void) interrupt 5
{
    TR2 = 0; // stop Timer 2
    if(PWM_Pin) {
        temp = (255 - PWM) * PWM_Freq_Num;
        TH2  = 0xFF - (temp >> 8) & 0xFF;
        TL2  = 0xFF - temp & 0xFF;
    } else {
        temp = PWM * PWM_Freq_Num;
        TH2  = 0xFF - (temp >> 8) & 0xFF;
        TL2  = 0xFF - temp & 0xFF;
    }
    PWM_Pin = !PWM_Pin;
    TF2 = 0; // wyczysc flage
    TR2 = 1; // start Timer 2
}
```

Sprawdzenie poprawności

Kompilowanie

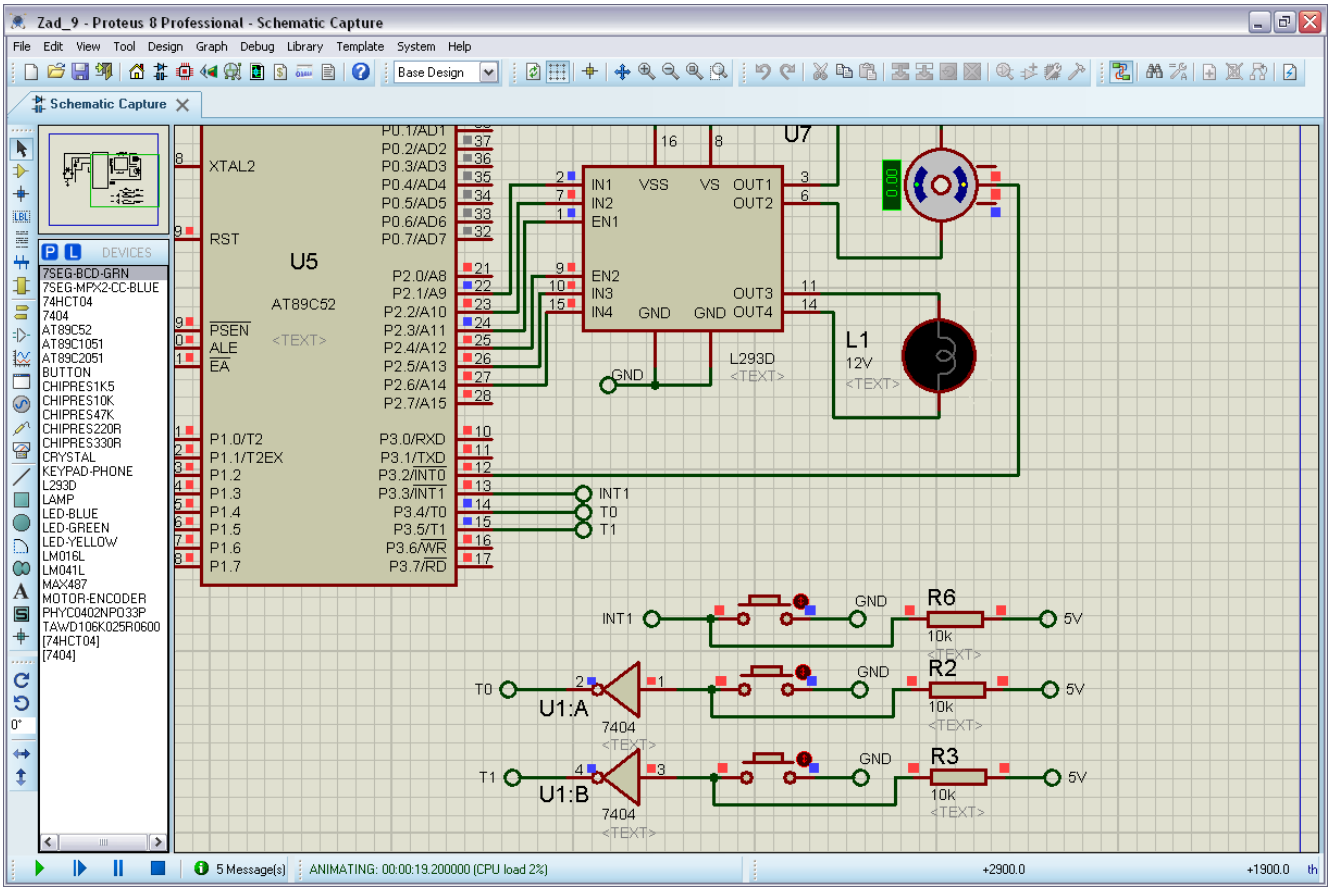


Linkowanie

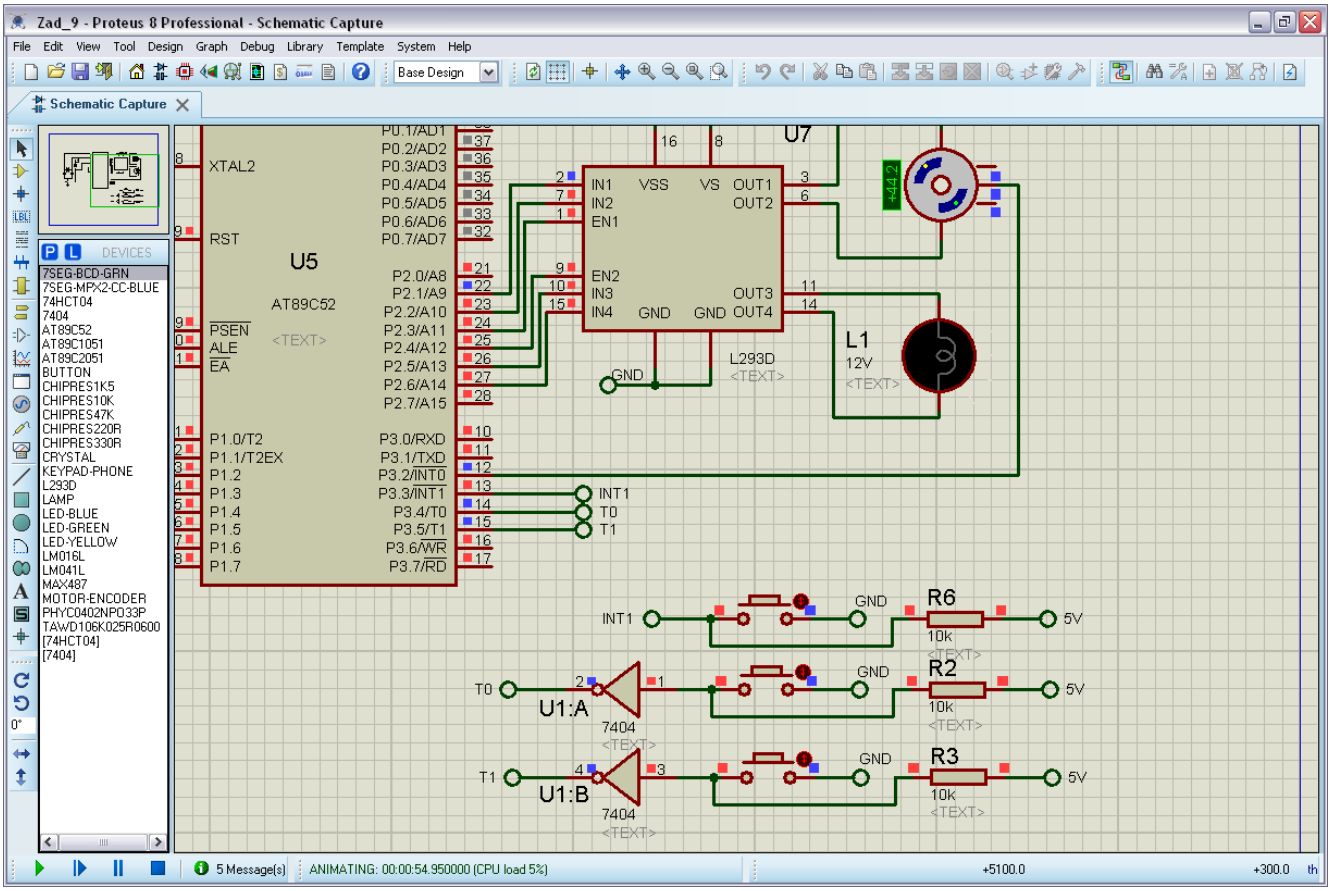


Prezentacja realizacji zadania przez program

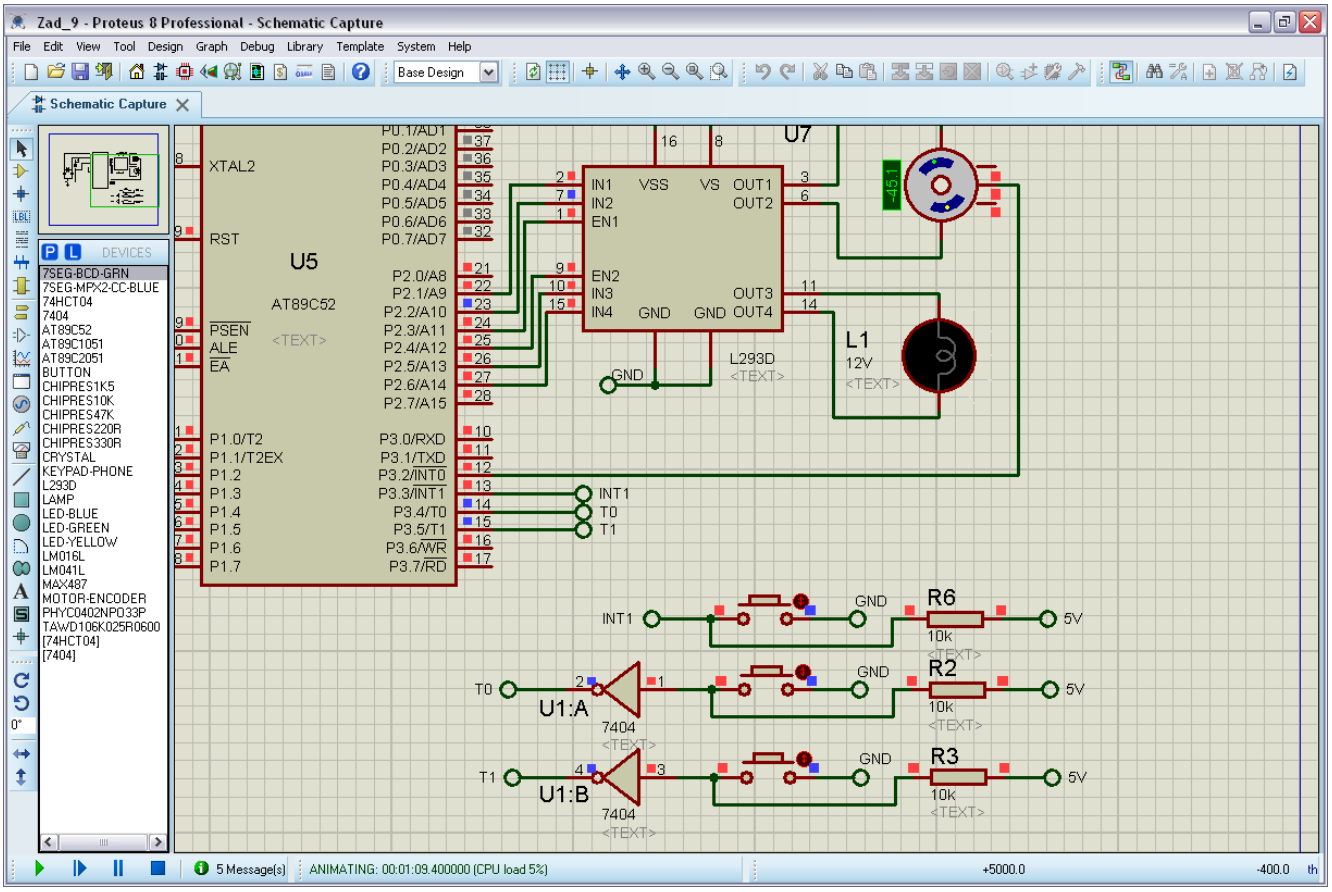
Stan początkowy, brak ruchu



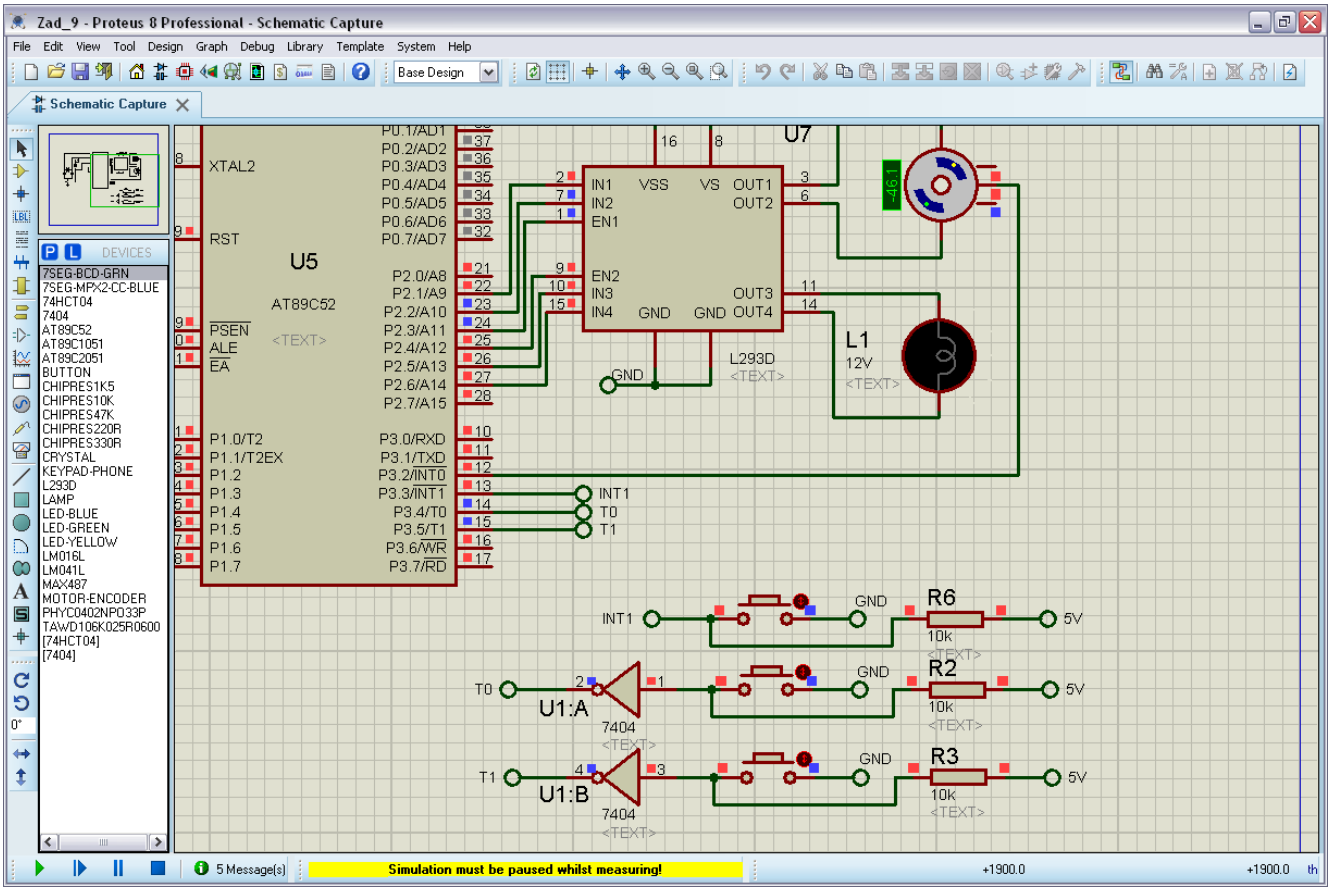
2 wciśnięcia dla T0 – wlaczenie silnika (początkowe - dodatnie)



Wciśnięcie int1 – zmiana kierunku obrotów



Wciskanie na przemian T0 i T1 – brak zmian


Zad_9 - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
DEVICES
7SEG-BCD-GRN
7SEG-MPX2-CC-BLUE
74HCT04
7404
AT89C52
AT89C1051
AT89C2051
BUTTON
CHIPRES1K5
CHIPRES10K
CHIPRES47K
CHIPRES220R
CHIPRES330R
CRYSTAL
KEYPAD-PHONE
L293D
LAMP
LED-BLUE
LED-GREEN
LED-YELLOW
LM016L
LM041L
MAX487
MOTOR-ENCODER
PHYC0402NP033P
TAWD106K025R0600
[74HCT04]
[7404]
U5
AT89C52
XTAL2
RST
PSEN
ALE
EA
P1.0/T2
P1.1/T2EX
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
INT1
T0
T1
U7
L293D
IN1
IN2
EN1
EN2
IN3
IN4
VSS
VS
OUT1
OUT2
OUT3
OUT4
GND
L1
12V
R6
R2
R3
10k
5V
U1:A
U1:B
7404
5 Message(s)
Simulation must be paused whilst measuring!
+1900.0