

Wojskowa Akademia Techniczna im. Jarosława Dąbrowskiego

Laboratorium z przedmiotu:

Systemy wbudowane

Sprawozdanie z ćwiczenia laboratoryjnego nr 1:
Wytwarzanie i testowanie oprogramowania.

Prowadzący:

mgr inż. Artur Miktus

Wykonał: Radosław Relidzyński

Grupa: WCY20IY4S1

Data laboratoriów: 27.04.2022 r.

Deklarowana ocena: 3, 4, 5

Spis treści

A.	Treść zadania	2
	<i>Zadanie na Lab 1 SWB</i>	2
B.	Zadanie na ocenę dostateczną	6
	Opis mojego rozwiązania.....	6
	Schemat blokowy rozwiązania	6
	Listing programu.....	6
	Sprawdzenie poprawności.....	7
	Prezentacja realizacji zadania przez program	9
C.	Zadanie na ocenę dobrą	14
	Opis mojego rozwiązania.....	14
	Schemat blokowy rozwiązania	15
	Listing programu.....	15
	Sprawdzenie poprawności.....	16

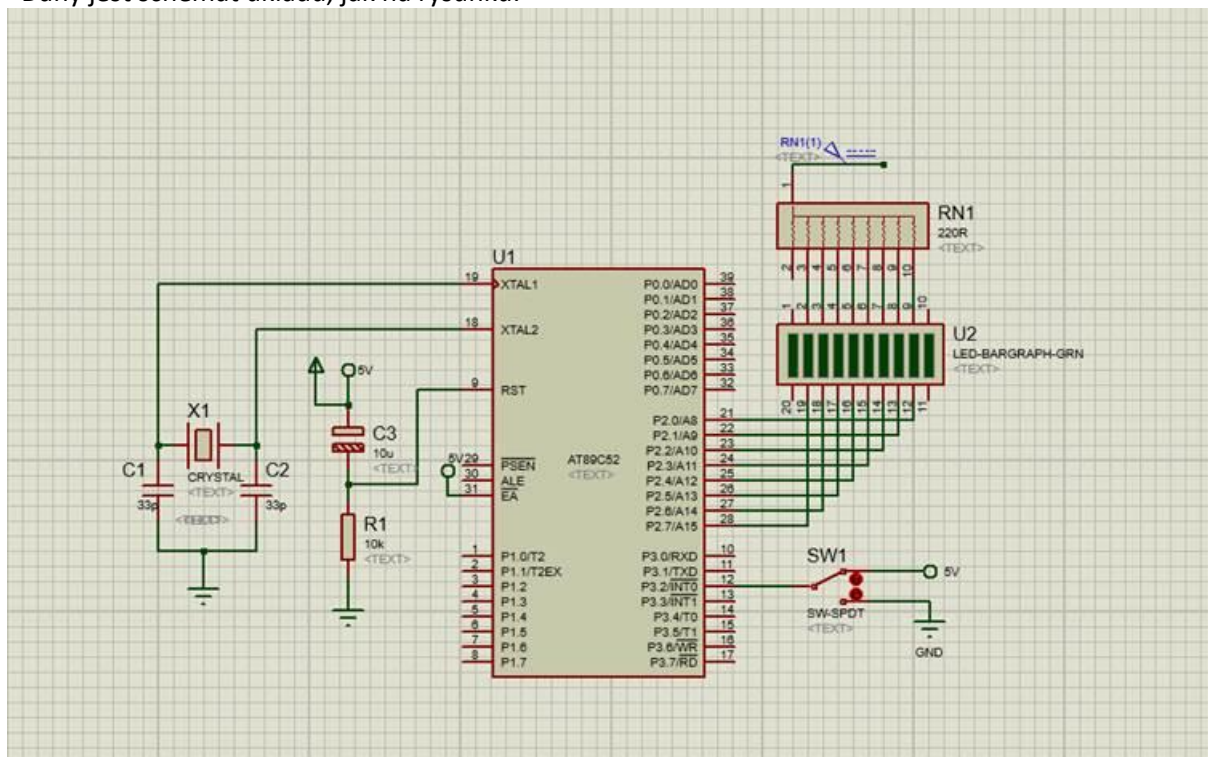
Prezentacja realizacji zadania przez program	17
D. Zadanie na ocenę bardzo dobrą	22
Opis mojego rozwiązania.....	22
Schemat blokowy rozwiązania	23
Listing programu.....	23
Sprawdzenie poprawności.....	24
Prezentacja realizacji zadania przez program	26

A. Treść zadania

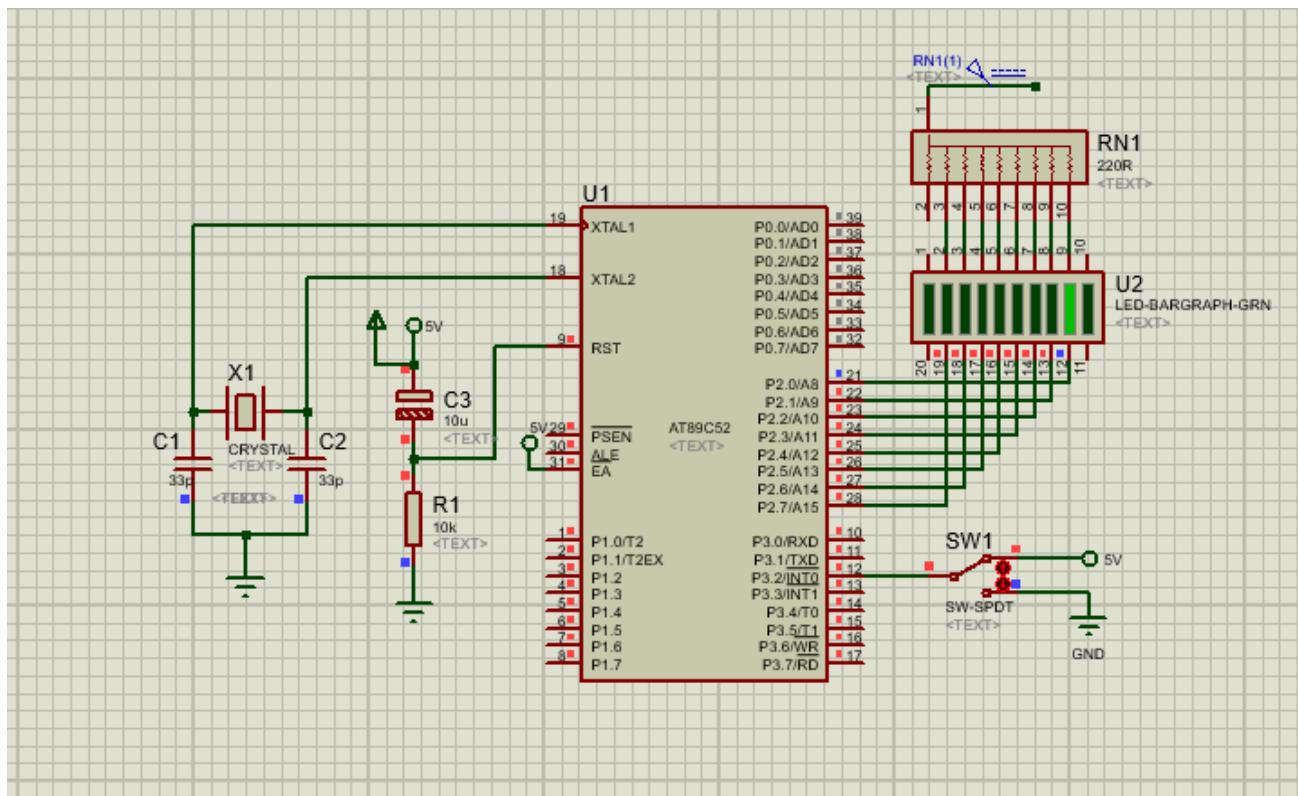
Zadanie na Lab 1 SWB

Link [do opisu importowania i pliku projektu Proteusa](#) z prezentowanym układem.

1. Dany jest schemat układu, jak na rysunku:



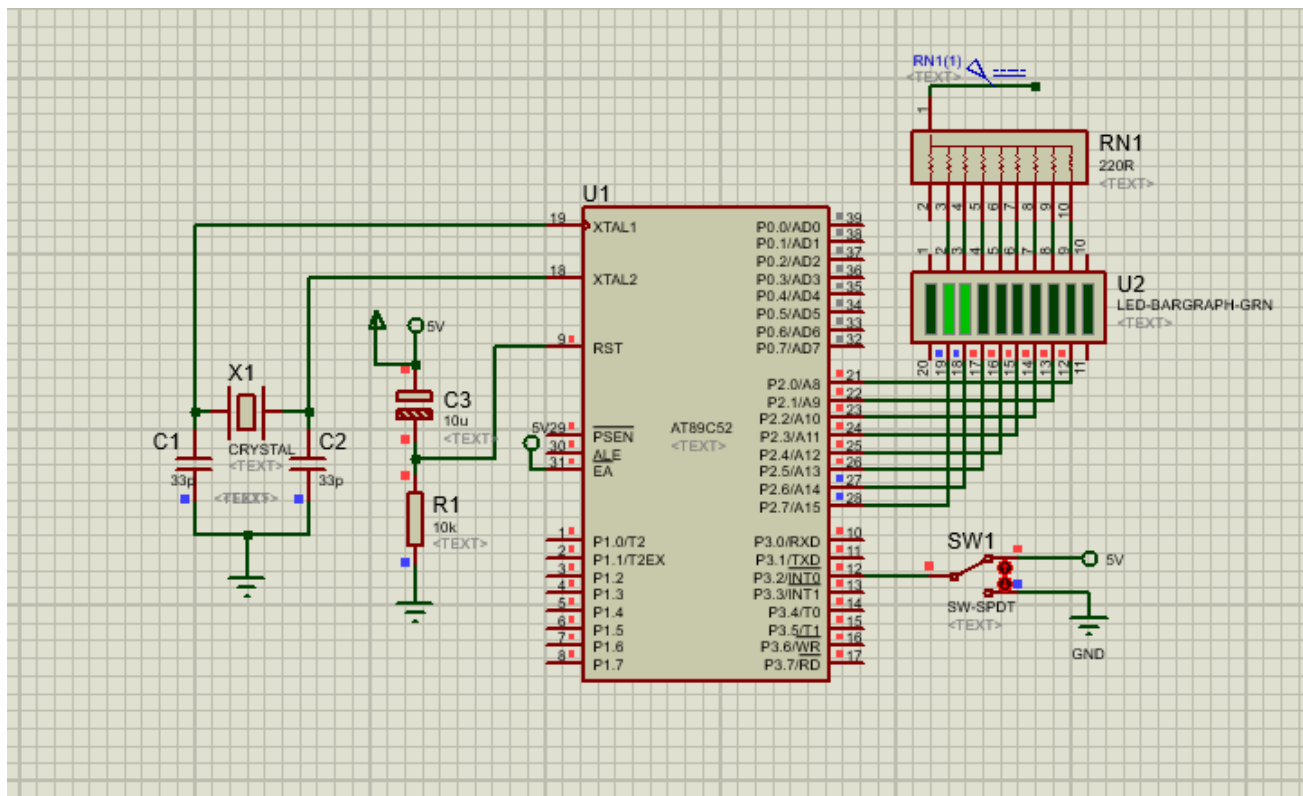
2. Zadanie na ocenę **dostateczną**: napisz program **prog1.c**, powodujący **cykliczne wędrujące zapalanie się pojedynczej diody** w grupie U2 od strony prawej (wyjście P2.0) do lewej (P2.7) z opóźnieniem, wygodnym do obserwacji (należy dobrać właściwy czas opóźnienia Delay ()). Program nie reaguje na zmiany stanu P3.2.



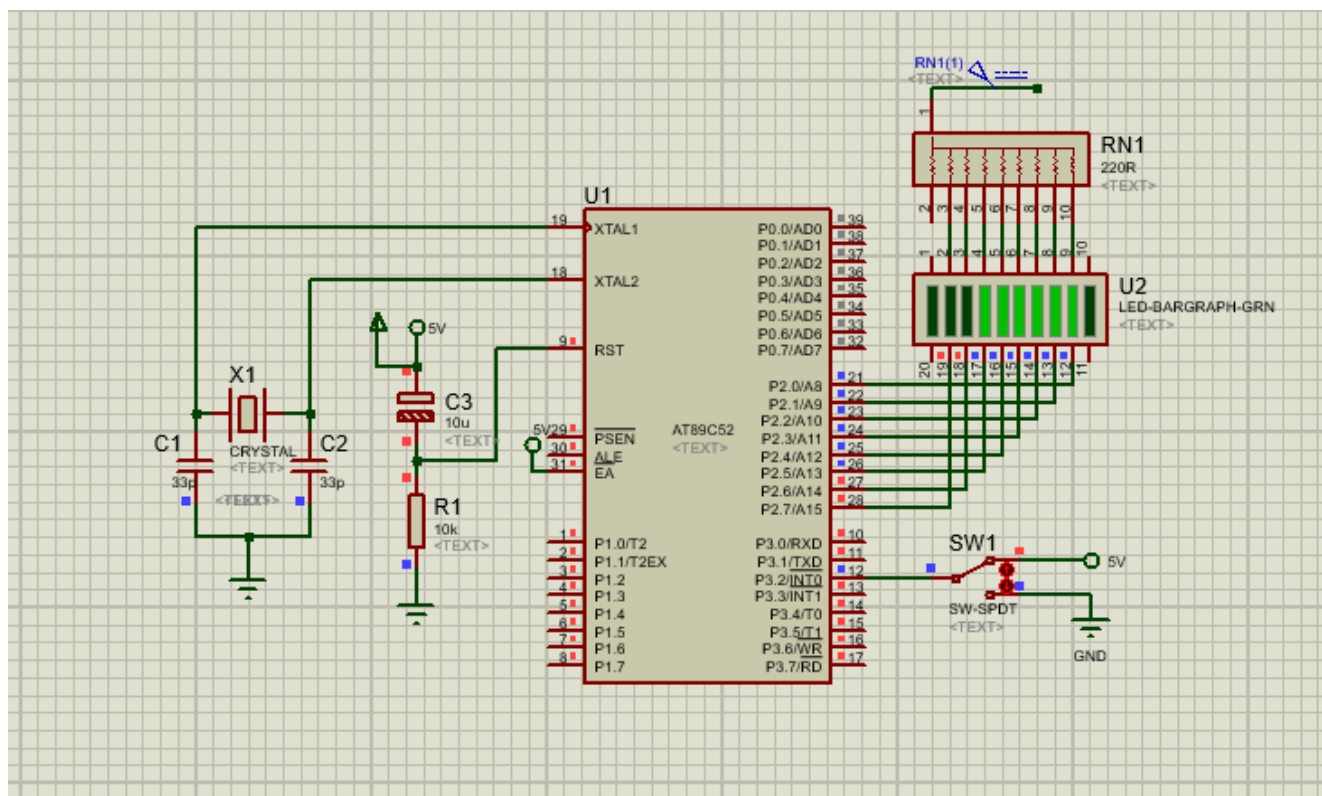
3. Zadanie na ocenę **dobrą**: to co na ocenę dostateczną i ponadto napisz program **prog2.c**, powodujący cykliczne **wędrujące zapalanie się jednocześnie po 2 diody** w grupie U2 od strony lewej (wyjście P2.7) do prawej (wyjście P2.0), przy czym co druga zmiana diod ma się odbywać z podwójnym opóźnieniem (np. 1s, 2s, 1s, 2s itd.), a diody zapalają się z przeskokiem o 1 pozycję, czyli w 1 cyklu wykona się **7 kroków dla zapalania diod sąsiednich** i **6 kroków dla zapalania diod przedzielonych jedną zgaszoną**. Program nie reaguje na zmiany stanu P3.2.

Studenci o numerach na liście grupy nieparzystych zapalają i przesuwają dwie sąsiednie diody [np. tak, jak na rysunku poniżej świecą kolejno diody pod numerami (2 i 3), (3 i 4) itd.] - **7 kroków**.

Studenci o numerach na liście grupy parzystych zapalają i przesuwają dwie diody, przedzielone jedną zgaszoną [np. na rysunku poniżej świecą kolejno diody pod numerami (2 i 4), (3 i 5) itd.] - **6 kroków**.



4. Na ocenę **bardzo dobrą**: to co na ocenę dobrą i ponadto napisz program **prog3.c**, w którym cyklicznie **diody zapalają się pojedynczo, a już zapalone nie gasną** (coraz szerszy świecący pasek), w kolejności od początku prawej do lewej (Led 9 --> Led 2) **gdy P3.2 = 1** i od początku lewej do prawej (Led 2 --> Led 9) **gdy P3.2 = 0**. Po dojściu do końca paska LED wszystkie diody na chwilę (proszę wybrać takie opóźnienie, aby można było to wygodnie zaobserwować) gasną, a proces zapalania LED się powtarza. Sprawdzanie stanu przełącznika SW1, podłączonego do P3.2 ma się odbywać w każdej iteracji pętli tak, aby w trakcie symulacji można było zmieniać kierunek narastania świecącego paska przez zmianę stanu P3.2 - w takiej sytuacji dotychczas zapalone diody gasną, a pasek wydłuża się od początku drugiej strony.



Każdy program ma zostać skompilowany i zlinkowany w środowisku Keil uVision **do postaci .hex** (wykonywalnej przez mikrokontroler po załadowaniu) i przetestowany z wykorzystaniem pliku projektu układu, zamieszczonego w powyższym linku.

Po napisaniu i przetestowaniu programu należy wytworzyć plik sprawozdania (za pomocą edytora typu MS Office, LibreOffice czy OpenOffice), zawierający

a) stronę tytułową z nazwą przedmiotu, tematem zajęć "**Lab1: Wytwarzanie i testowanie oprogramowania.**", nazwiskiem, imieniem i numerem grupy wykonawcy, datą wykonania ćwiczenia, bez żadnych „ozdobników” graficznych typu logo WAT; Poniżej proszę zamieścić deklarację, na jakie oceny zostały wykonane zadania w postaci napisu np. "Wykonano zadania na ocenę dst i db".

a na kolejnych stronach

b) treść zadania na daną ocenę z mojej witryny;

c) **opis problemu na odpowiednią ocenę i krótki pomysł na jego rozwiązanie (sam program bez wyjaśnień w postaci słownego opisu, jakie dane program wykorzystuje, co robi i dlaczego zostały użyte akurat takie wartości liczbowe nie będzie akceptowany);**

d) narysować algorytm schemat blokowy, (np. za pomocą DiagramDesignera, <https://www.dobreprogramy.pl/Diagram-Designer,Program,Windows,12518.html>),

e) zamieścić listing (treść) programu i zrzuty ekranowe z kompilacji i z linkowania, pokazujące brak błędów i rozmiar kodu i danych

f) oraz 4 zrzuty ekranowe z każdego zadania z debugera Keil i 4 zrzuty ekranowe z każdego

zadania dla symulatora Proteus, pokazujące realizację przez badany program postawionego zadania.

Nazwa pliku ze sprawozdaniem w edytorze to Lab.**Nr_zajęc**.Nazwisko.Imię.Grupa.docx, na przykład

Lab.1.Nowak.Jerzy.WCY19IY8S1.docx

Uwaga, po zakończeniu tworzenia sprawozdania zapisujemy je (Plik / Zapisz jako ...) w postaci .pdf.

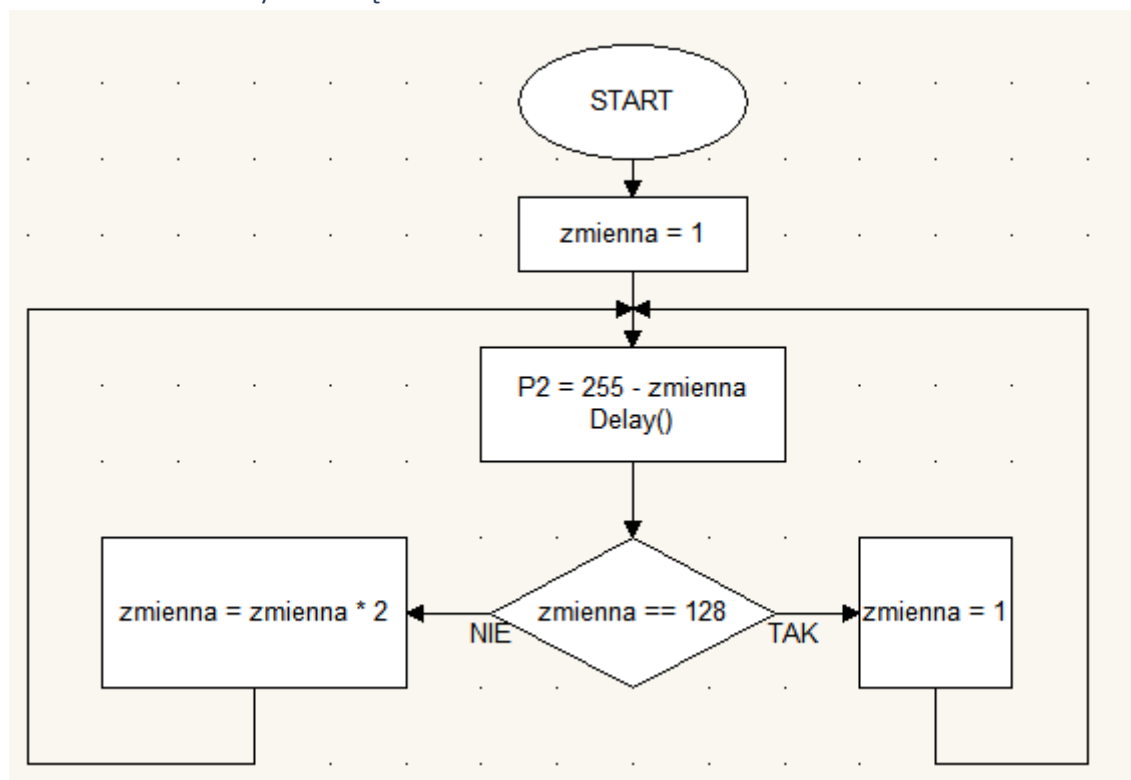
Nie przyjmuję sprawozdań w postaci edytowalnej typu .docx.

B. Zadanie na ocenę dostateczną

Opis mojego rozwiązania

W ramach realizacji tego zadania wykorzystałem zmienną, która będzie wskazywać wartość bitu wyjścia, na którym będzie zapalona dioda. Na wyjście trzeba przekazać wartość binarną 1111 1111 pomniejszoną o wartość bitu, na którym dioda ma się zapalić. Dziesiętnie będzie to 2^x , gdzie x to kolejna dioda od prawej licząc od 0 dla skrajnej. Wartość ta będzie przechowywana w zmiennej, cyklicznie zmieniającej wartość (cykl zamyka się zmieniając wartość ze 128 na 1). Dlatego na wyjście przekazuję wartość 255 pomniejszoną o wspomnianą wcześniej zmienną.

Schemat blokowy rozwiązania



Listing programu

```
#include <REGX52.H>

void Delay(void) // funkcja okreslajaca dlugosc przerwy miedzy kazda iteracja
{
    unsigned char j;
```

```

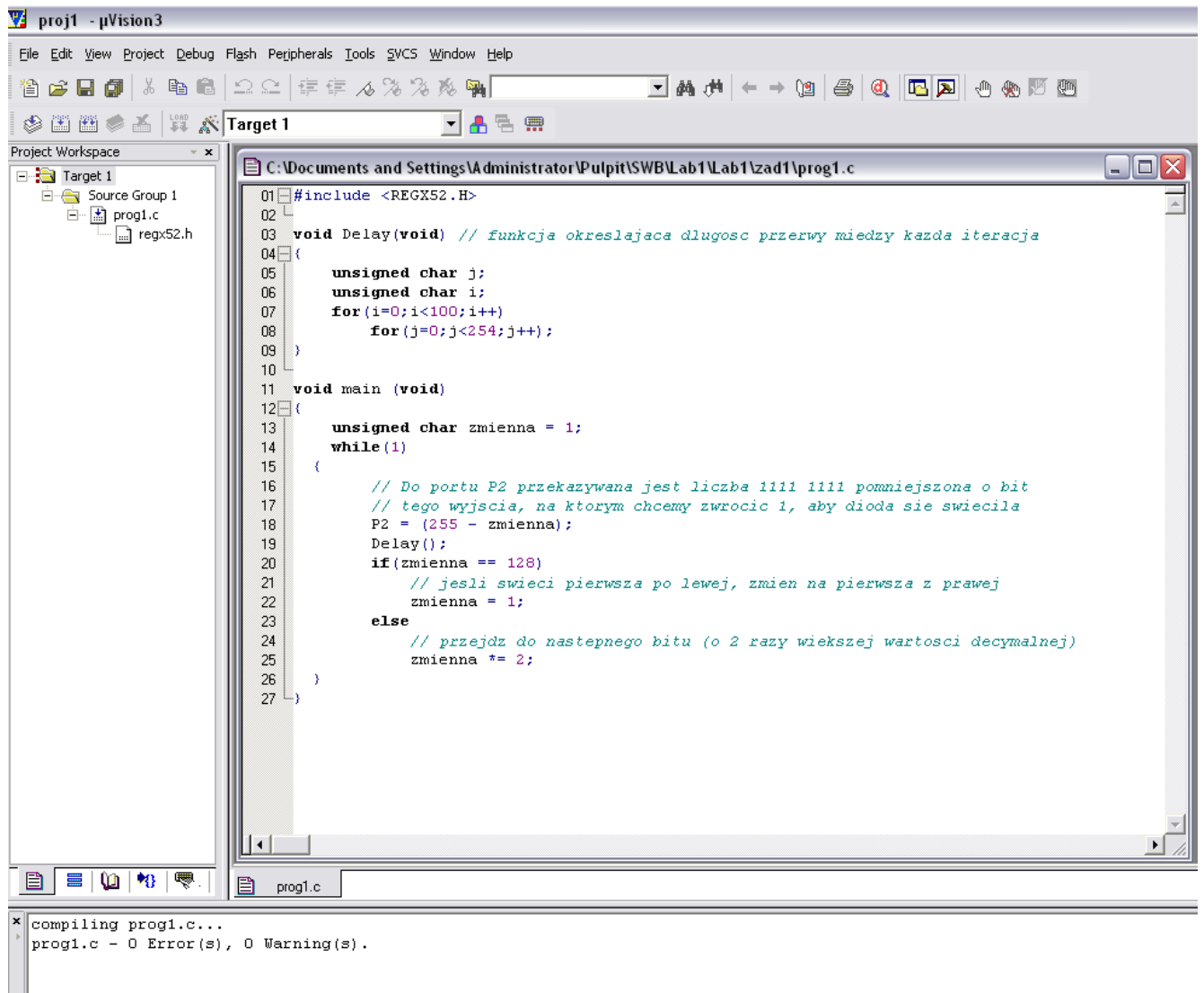
    unsigned char i;
    for(i=0;i<100;i++)
        for(j=0;j<254;j++);
}

void main (void)
{
    unsigned char zmienna = 1;
    while(1)
    {
        // Do portu P2 przekazywana jest liczba 1111 1111 pomniejszona o bit
        // tego wyjścia, na którym chcemy zwrocic 1, aby dioda sie swieciła
        P2 = (255 - zmienna);
        Delay();
        if(zmienna == 128)
            // jeśli świeci pierwsza po lewej, zmien na pierwszą z prawej
            zmienna = 1;
        else
            // przejdź do następnego bitu (o 2 razy większej wartości decymalnej)
            zmienna *= 2;
    }
}

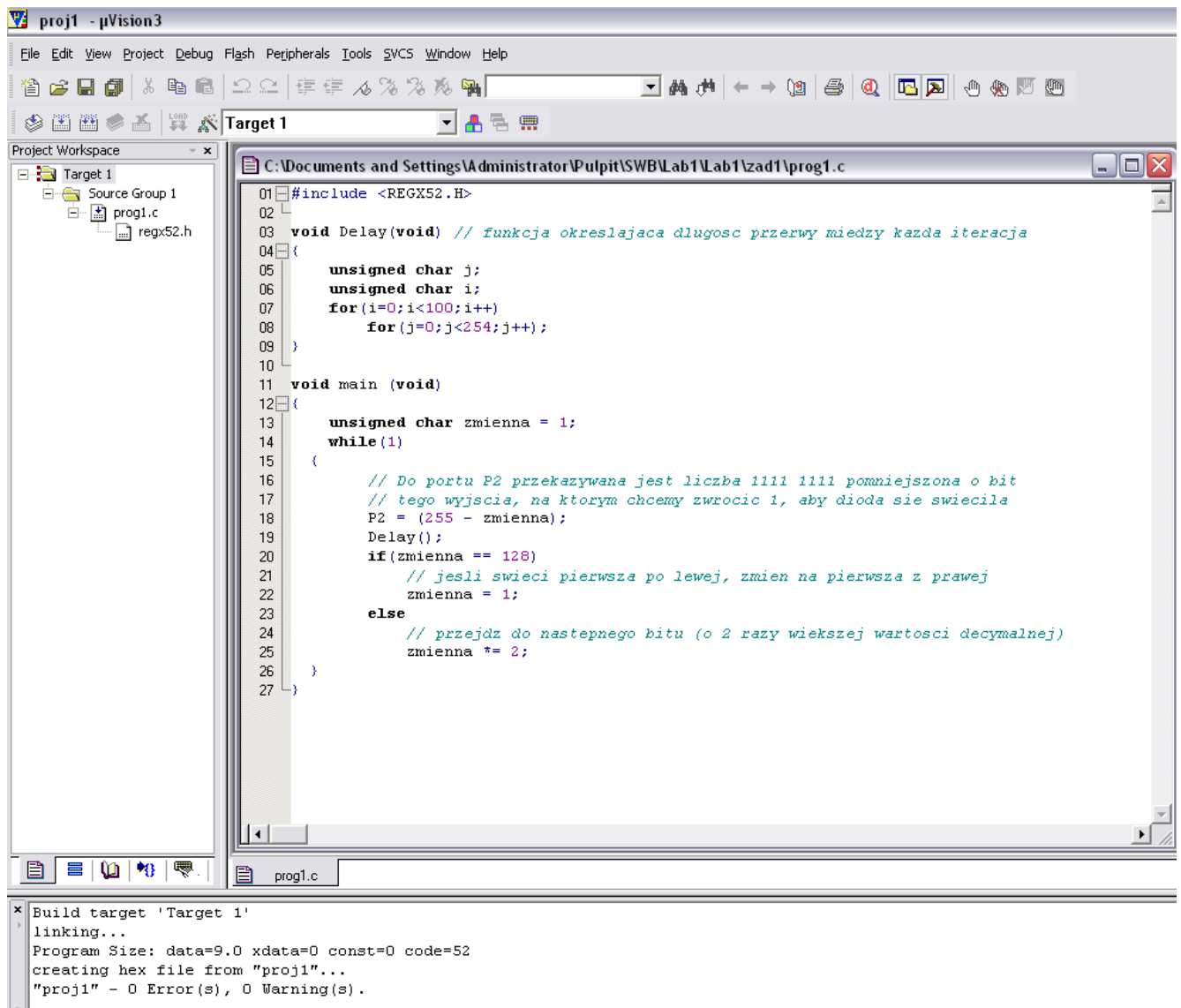
```

Sprawdzenie poprawności

Kompilowanie



Linkowanie



Prezentacja realizacji zadania przez program

Breakpoint ustawiam na linii 19 (w momencie wywołania funkcji „Delay”). Kolejne iteracje będą przedstawiały zmiany

Keil:

proj1 - uVision3

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x01
r6	0x00
r7	0x00

Sys

a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC	0x0000
states	394
sec	0.0034...
psw	0x01

C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad1\proj1.c

```

01 #include <REG52.H>
02
03 void Delay(void) // funkcja okreslajaca dlugosc przerwy miedzy kazda iteracja
04 {
05     unsigned char j;
06     unsigned char i;
07     for(i=0; i<100; i++)
08         for(j=0; j<254; j++);
09 }
10
11 void main (void)
12 {
13     unsigned char zmienna = 1;
14     while(1)
15     {
16         // Do portu P2 przekazywana jest liczba 1111 1111 pomniejszona o bit
17         // tego wyjscia, na którym chcemy zwrócic 1, aby dioda sie swieciła
18         P2 = (255 - zmienna);
19         Delay();
20         if(zmienna == 128)
21             // jeśli świeci pierwsza po lewej, zmien na pierwszą z prawej
22             zmienna = 1;
23         else
24             // przejdź do następnego bitu (o 2 razy większej wartości decymalnej)
25             zmienna *= 2;
26     }
27 }

```

Parallel Port 2

Port 2

P2: 0xFE 7 Bits 0

Pins: 0xFE

Output Window

Load "C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad1\proj1"

BS \PROG1:19

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALUATE EXIT FUNC Go

Simulation t: 0.00342014 sec L: 27 C: 51 R/W

proj1 - uVision3

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x02
r6	0x00
r7	0x64

Sys

a	0x00
b	0x00
sp	0x07
sp_max	0x09
dptr	0x0000
PC	0x0000
states	77111
sec	0.6693...
psw	0x01

C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad1\proj1.c

```

01 #include <REG52.H>
02
03 void Delay(void) // funkcja okreslajaca dlugosc przerwy miedzy kazda iteracja
04 {
05     unsigned char j;
06     unsigned char i;
07     for(i=0; i<100; i++)
08         for(j=0; j<254; j++);
09 }
10
11 void main (void)
12 {
13     unsigned char zmienna = 1;
14     while(1)
15     {
16         // Do portu P2 przekazywana jest liczba 1111 1111 pomniejszona o bit
17         // tego wyjscia, na którym chcemy zwrócic 1, aby dioda sie swieciła
18         P2 = (255 - zmienna);
19         Delay();
20         if(zmienna == 128)
21             // jeśli świeci pierwsza po lewej, zmien na pierwszą z prawej
22             zmienna = 1;
23         else
24             // przejdź do następnego bitu (o 2 razy większej wartości decymalnej)
25             zmienna *= 2;
26     }
27 }

```

Parallel Port 2

Port 2

P2: 0xFD 7 Bits 0

Pins: 0xFD

Output Window

Load "C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad1\proj1"

BS \PROG1:19

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALUATE EXIT FUNC Go

Simulation t: 0.66936632 sec L: 27 C: 78 R/W

proj1 - µVision3

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

Register Value

Reg	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x04
r6	0x0e
r7	0x64

Sys

a	0x0b
b	0x00
sp	0x07
sp_max	0x09
dptr	0x0000
PC	0x0000
states	153828
sec	1.33531250
psw	0x01

C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad1\proj1.c

```

01 #include <REG52.H>
02
03 void Delay(void) // funkcja okreslajaca dlugosc przerwy miedzy kazda iteracja
04 {
05     unsigned char j;
06     unsigned char i;
07     for(i=0;i<100;i++)
08         for(j=0;j<254;j++);
09 }
10
11 void main (void)
12 {
13     unsigned char zmienna = 1;
14     while(1)
15     {
16         // Do portu P2 przekazywana jest liczba 1111 1111 pomniejszona o bit
17         // tego wyjscia, na którym chcemy zwrócic 1, aby dioda sie swieciła
18         P2 = (255 - zmienna);
19         Delay();
20         if(zmienna == 128)
21             // jeśli świeci pierwsza po lewej, zmien na pierwszą z prawej
22             zmienna = 1;
23         else
24             // przejdź do następnego bitu (o 2 razy większej wartości decymalnej)
25             zmienna *= 2;
26     }
27 }

```

Parallel Port 2

Port 2

P2: 0x0b 7 Bits 0

Pins: 0x0b 7 Bits 0

Output Window

Load "C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad1\proj1"

BS \PROG1:19

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALUATE EXIT FUNC Go

Simulation t: 1.33531250 sec L: 19 C: 1 R/W

proj1 - µVision3

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

Register Value

Reg	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x08
r6	0x0e
r7	0x64

Sys

a	0x07
b	0x00
sp	0x07
sp_max	0x09
dptr	0x0000
PC	0x0000
states	230545
sec	2.00125868
psw	0x01

C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad1\proj1.c

```

01 #include <REG52.H>
02
03 void Delay(void) // funkcja okreslajaca dlugosc przerwy miedzy kazda iteracja
04 {
05     unsigned char j;
06     unsigned char i;
07     for(i=0;i<100;i++)
08         for(j=0;j<254;j++);
09 }
10
11 void main (void)
12 {
13     unsigned char zmienna = 1;
14     while(1)
15     {
16         // Do portu P2 przekazywana jest liczba 1111 1111 pomniejszona o bit
17         // tego wyjscia, na którym chcemy zwrócic 1, aby dioda sie swieciła
18         P2 = (255 - zmienna);
19         Delay();
20         if(zmienna == 128)
21             // jeśli świeci pierwsza po lewej, zmien na pierwszą z prawej
22             zmienna = 1;
23         else
24             // przejdź do następnego bitu (o 2 razy większej wartości decymalnej)
25             zmienna *= 2;
26     }
27 }

```

Parallel Port 2

Port 2

P2: 0x07 7 Bits 0

Pins: 0x07 7 Bits 0

Output Window

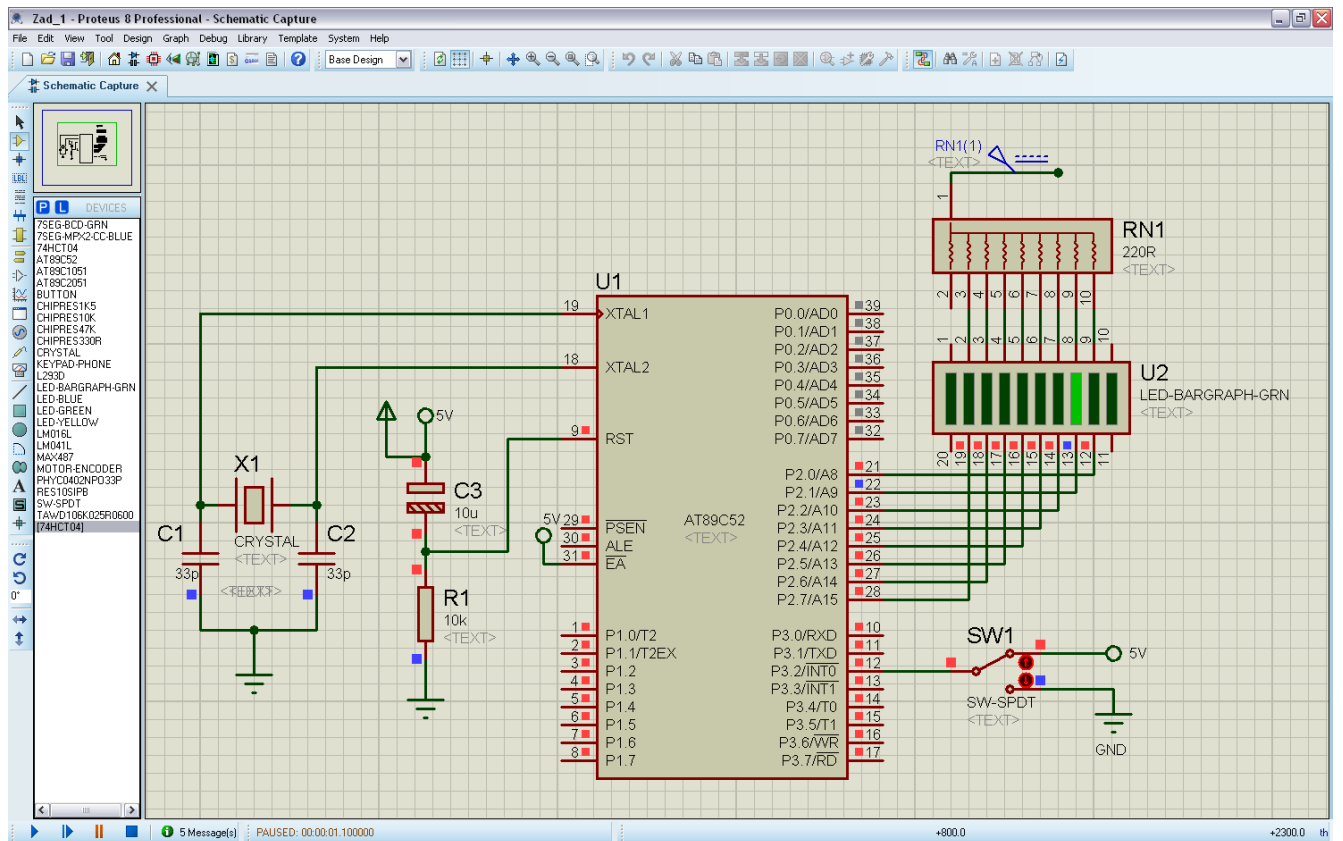
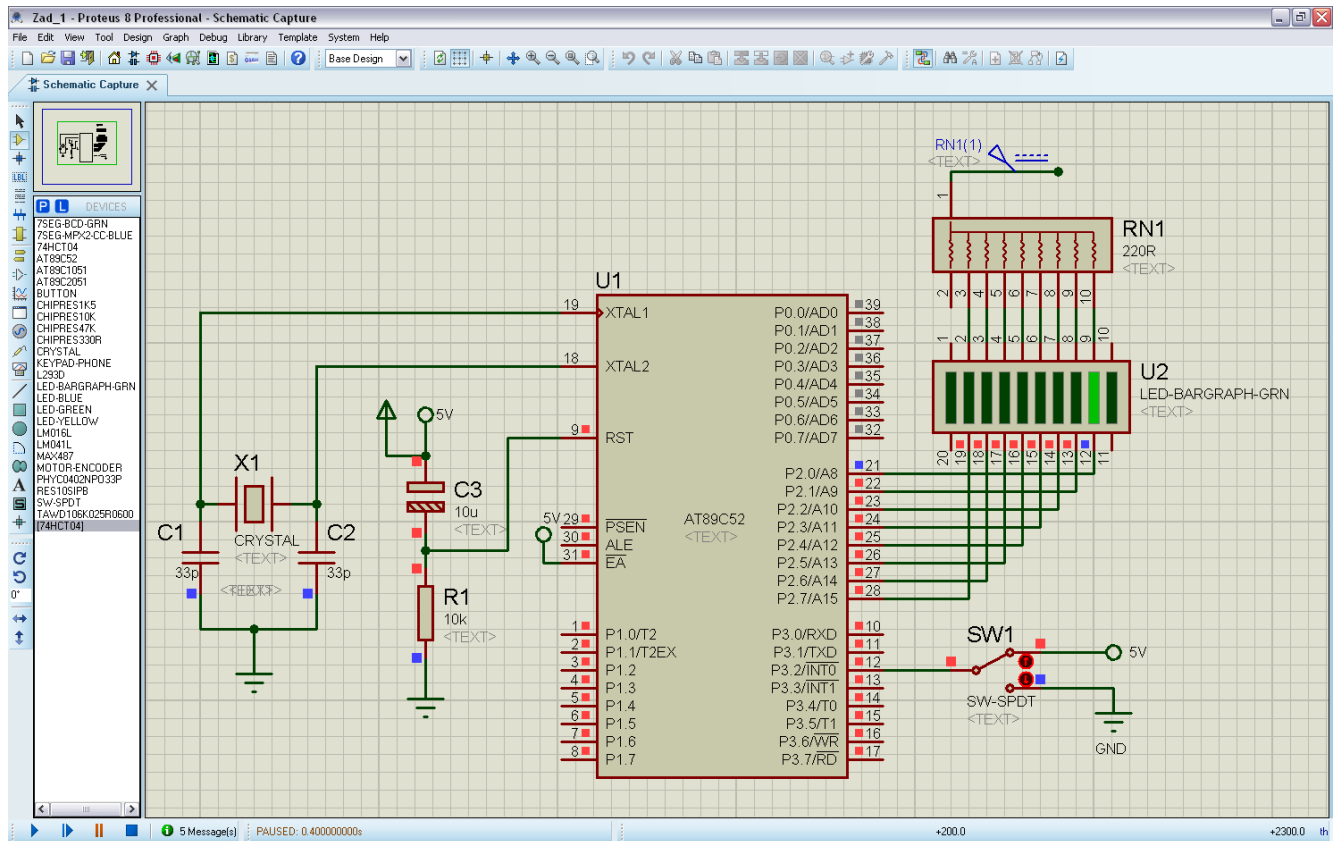
Load "C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad1\proj1"

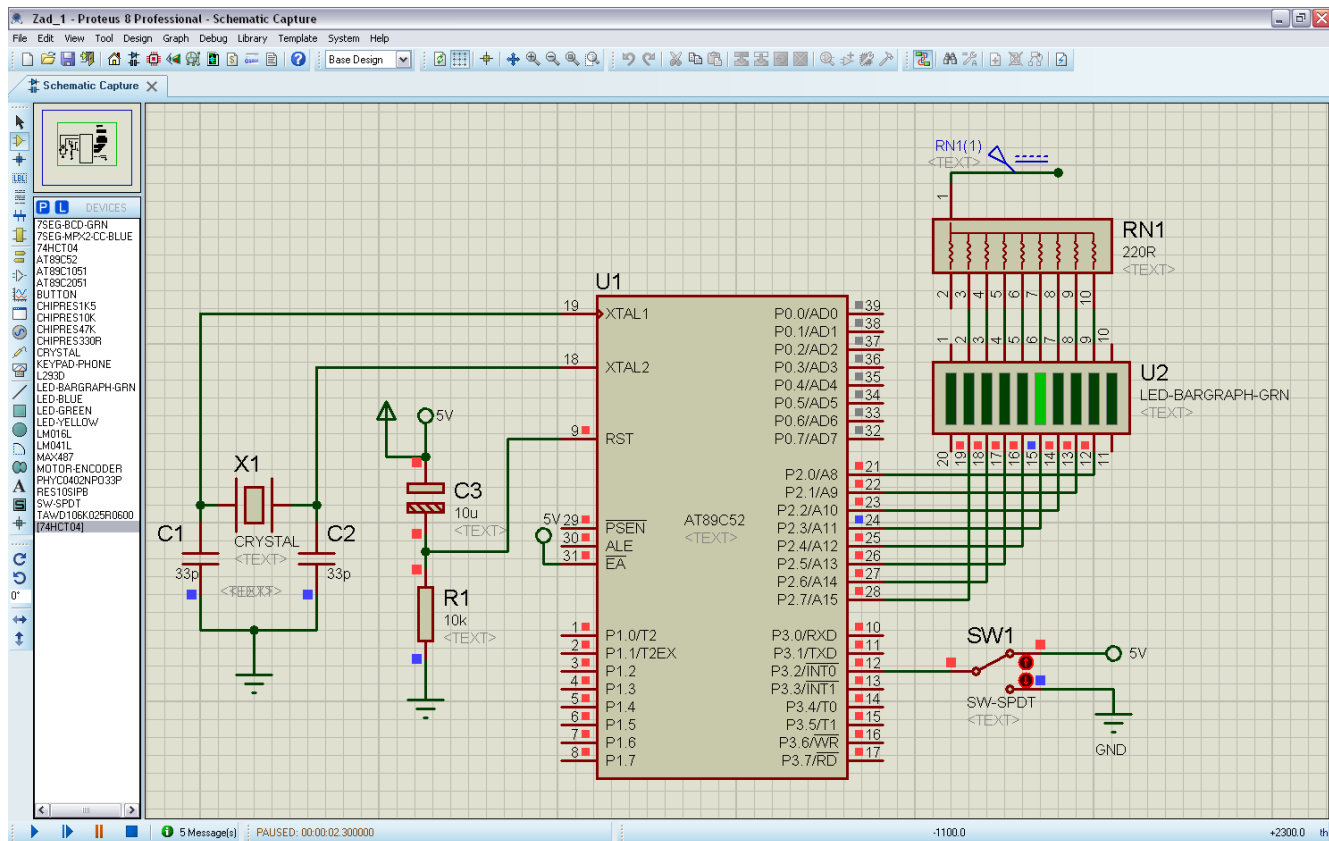
BS \PROG1:19

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALUATE EXIT FUNC Go

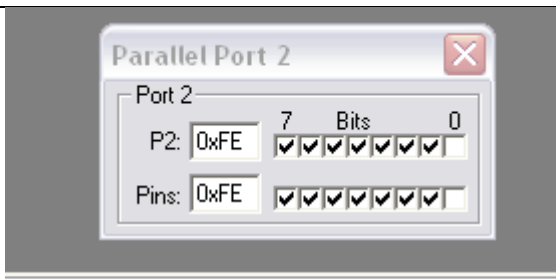

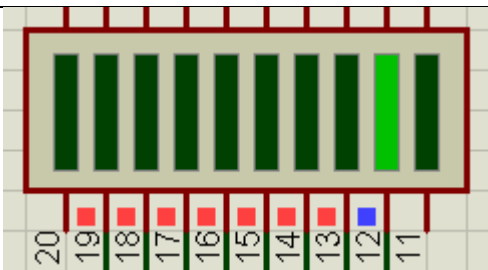
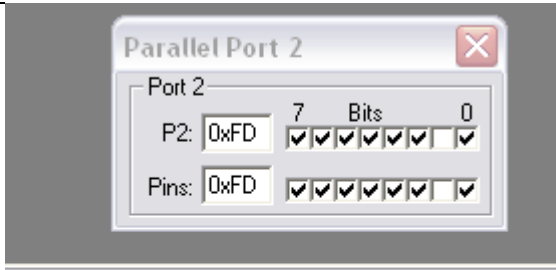

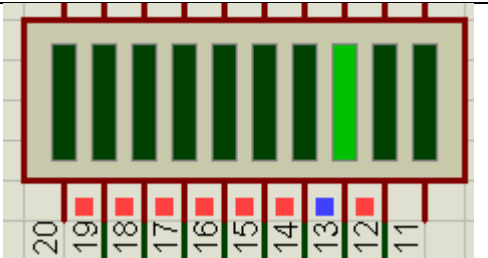
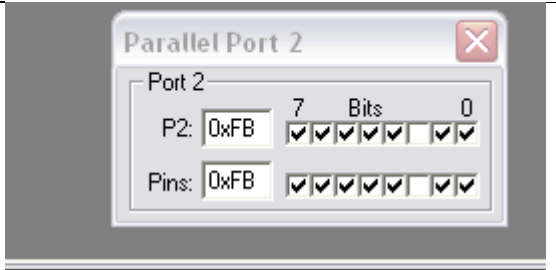
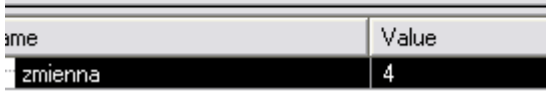
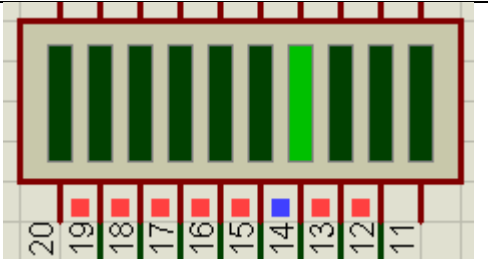
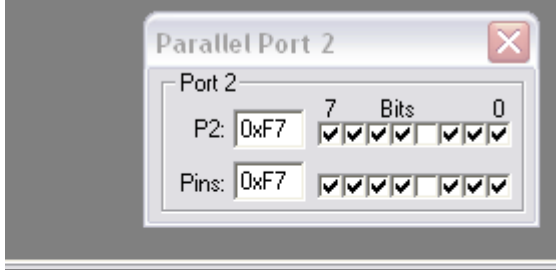
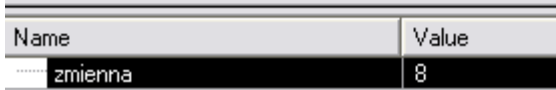
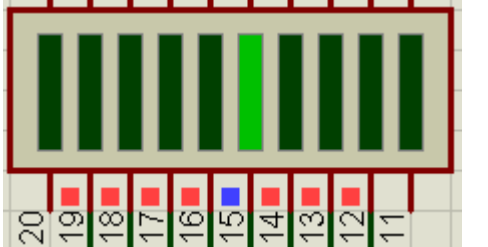
Simulation t: 2.00125868 sec L: 24 C: 54 R/W

Proteus:





Krok	Keil – wartość zmiennej oraz portu	Proteus – stan diod
------	------------------------------------	---------------------

1	 	
2	 	
3	 	
4	 	

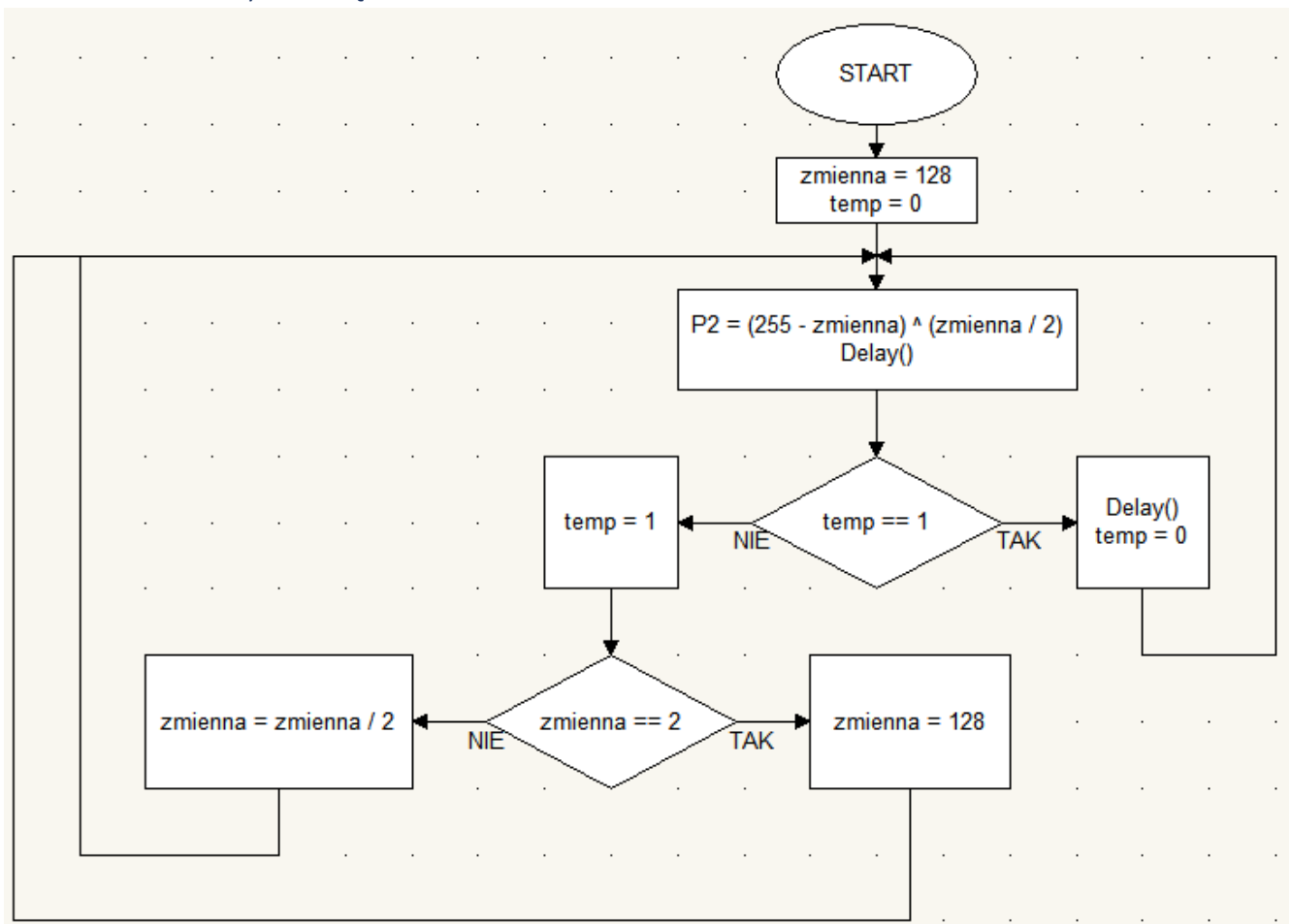
C. Zadanie na ocenę dobrą

Opis mojego rozwiązania

Zadanie wykonałem podobnie do zadania za ocenę dostateczną. Będę dodatkowo zerować prawy bit od tego wskazywane przez zmienną. Stworzę warunek, który przy pomocy dodatkowej zmiennej cyklicznie w ramach jednej iteracji będzie albo robił opóźnienie albo wykonywał następne przejście programu (da to sytuację

podwójnego opóźnienia co drugie przejście). Zamknięcie cyklu następuje przy liczbie 2, a nie 1, żeby pominąć sytuację, gdy powinny zapalić się 2 skrajne diody.

Schemat blokowy rozwiązania



Listing programu

```
#include <REGX52.H>

void Delay(void) // funkcja okreslajaca dlugosc przerwy miedzy kazda iteracja
{
    unsigned char j;
    unsigned char i;
    for(i=0;i<100;i++)
        for(j=0;j<254;j++);
}

void main (void)
{
    unsigned char zmienna = 128;
    bit temp = 0; // flaga do podwojenia przerwy miedzy co druga iteracja
    while(1)
    {
        // Chcemy zapalic lampke jak poprzednio oraz na prawo od niej
        // Operacja '^' zaklada przypadek, gdzie chcemy zapalic 2 skrajne, wtedy nie
        // mozemy odejmowac
        P2 = ((255 - zmienna) ^ (zmienna / 2));
        Delay();
        if(temp==1)
```

```

{
    // wykonuje sie co druga operacje
    Delay();
    temp=0;
}
else
    temp=1;
// Robimy przeskok na 128 z 2, a nie z 1, zeby
// pominac krok ze swieceniem sie po obu stronach
if(zmienna == 2)
    zmienna = 128;
else
    zmienna /= 2;
}
}

```

Sprawdzenie poprawności

Kompilowanie

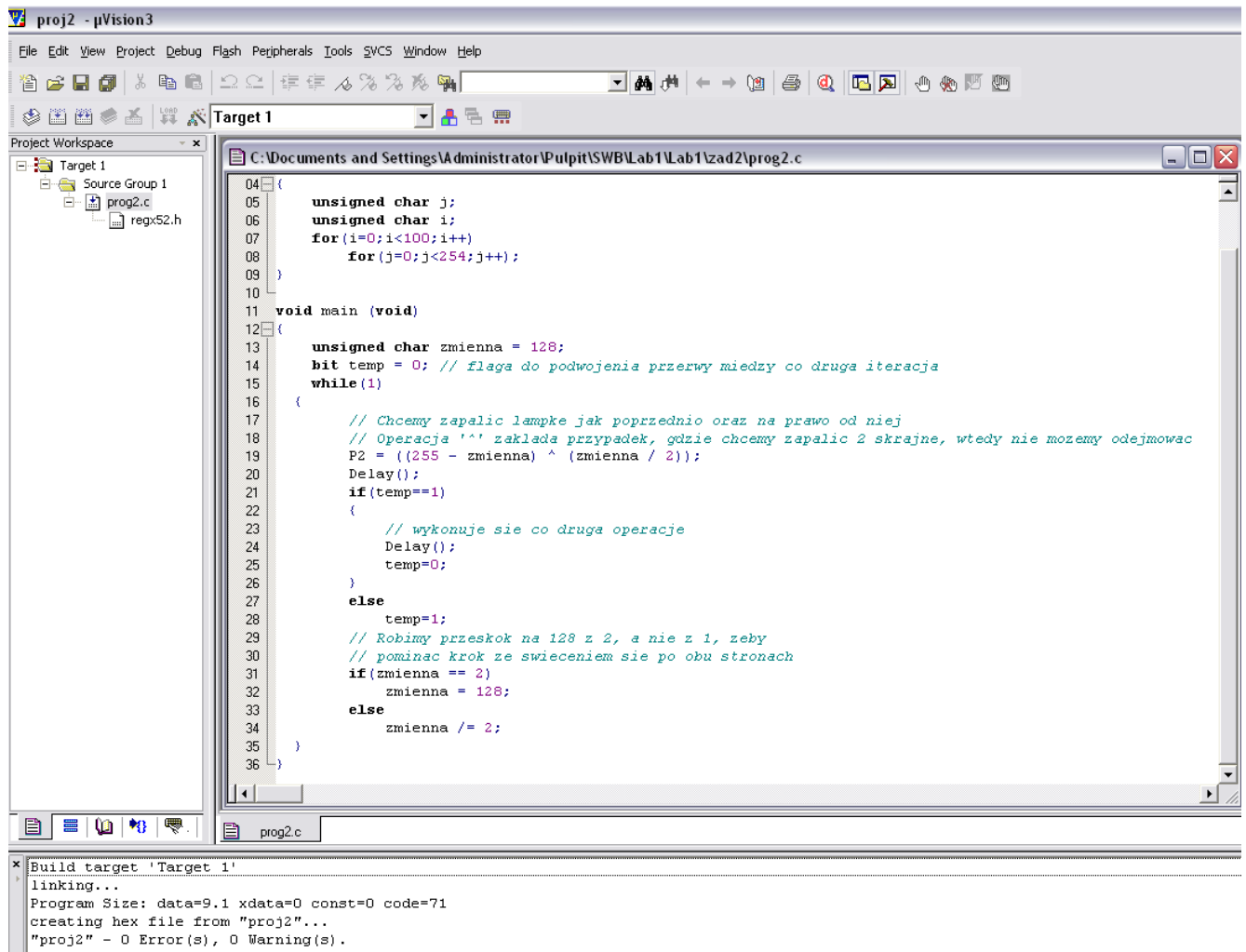
```

04 {
05     unsigned char j;
06     unsigned char i;
07     for(i=0; i<100; i++)
08         for(j=0; j<254; j++)
09     }
10
11 void main (void)
12 {
13     unsigned char zmienna = 128;
14     bit temp = 0; // flaga do podwojenia przerwy miedzy co druga iteracja
15     while(1)
16     {
17         // Chcemy zapalic lampke jak poprzednio oraz na prawo od niej
18         // Operacja '^' zaklada przypadek, gdzie chcemy zapalic 2 skrajne, wtedy nie mozemy odejmowac
19         P2 = ((255 - zmienna) ^ (zmienna / 2));
20         Delay();
21         if(temp==1)
22         {
23             // wykonuje sie co druga operacje
24             Delay();
25             temp=0;
26         }
27         else
28             temp=1;
29         // Robimy przeskok na 128 z 2, a nie z 1, zeby
30         // pominac krok ze swieceniem sie po obu stronach
31         if(zmienna == 2)
32             zmienna = 128;
33         else
34             zmienna /= 2;
35     }
36 }

```

compiling prog2.c...
prog2.c - 0 Error(s), 0 Warning(s).

Linkowanie



Prezentacja realizacji zadania przez program

Breakpoint ustawiam na linii 20 (w momencie wywołania funkcji „Delay”). Kolejne iteracje będą przedstawiały zmiany

Keil:

proj2 - pVision3

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x80
r6	0x00
r7	0x40
sp	0x20
sp_max	0x0000
pc	0x0000
states	400
sec	0.0034...
psw	0x00

04 {
05 unsigned char j;
06 unsigned char i;
07 for(i=0;i<100;i++)
08 for(j=0;j<254;j++)
09 }
10
11 void main(void)
12 {
13 unsigned char zmienna = 128;
14 bit temp = 0; // flaga do podwojenia przerwy miedzy co druga iteracja
15 while(1)
16 {
17 // Chcemy zapalic lampke jak poprzednio oraz na prawo od niej
18 // Operacja '<<' zaklada przypadek, gdzie chcemy zapalic 2 skrajne, wtedy nie mozemy odejmowac
19 P2 = ((255 - zmienna) ^ (zmienna / 2));
20 Delay();
21 if(temp==1)
22 {
23 // wykonuje sie co druga operacja
24 Delay();
25 temp=0;
26 }
27 else
28 temp=1;
29 // Robimy przeskok na 128 z 2, a nie z 1, zeby
30 // pominac krok ze swieceniem sie po obu stronach
31 if(zmienna == 2)
32 zmienna = 128;
33 else
34 zmienna /= 2;
35 }
36 }

Parallel Port 2

Port 2 7 Bits 0
P2: 0x3F ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒
Pins: 0x3F ☒ ☒ ☒ ☒ ☒ ☒ ☒

Load "C:\\Documents and Settings\\Administrator\\Pulpit\\SWB\\Lab1\\Lab1\\zad2\\proj2"

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVAluate EXIT FUNC Go

Output Window

Simulation t1: 0.00347222 sec L:15 C:30 R/W

proj2 - pVision3

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x40
r6	0x00
r7	0x20
sp	0x20
sp_max	0x22
dpr	0x0000
pc	0x0000
states	77126
sec	0.6694...
psw	0x00

04 {
05 unsigned char j;
06 unsigned char i;
07 for(i=0;i<100;i++)
08 for(j=0;j<254;j++)
09 }
10
11 void main(void)
12 {
13 unsigned char zmienna = 128;
14 bit temp = 0; // flaga do podwojenia przerwy miedzy co druga iteracja
15 while(1)
16 {
17 // Chcemy zapalic lampke jak poprzednio oraz na prawo od niej
18 // Operacja '<<' zaklada przypadek, gdzie chcemy zapalic 2 skrajne, wtedy nie mozemy odejmowac
19 P2 = ((255 - zmienna) ^ (zmienna / 2));
20 Delay();
21 if(temp==1)
22 {
23 // wykonuje sie co druga operacja
24 Delay();
25 temp=0;
26 }
27 else
28 temp=1;
29 // Robimy przeskok na 128 z 2, a nie z 1, zeby
30 // pominac krok ze swieceniem sie po obu stronach
31 if(zmienna == 2)
32 zmienna = 128;
33 else
34 zmienna /= 2;
35 }
36 }

Parallel Port 2

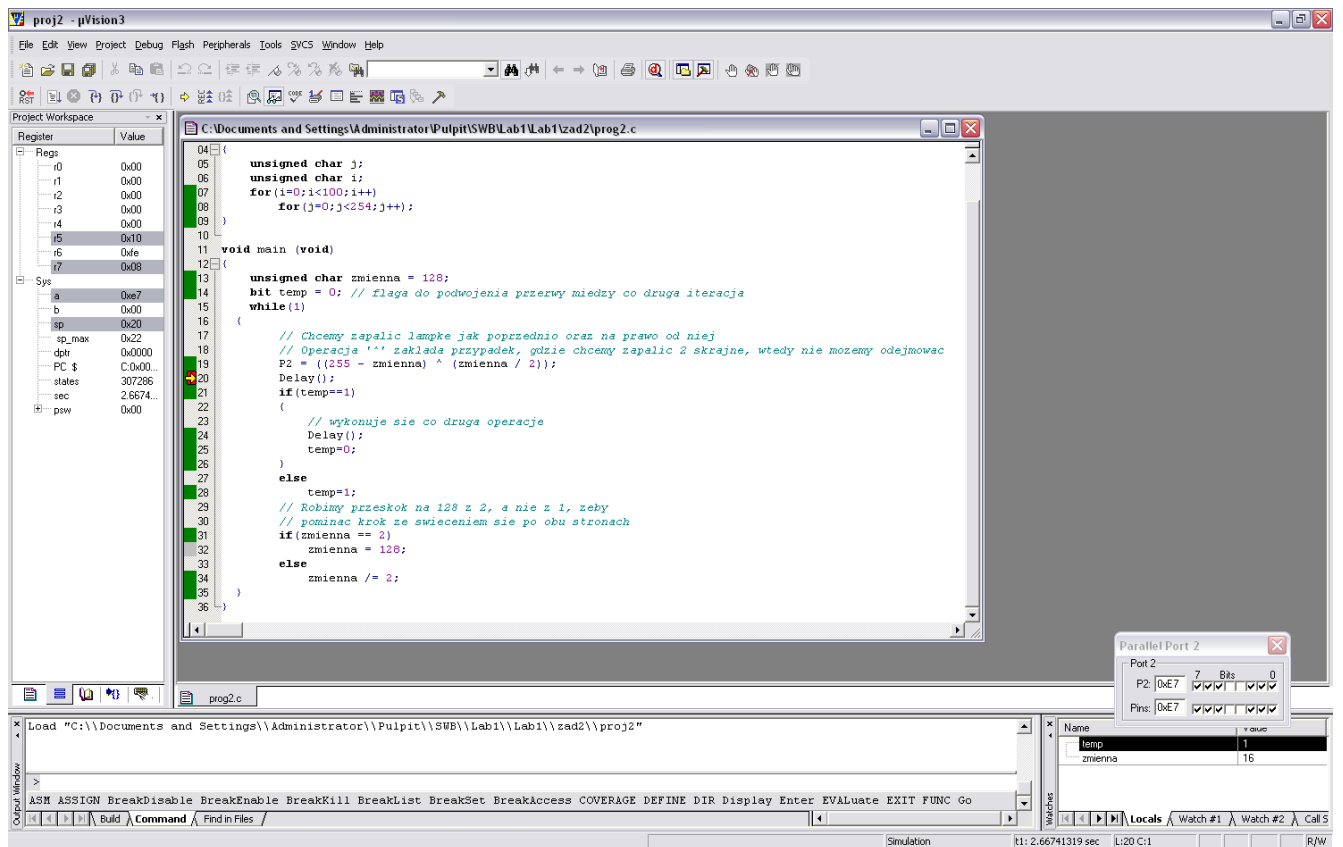
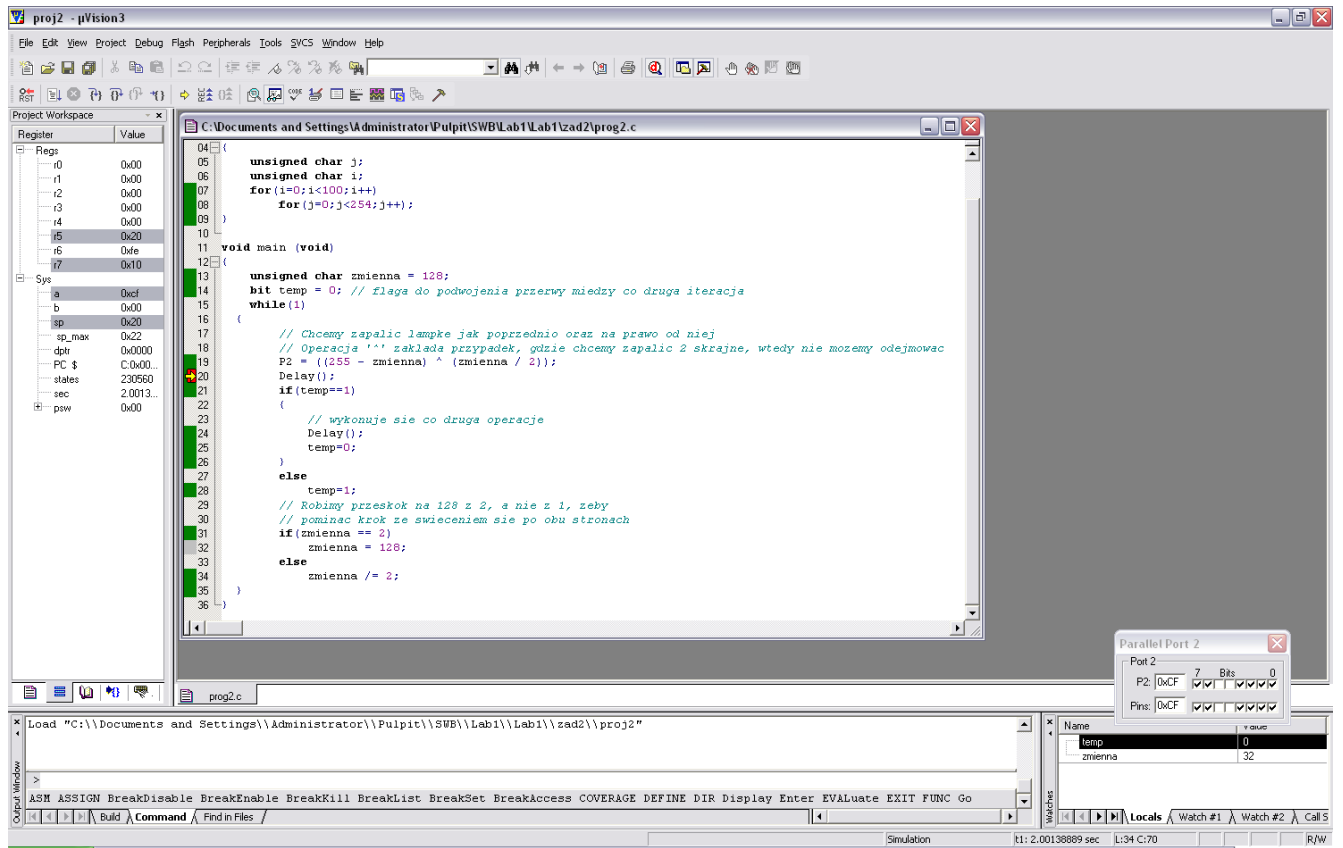
Port 2 7 Bits 0
P2: 0x3F ☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒
Pins: 0x3F ☒ ☒ ☒ ☒ ☒ ☒ ☒

Load "C:\\Documents and Settings\\Administrator\\Pulpit\\SWB\\Lab1\\Lab1\\zad2\\proj2"

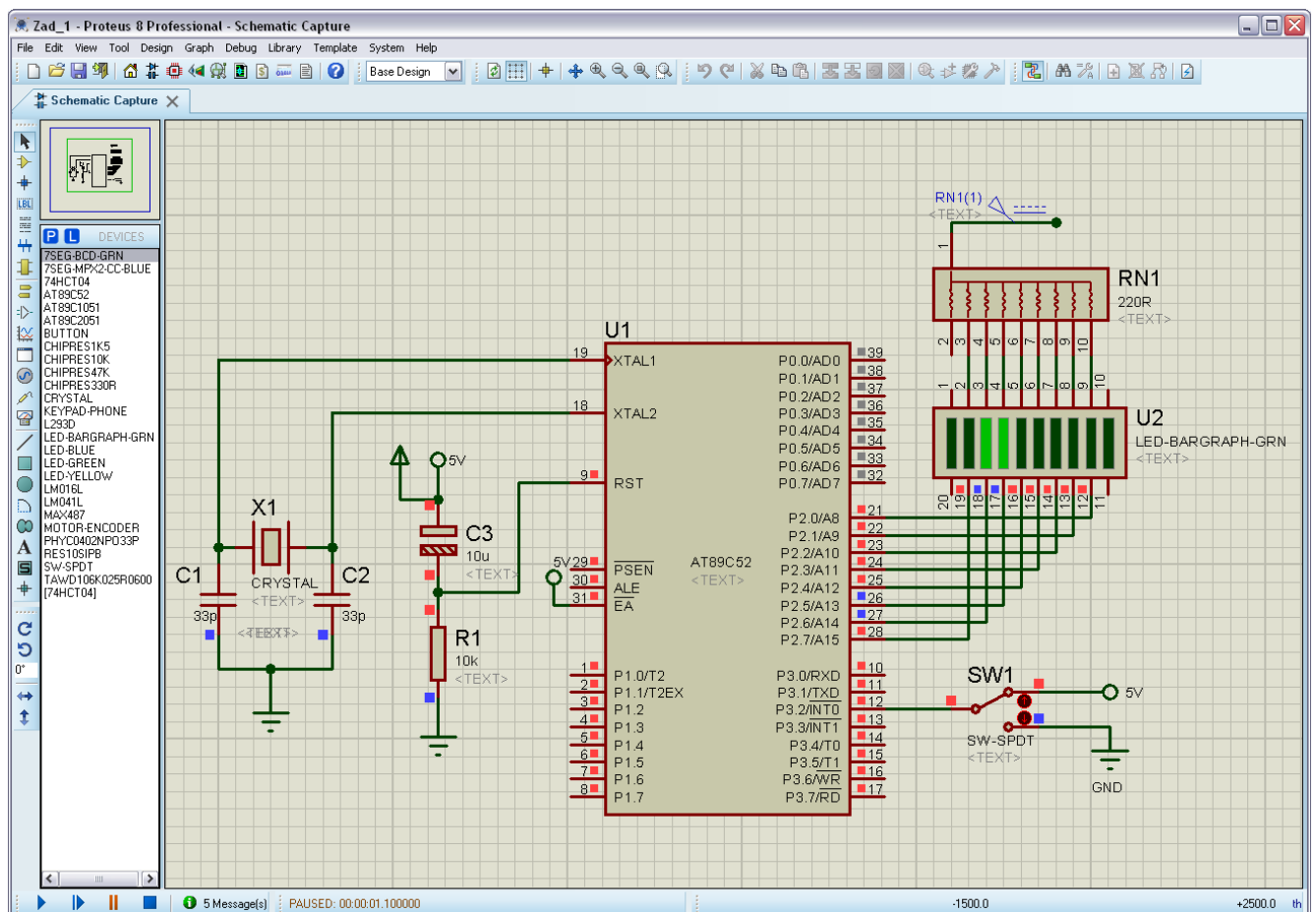
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVAluate EXIT FUNC Go

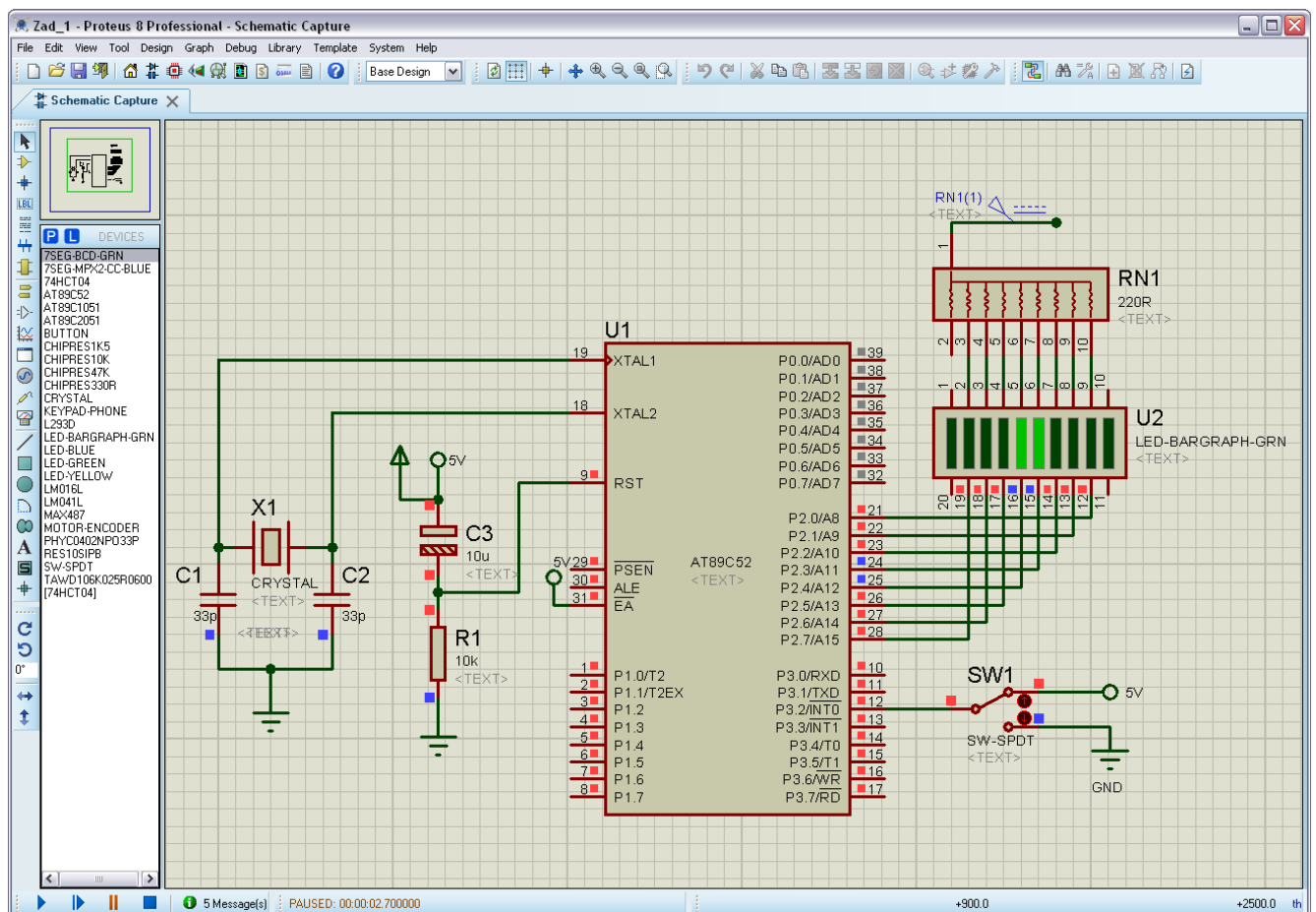
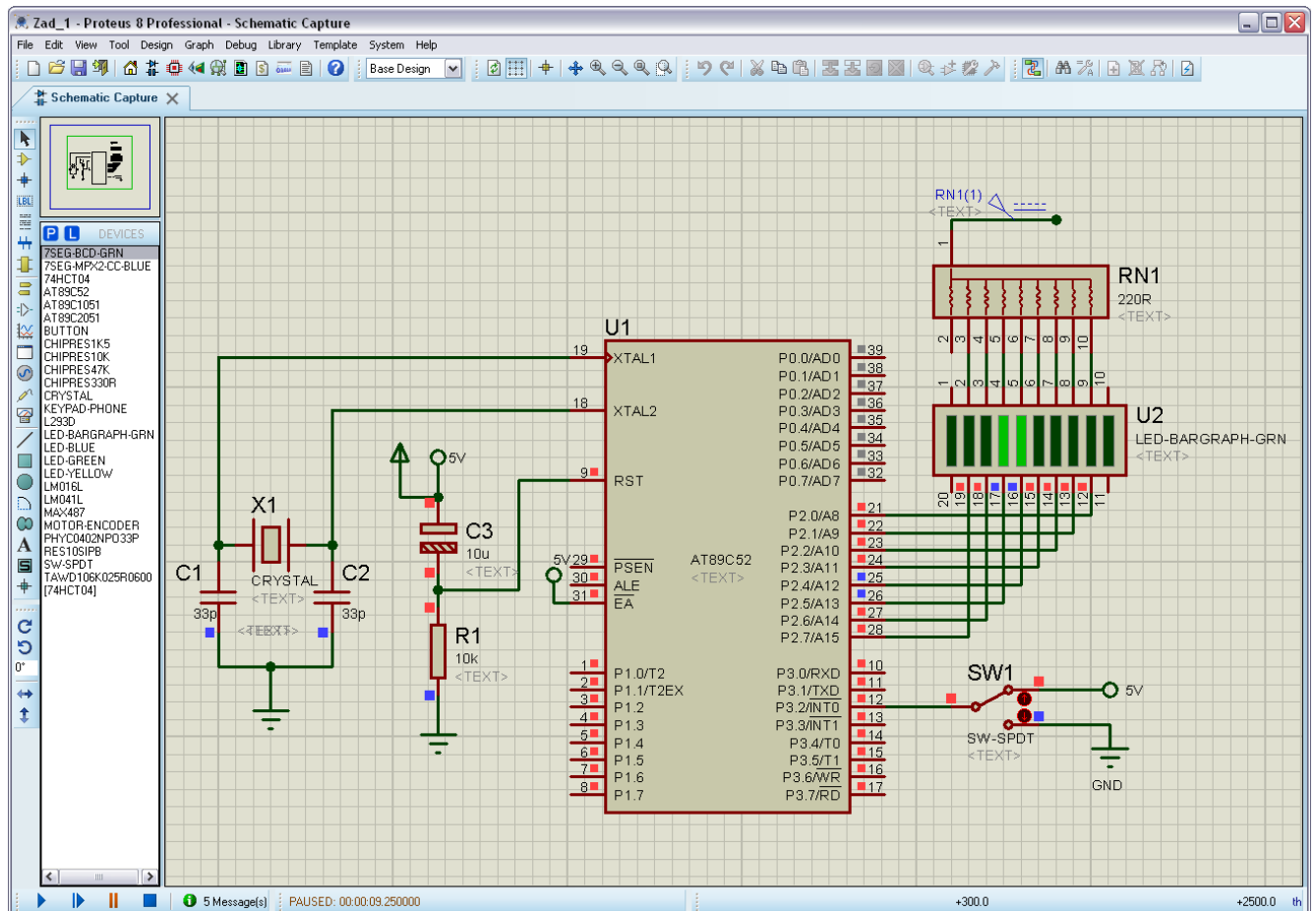
Output Window

Simulation t1: 0.66949653 sec L:20 C:1 R/W



Proteus:





Podsumowanie:

Krok	Keil – wartości zmiennych oraz portu	Proteus – stan diod
------	--------------------------------------	---------------------

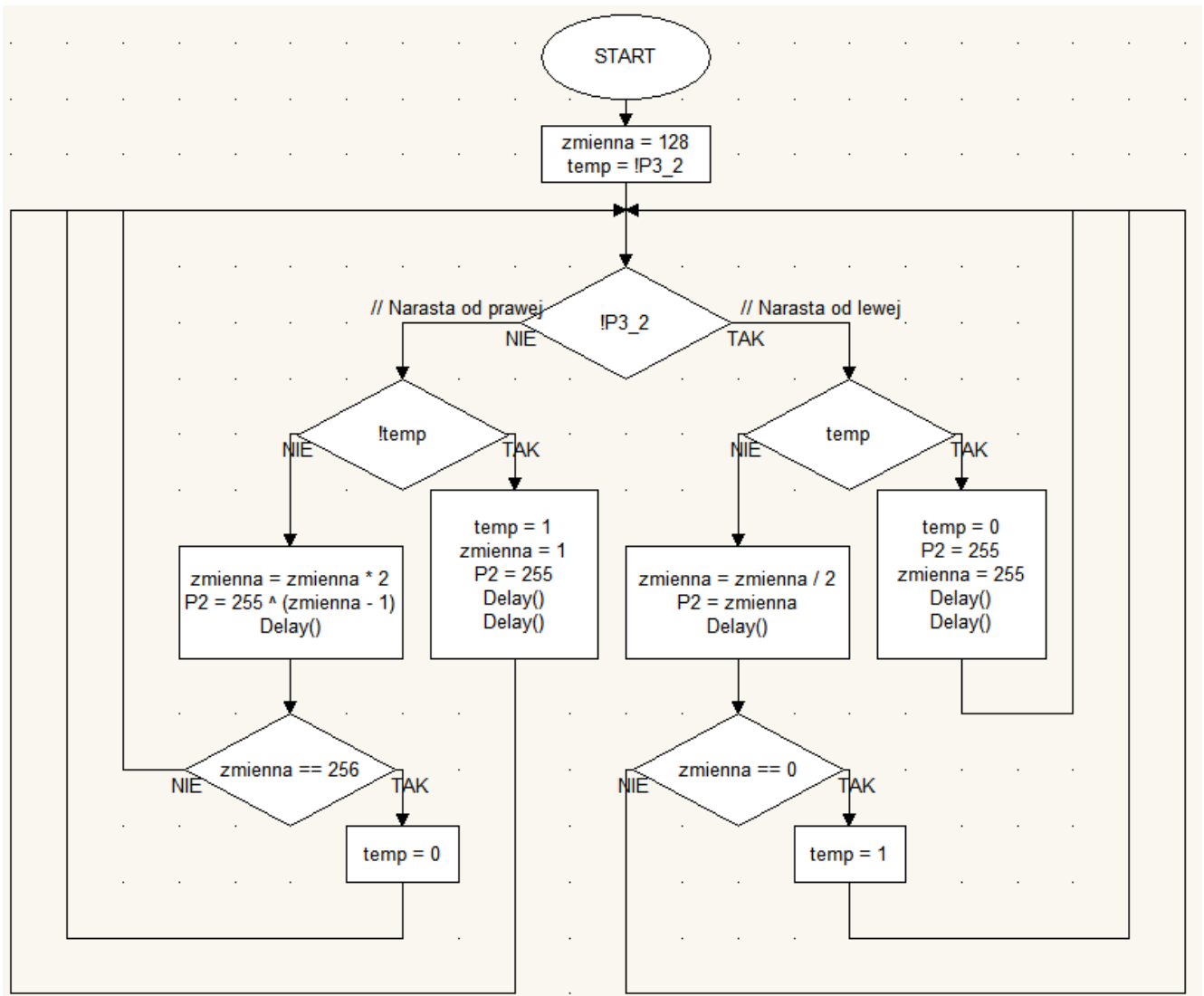
1		
2		
3		
4		

D. Zadanie na ocenę bardzo dobrą

Opis mojego rozwiązania

W ramach tego zadania zacznę od sprawdzania przy każdej iteracji stanu wejścia, żeby móc określić, z której strony powinno teraz narastać. Dodatkowo, będę sprawdzał też, czy są już zapalone wszystkie diody, żeby w razie takiej sytuacji wrócić do stanu początkowego. Dla narastania z lewej będę zwyczajnie dzielił bieżącą wartość przez 2, co wyeliminuje bit o największej wartości. Natomiast dla narastania z prawej, będę od wartości 255 odejmował wartość $2^x - 1$, dla x rosnącego o 1 zaczynając od 0.

Schemat blokowy rozwiązania



Listing programu

```
#include <REGX52.H>

void Delay(void)
{
    unsigned char j;
    unsigned char i;
    for(i=0;i<100;i++)
        for(j=0;j<254;j++);
}

void main (void)
{
    unsigned char zmienna;
    // zmienna temp sluzy do sprawdzania, kiedy zostana
    // zapalone wszystkie diody i trzeba je zgasic
    bit temp = !P3_2;
    while(1)
    {
        if(!P3_2)
        {
            // Narasta od lewej
```

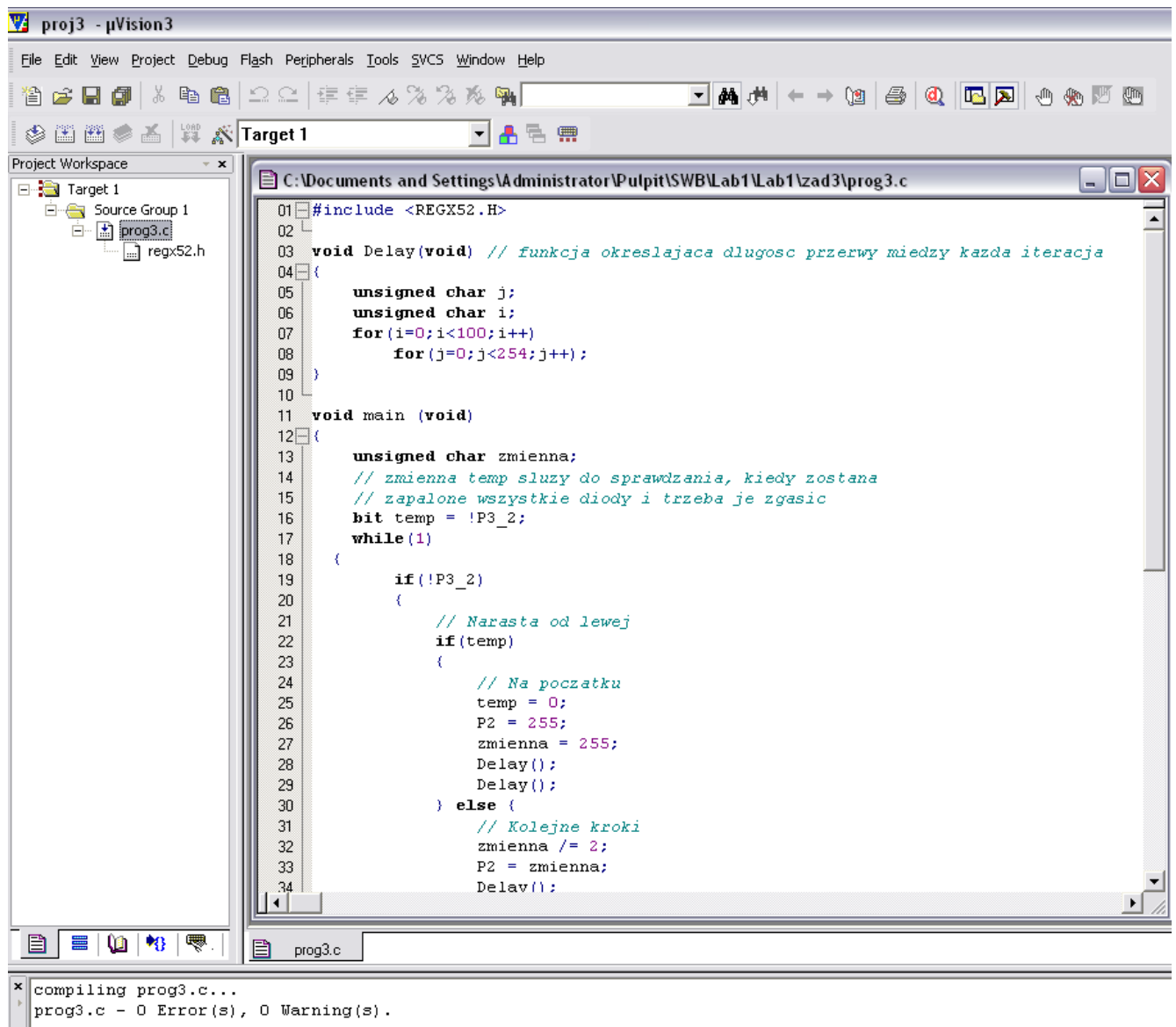
```

    if(temp)
    {
        // Na początku
        temp = 0;
        P2 = 255;
        zmienna = 255;
        Delay();
        Delay();
    } else {
        // Kolejne kroki
        zmienna /= 2;
        P2 = zmienna;
        Delay();
        if(zmienna == 0)
            // Jak wszystkie sie zapala, to zerujemy
            temp = 1;
    }
}
else
{
    //Narasta od prawej
    if(!temp)
    {
        // Na początku
        temp = 1;
        zmienna = 1;
        P2 = 255;
        Delay();
        Delay();
    } else {
        // Kolejne kroki
        zmienna *= 2;
        P2 = 255 ^ (zmienna-1);
        Delay();
        if((zmienna-1) == 255)
            // Jak wszystkie sie zapala, to zerujemy
            temp = 0;
    }
}
}
}

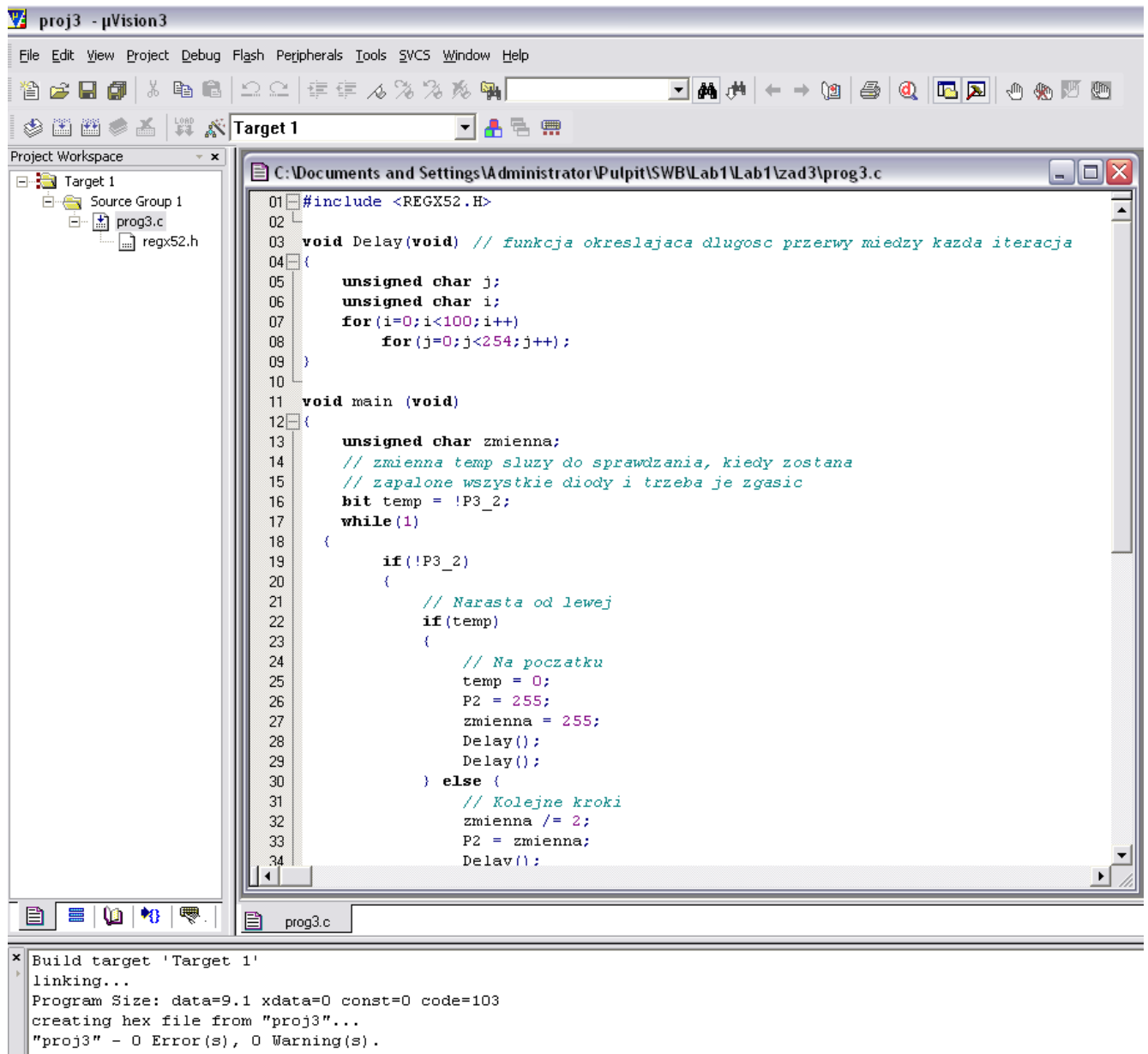
```

Sprawdzenie poprawności

Kompilowanie



Linkowanie



Prezentacja realizacji zadania przez program

Breakpointy ustawiam na instrukcjach warunkowych sprawdzających, czy należy już wyzerować diody.

Keil:

proj3 - pVision3 - [C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad3\proj3.c]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x01
r6	0xfe
r7	0x64

System

a	0x00
b	0x00
sp	0x20
sp_max	0x22
dptr	0x0000
PC	\$ C:0x00...
states	153817
sec	1.3352...
psw	0x00

```

13 unsigned char zmienna;
14 // zmienna temp sluzi do sprawdzania, kiedy zostana
15 // zapalone wszystkie diody i trzeba je zgasic
16 bit temp = !P3_2;
17 while(1)
18 {
19     if(!P3_2)
20     {
21         // Narasta od lewej
22         if(temp)
23         {
24             // Na poczatku
25             temp = 0;
26             P2 = 255;
27             zmienna = 255;
28             Delay();
29             Delay();
30         } else {
31             // Kolejne kroki
32             zmienna /= 2;
33             P2 = zmienna;
34             Delay();
35             if(zmienna == 0)
36                 // Jak wszystkie sie zapala, to zerujemy
37                 temp = 1;
38         }
39     }
40     else
41     {
42         //Narasta od prawej
43         if(!temp)
44         {
45             // Na poczatku
46             temp = 1;
47             zmienna = 1;
48             P2 = 255;
49             Delay();
50             Delay();
51         } else {

```

Parallel Port 2

Port 2 7 Bits 0

P2: 0xFF

Pins: 0xFF

Output Window

Load "C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad3\proj3"

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go

Build Command Find in Files

Simulation t: 1.33521701 sec L:43 C:1 R/W

proj3 - pVision3 - [C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad3\proj3.c]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x02
r6	0xfe
r7	0x01

System

a	0x01
b	0x00
sp	0x20
sp_max	0x22
dptr	0x0000
PC	\$ C:0x00...
states	230538
sec	2.0011...
psw	0x81

```

13 unsigned char zmienna;
14 // zmienna temp sluzi do sprawdzania, kiedy zostana
15 // zapalone wszystkie diody i trzeba je zgasic
16 bit temp = !P3_2;
17 while(1)
18 {
19     if(!P3_2)
20     {
21         // Narasta od lewej
22         if(temp)
23         {
24             // Na poczatku
25             temp = 0;
26             P2 = 255;
27             zmienna = 255;
28             Delay();
29             Delay();
30         } else {
31             // Kolejne kroki
32             zmienna /= 2;
33             P2 = zmienna;
34             Delay();
35             if(zmienna == 0)
36                 // Jak wszystkie sie zapala, to zerujemy
37                 temp = 1;
38         }
39     }
40     else
41     {
42         //Narasta od prawej
43         if(!temp)
44         {
45             // Na poczatku
46             temp = 1;
47             zmienna = 1;
48             P2 = 255;
49             Delay();
50             Delay();
51         } else {

```

Parallel Port 2

Port 2 7 Bits 0

P2: 0xFE

Pins: 0xFE

Output Window

Load "C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad3\proj3"

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go

Build Command Find in Files

Simulation t: 2.00119792 sec L:38 C:77 R/W

proj3 - pVision3 - [C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad3\proj3.c]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x04
r6	0x0e
r7	0x03

Sys

a	0x03
b	0x00
sp	0x20
sp_max	0x22
dptr	0x0000
PC	0x0000
states	307259
sec	2.6671...
psw	0x80

```

25      temp = 0;
26      P2 = 255;
27      zmienna = 255;
28      Delay();
29      Delay();
30  } else {
31      // Kolejne kroki
32      zmienna /= 2;
33      P2 = zmienna;
34      Delay();
35      if(zmienna == 0)
36          // Jak wszystkie sie zapala, to zerujemy
37          temp = 1;
38  }
39  }
40  else
41  {
42      //Narasta od prawej
43      if(!temp)
44      {
45          // Na poczatku
46          temp = 1;
47          zmienna = 1;
48          P2 = 255;
49          Delay();
50          Delay();
51      } else {
52          // Kolejne kroki
53          zmienna *= 2;
54          P2 = 255 ^ (zmienna-1);
55          Delay();
56          if((zmienna-1) == 255)
57              // Jak wszystkie sie zapala, to zerujemy
58              temp = 0;
59      }
60  }
61  }
62  }

```

Output Window

Load "C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad3\proj3"

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go

Simulation t1: 2.66717882 sec L:43 C:1 R/W

Parallel Port 2

Port 2	7 Bits	0
P2	0x0C	0
Pins	0x0C	0

Watches

Name	Value
temp	1
zmienna	4

proj3 - pVision3 - [C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad3\proj3.c]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x08
r6	0x0e
r7	0x07

Sys

a	0x07
b	0x00
sp	0x20
sp_max	0x22
dptr	0x0000
PC	0x0000
states	383980
sec	3.3331...
psw	0x81

```

25      temp = 0;
26      P2 = 255;
27      zmienna = 255;
28      Delay();
29      Delay();
30  } else {
31      // Kolejne kroki
32      zmienna /= 2;
33      P2 = zmienna;
34      Delay();
35      if(zmienna == 0)
36          // Jak wszystkie sie zapala, to zerujemy
37          temp = 1;
38  }
39  }
40  else
41  {
42      //Narasta od prawej
43      if(!temp)
44      {
45          // Na poczatku
46          temp = 1;
47          zmienna = 1;
48          P2 = 255;
49          Delay();
50          Delay();
51      } else {
52          // Kolejne kroki
53          zmienna *= 2;
54          P2 = 255 ^ (zmienna-1);
55          Delay();
56          if((zmienna-1) == 255)
57              // Jak wszystkie sie zapala, to zerujemy
58              temp = 0;
59      }
60  }
61  }
62  }

```

Output Window

Load "C:\Documents and Settings\Administrator\Pulpit\SWB\Lab1\Lab1\zad3\proj3"

ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate EXIT FUNC Go

Simulation t1: 3.33315972 sec L:44 C:55 SCRL R/W

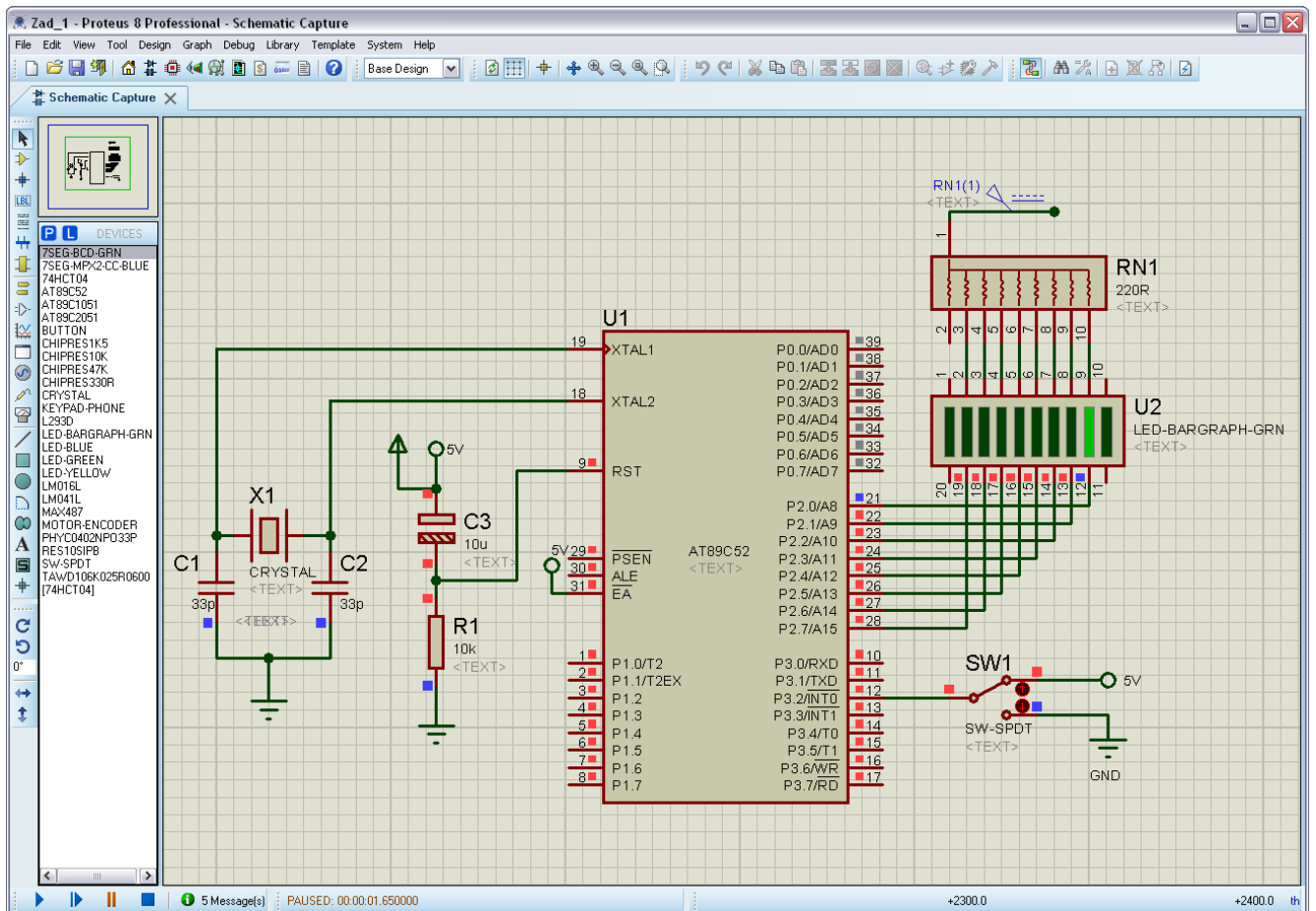
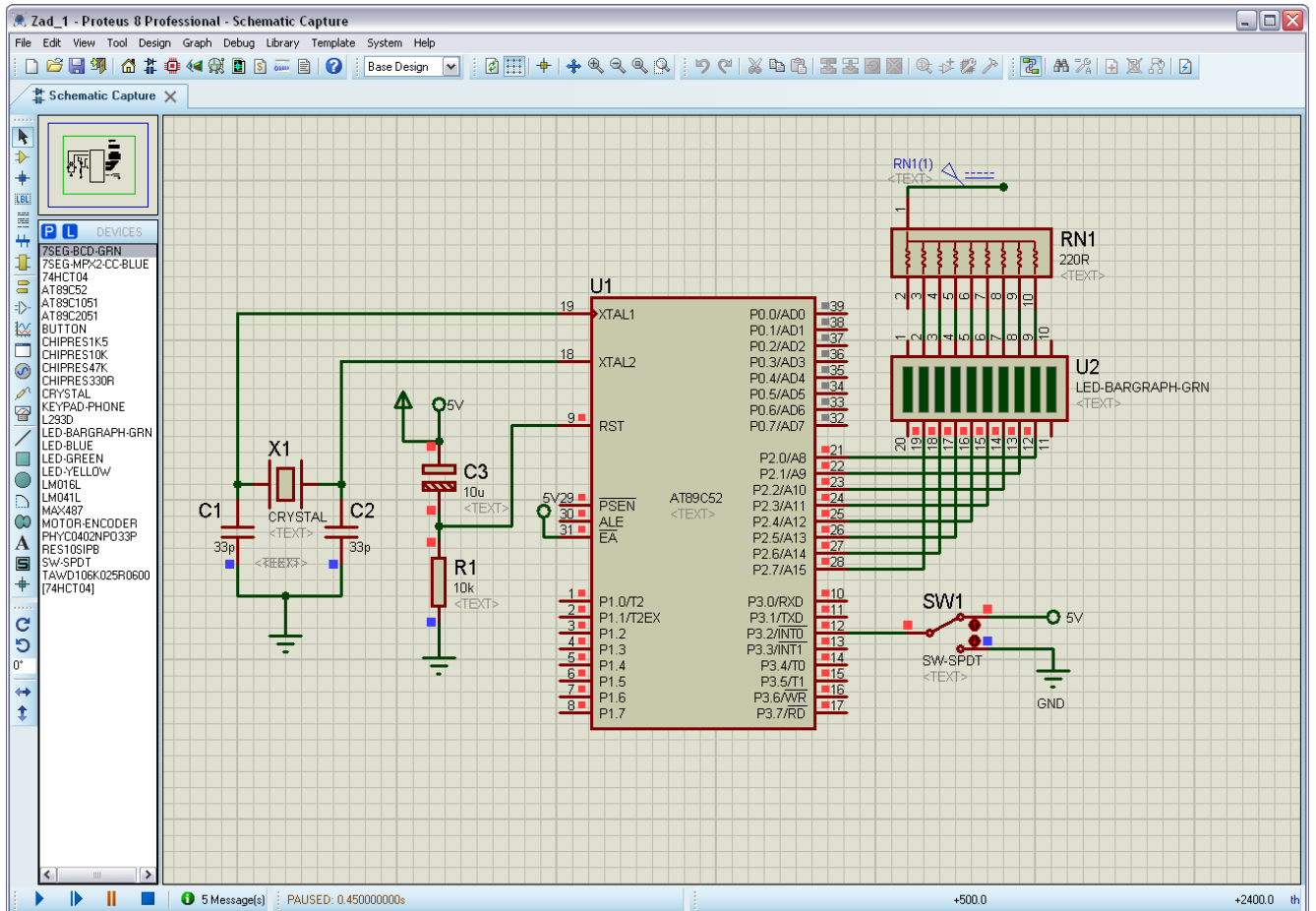
Parallel Port 2

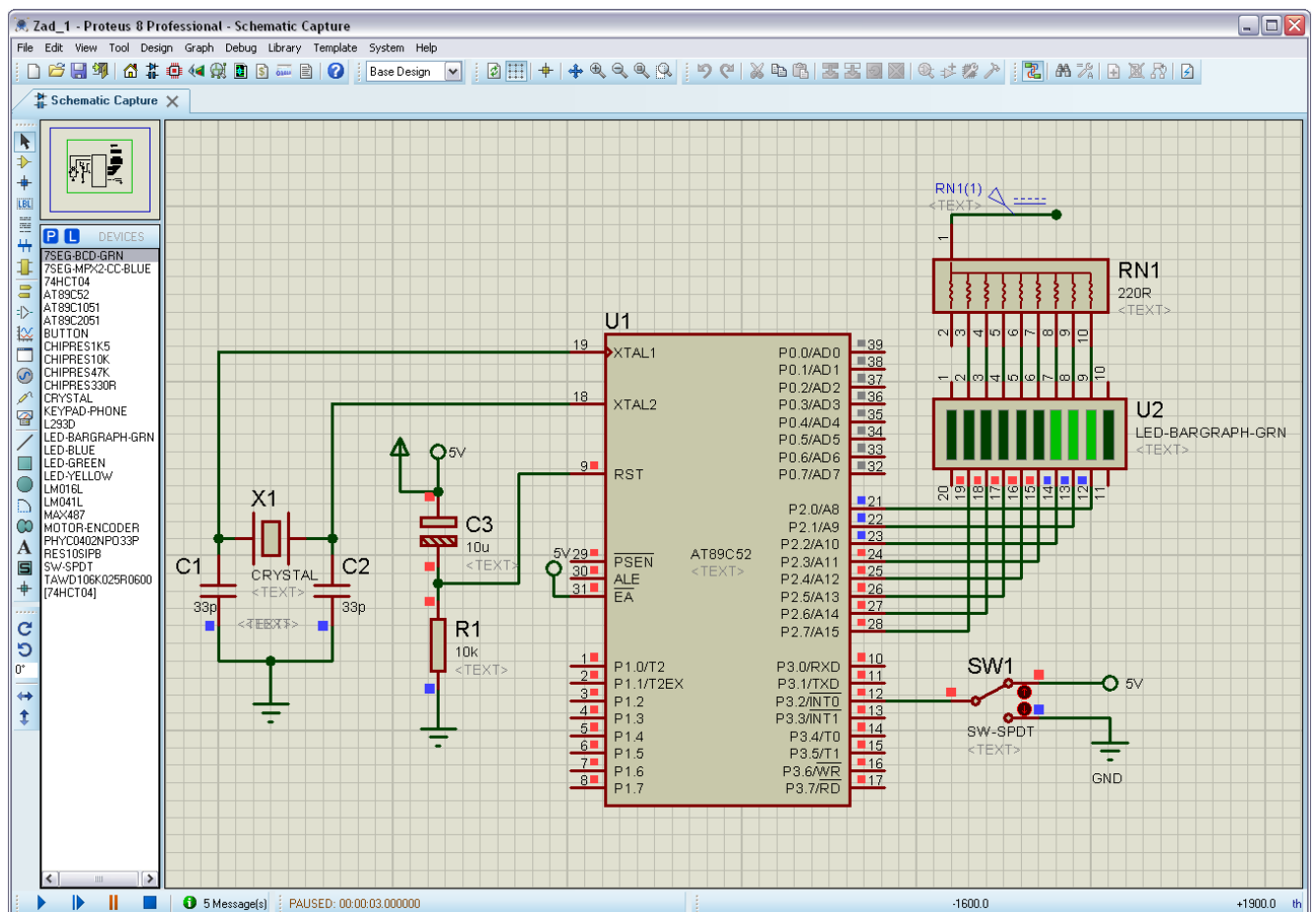
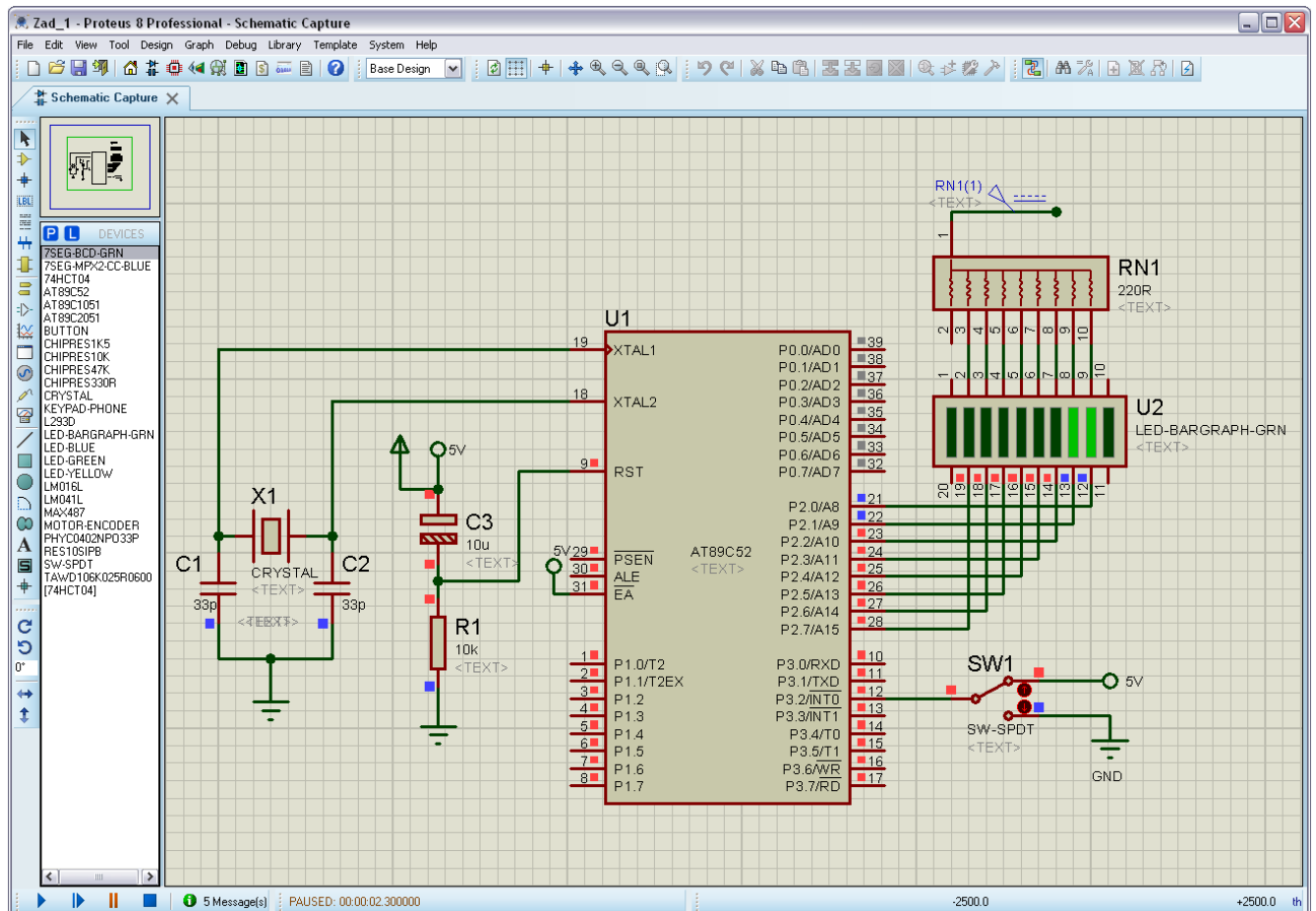
Port 2	7 Bits	0
P2	0x08	0
Pins	0x08	0

Watches

Name	Value
temp	1
zmienna	8

Proteus:





Podsumowanie:

Krok	Keil – wartości zmiennych oraz portu	Proteus – stan diod
------	--------------------------------------	---------------------

1	<div><div>Parallel Port 2</div><div><div>Port 2</div><div>P2: 0xFF 7 Bits 0</div><div>Pins: 0xFF</div></div></div> <table><tr><td>ne</td><td>Value</td></tr><tr><td>temp</td><td>1</td></tr><tr><td>zmienna</td><td>1</td></tr></table>	ne	Value	temp	1	zmienna	1	
ne	Value							
temp	1							
zmienna	1							
2	<div><div>Parallel Port 2</div><div><div>Port 2</div><div>P2: 0xFE 7 Bits 0</div><div>Pins: 0xFE</div></div></div> <table><tr><td>ne</td><td>Value</td></tr><tr><td>temp</td><td>1</td></tr><tr><td>zmienna</td><td>2</td></tr></table>	ne	Value	temp	1	zmienna	2	
ne	Value							
temp	1							
zmienna	2							
3	<div><div>Parallel Port 2</div><div><div>Port 2</div><div>P2: 0xFC 7 Bits 0</div><div>Pins: 0xFC</div></div></div> <table><tr><td>ne</td><td>Value</td></tr><tr><td>temp</td><td>1</td></tr><tr><td>zmienna</td><td>4</td></tr></table>	ne	Value	temp	1	zmienna	4	
ne	Value							
temp	1							
zmienna	4							
4	<div><div>Parallel Port 2</div><div><div>Port 2</div><div>P2: 0xF8 7 Bits 0</div><div>Pins: 0xF8</div></div></div> <table><tr><td>ne</td><td>Value</td></tr><tr><td>temp</td><td>1</td></tr><tr><td>zmienna</td><td>8</td></tr></table>	ne	Value	temp	1	zmienna	8	
ne	Value							
temp	1							
zmienna	8							