***Wojskowa Akademia Techniczna***

***im. Jarosława Dąbrowskiego***

Laboratorium z przedmiotu:

Interfejsy komputerów cyfrowych

Sprawozdanie z ćwiczenia laboratoryjnego nr 3:

**TRANSMISJA RÓWNOLEGŁA**

Prowadzący:

mgr inż. Krzysztof Szajewski

**Wykonał:** Radosław Relidzyński

**Grupa:** WCY20IY4S1

**Data laboratoriów**: 16.05.2021 r.

Spis treści

[A. Treść zadania 2](#_Toc102832284)

[B. Schemat układu (Proteus) 3](#_Toc102832285)

[C. Program mikrokontrolera (Keil) 3](#_Toc102832286)

[D. Analiza i wnioski 5](#_Toc102832287)

[Cykl życia programu: 5](#_Toc102832288)

[Przykładowe działanie programu 5](#_Toc102832289)

[Podsumowanie 7](#_Toc102832290)

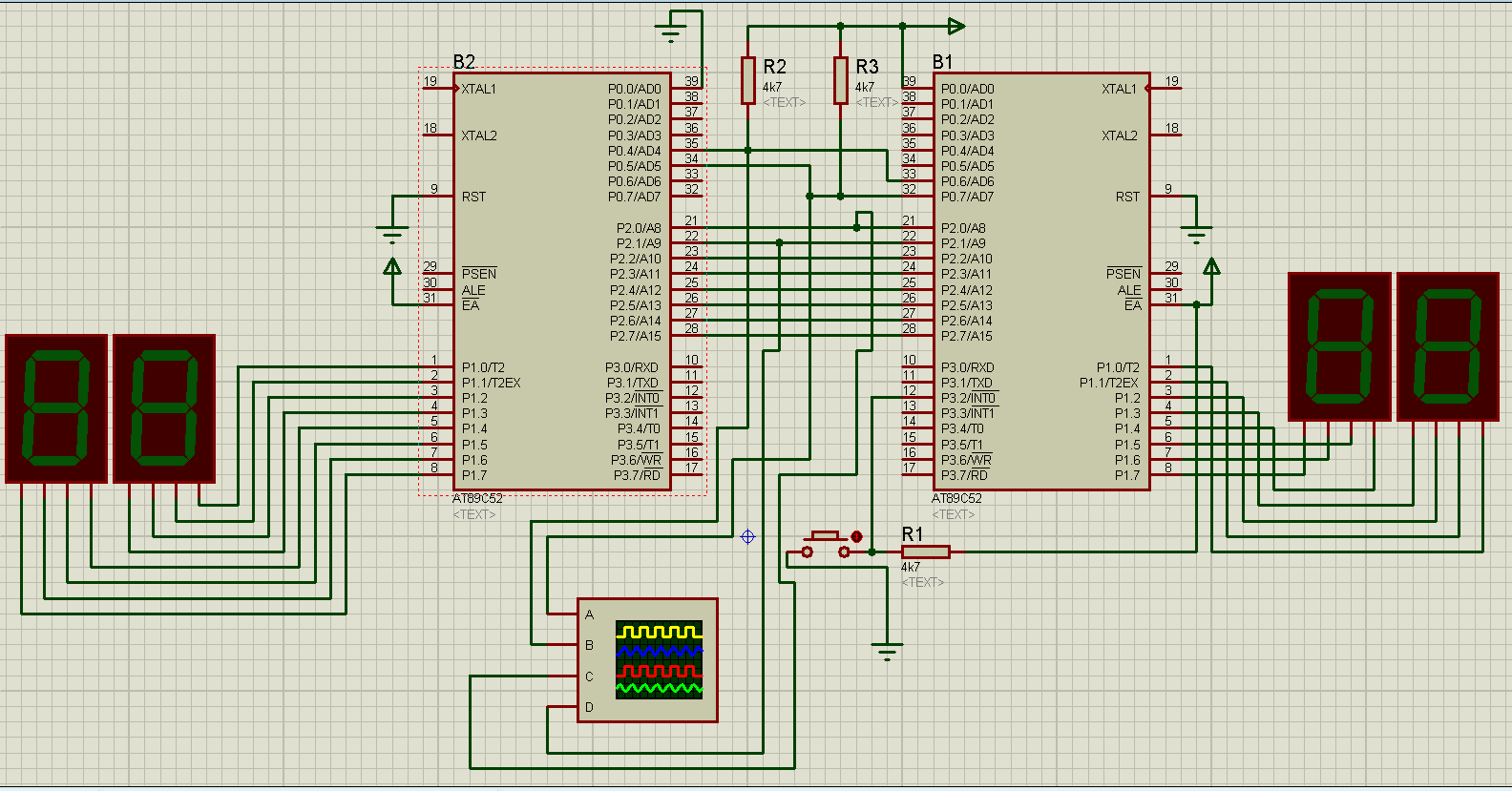
# Treść zadania

Materiały do wejściówki laboratorium 2 - temat 3

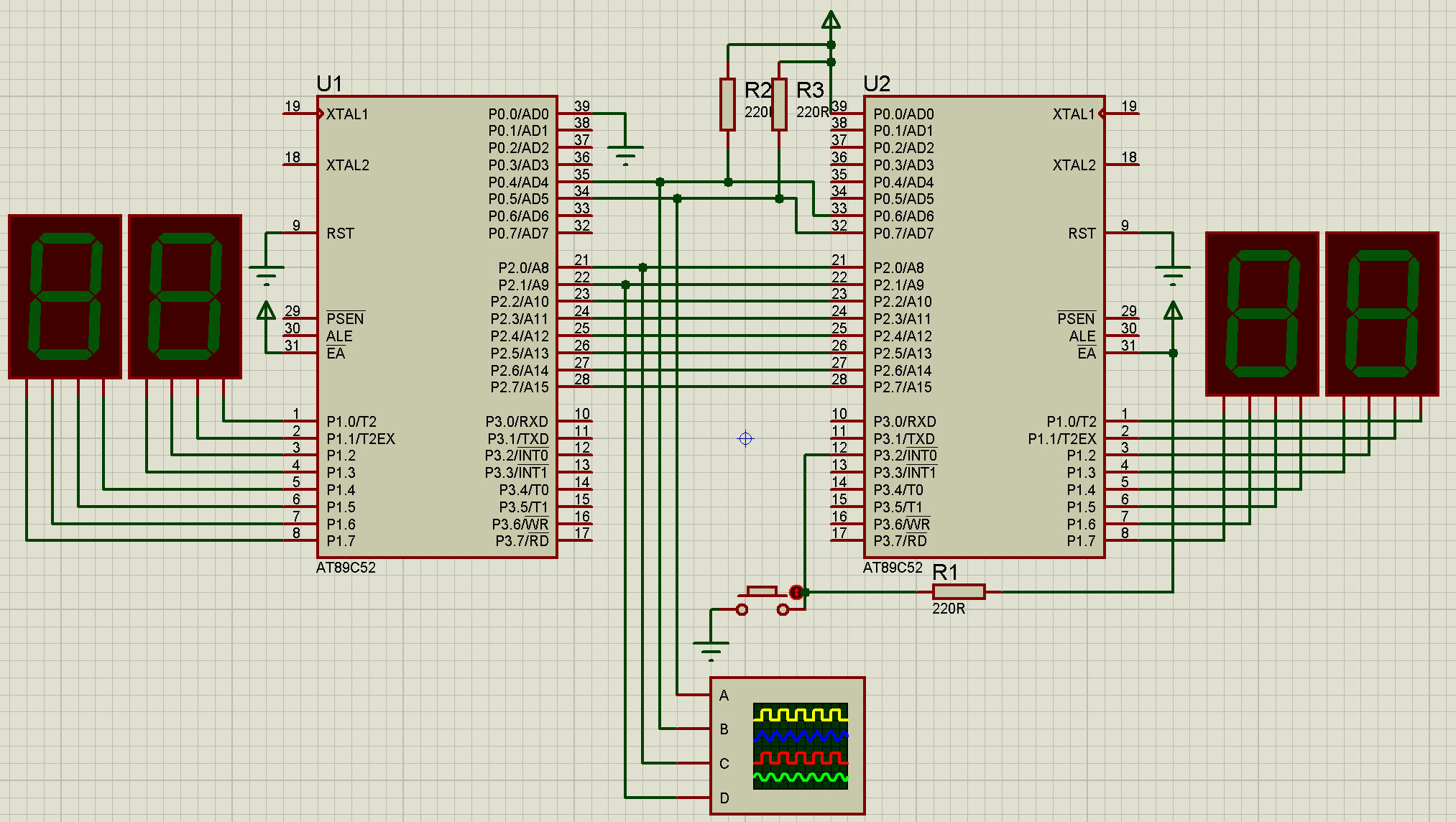
Zbudować w środowisku Proteus stanowisko realizujące transmisję równoległą synchroniczną. Stanowisko powinno zawierać:

* mikrokontroler AT89C52 - pełniący rolę nadajnika (MASTER) - sygnał "1" do pinu P0\_0
* mikrokontroler AT89C52 - pełniący rolę odbiornika (SLAVE) - sygnał "0" do pinu P0\_0
* po dwa siedmiosegmentowe wyświetlacze podlączone do pinów portu P1 nadajnika i odbiornika
* przycisk podłączony do pinu ~INT0 nadajnika
* linia STROBE  pomiędzy pinami P0\_a nadajnika  i P0\_a+2 odbiornika oraz za pośrednictwem rezystora R4k7 do Vcc
* linia ACK pomiędzy pinami P0\_b nadajnika  i P0\_b+2 odbiornika, gdzie a i b wskazuje prowadzący; jeśli nie są wskazane to a = 1, b = 3, oraz za pośrednictwem rezystora R4k7 do Vcc
* linie D0-D7 pomiędzy pinami portu P2

Układ powinien wyglądać jak na rysunku:



# Schemat układu (Proteus)



# Program mikrokontrolera (Keil)

#include <REGX52.H>

volatile unsigned char liczba = 0;

unsigned char bM = 1;

void delay()

{

    unsigned char i, j;

    for(i = 0; i < 255; i++)

    {

        for(j = 0; j < 200; j++);

    }

}

void countInt0() interrupt 0

{

    // Dla mastera, zwieksza wartosc

    EX0 = 0;

    liczba = (liczba + 1) % 100;

    EX0 = 1;

    bM = 0;

}

void main(void)

{

    IT0 = 1; // INT0 aktywne zero

    EX0 = 1; // Wlaczenia INT0

    EA = 1; // Wlaczenie wszstkich przerwan

    if (P0\_0 != 0)

    {

        // MASTER

        P1 = 0x00;

        P2 = 0xFF;

        while(1)

        {

            // Czekaj na wcisniecie

            while(bM);

            // Wypisz nowa liczbe

            P1 = liczba/10 \* 16 + liczba%10;

            // Wyslij sygnal do slave'a o wysylaniu

            P0\_6 = 0;

            // Wyslij liczbe

            P2 = liczba;

            // Czekaj

            delay();

            // Wyslij sygnal do slave'a o koncu wysylania

            P0\_6 = 1;

            // Czekaj na informacje zwrotna od slavea

            while(P0\_7 == 1);

            P2 = 0xFF;

            // Zwalnia przycisk

            bM = 1;

        }

    }

    else

    {

        // SLAVE

        P1 = 0x00;

        while(1)

        {

            // Wypisz liczbe - poczatkowo 0

            P1 = liczba/10 \* 16 + liczba%10;

            // Czekaj na sygnal od mastera

            while(P0\_4 == 1);

            // Wyslij do mastera informacje o odbieraniu sygnalu

            P0\_5 = 0;

            // Wczytaj liczbe od mastera

            liczba = P2;

            // Odczekaj

            delay();

            // Wyslij informacje o koncu odbierania

            P0\_5 = 1;

        }

    }

}

# Analiza i wnioski

### Cykl życia programu:

#### Master:

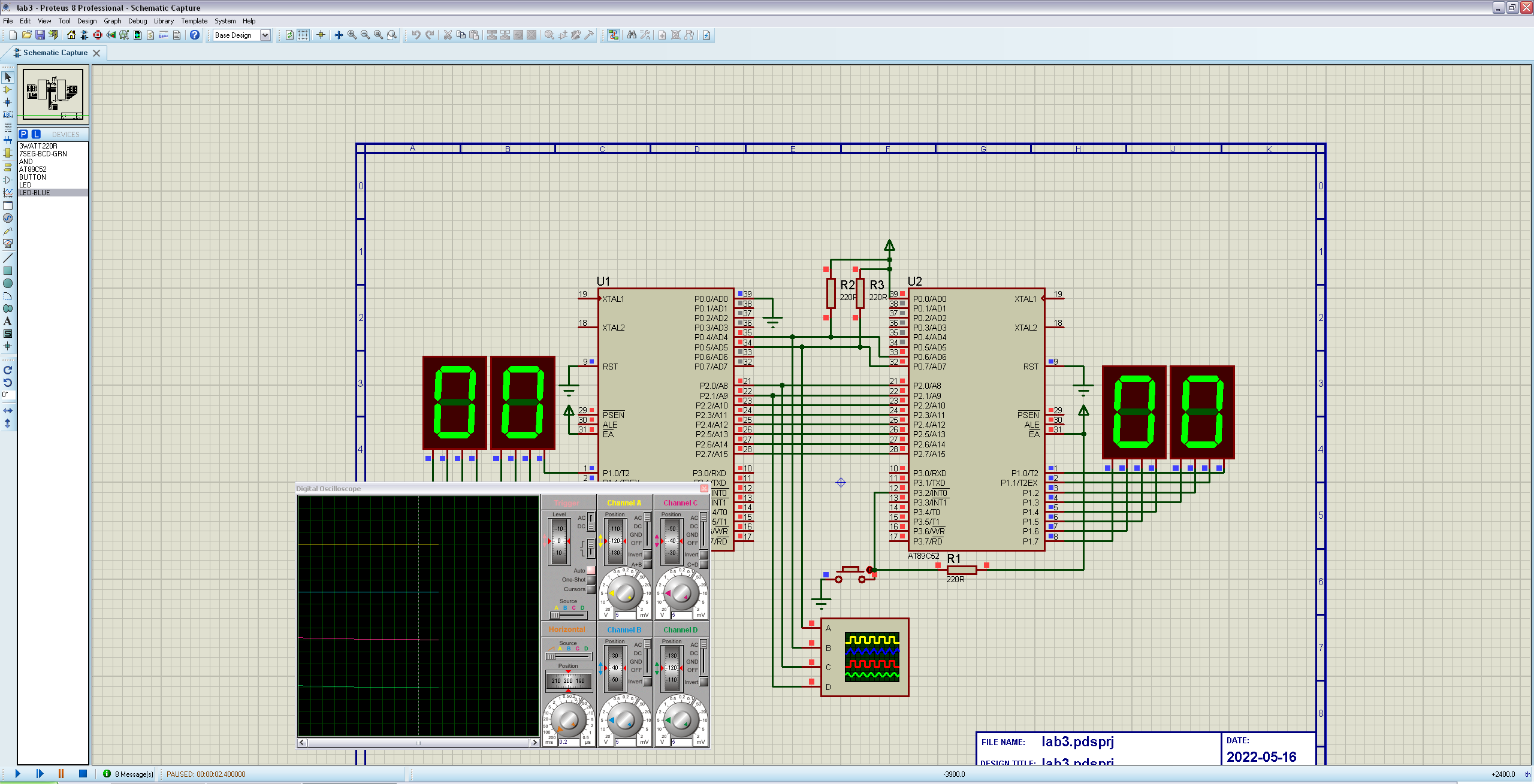
1. Początkowo na porcie P1 ustawia wartość 0, a na porcie P2 wartość 0xFF.
2. Czeka na przerwanie (wciśnięcie przycisku).
3. Występuje przerwanie, zwiększona jest wartość liczby.
4. Wyświetla nową liczbę (na port P1).
5. Wysyła sygnał do slave’a o tranmisji.
6. Wysyła na port P2 liczbę.
7. Wysyła sygnał do slave’a o końcu transmisji.
8. Czeka na informację zwrotną od slave’a.
9. Zmienia wartość portu P2 na 0xFF.
10. Zwalnia przycisk.

#### Slave:

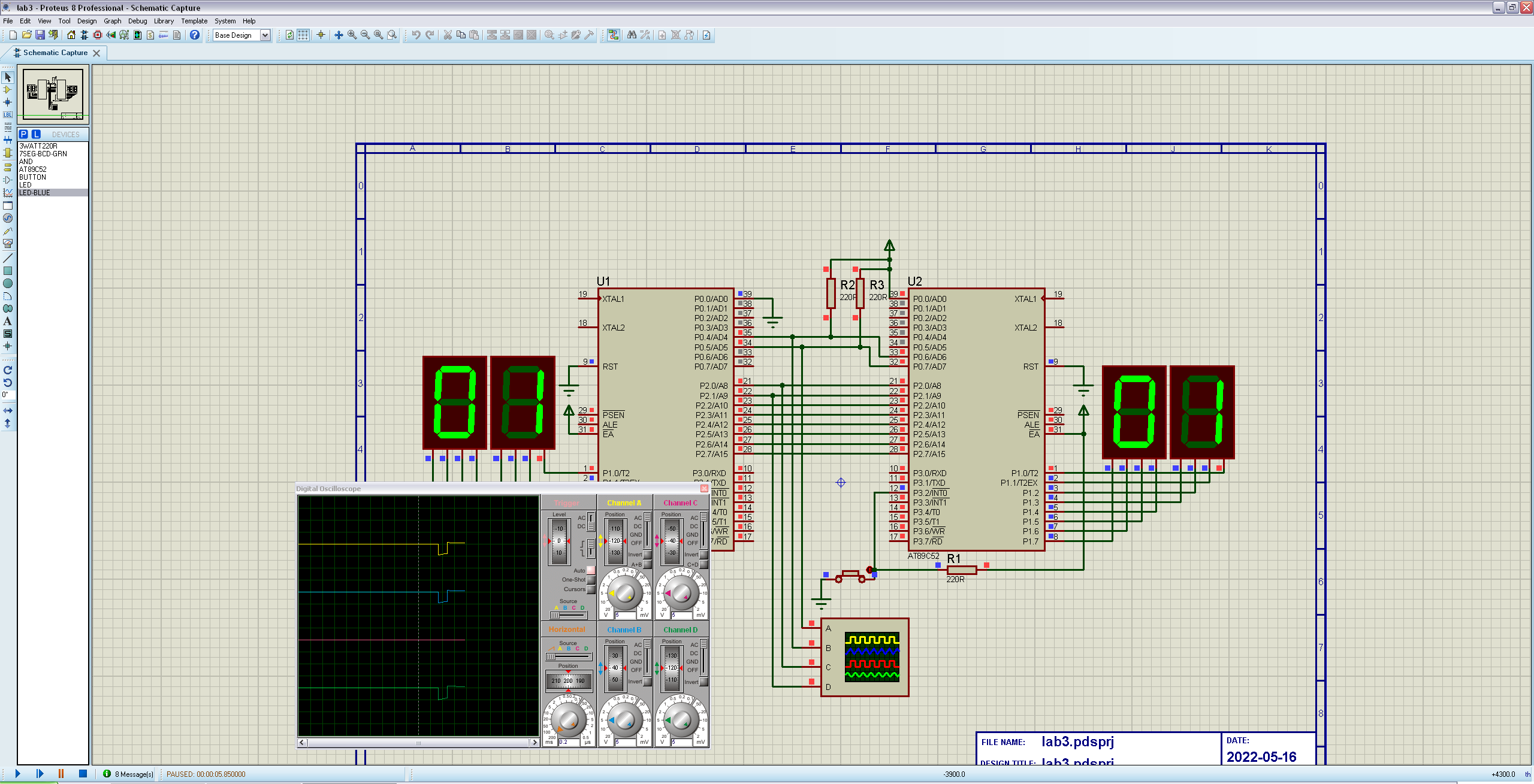
1. Odczytuje liczbę (na początku 0).
2. Czeka na informację od mastera o transmisji.
3. Wysyła informacje zwrotną do mastera o odbieraniu sygnału.
4. Odczytuje z portu P2 liczbe.
5. Odczekuje
6. Wysyła informację do mastera o końcu odbierania.

### Przykładowe działanie programu

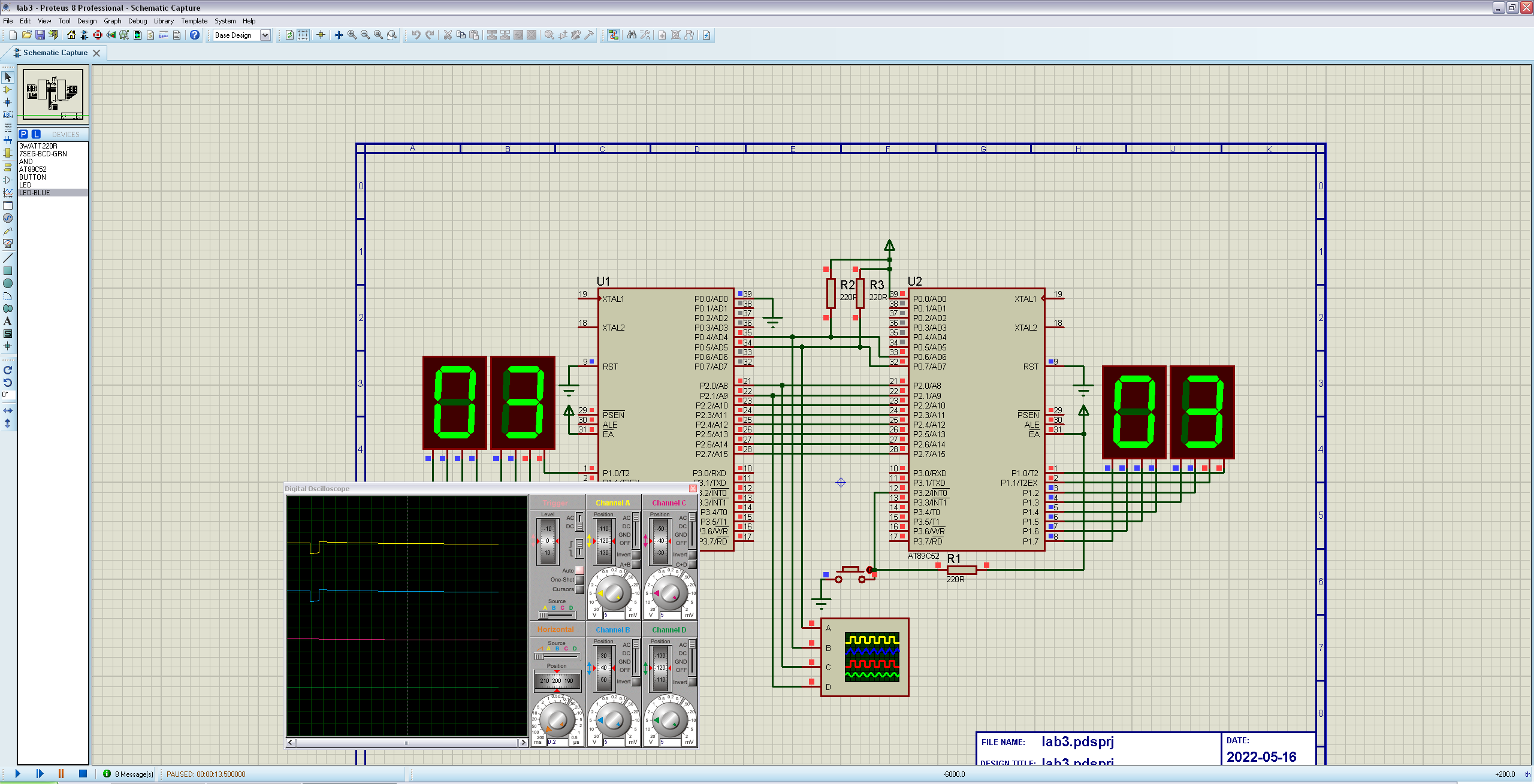
Początek, nic się nie dzieje



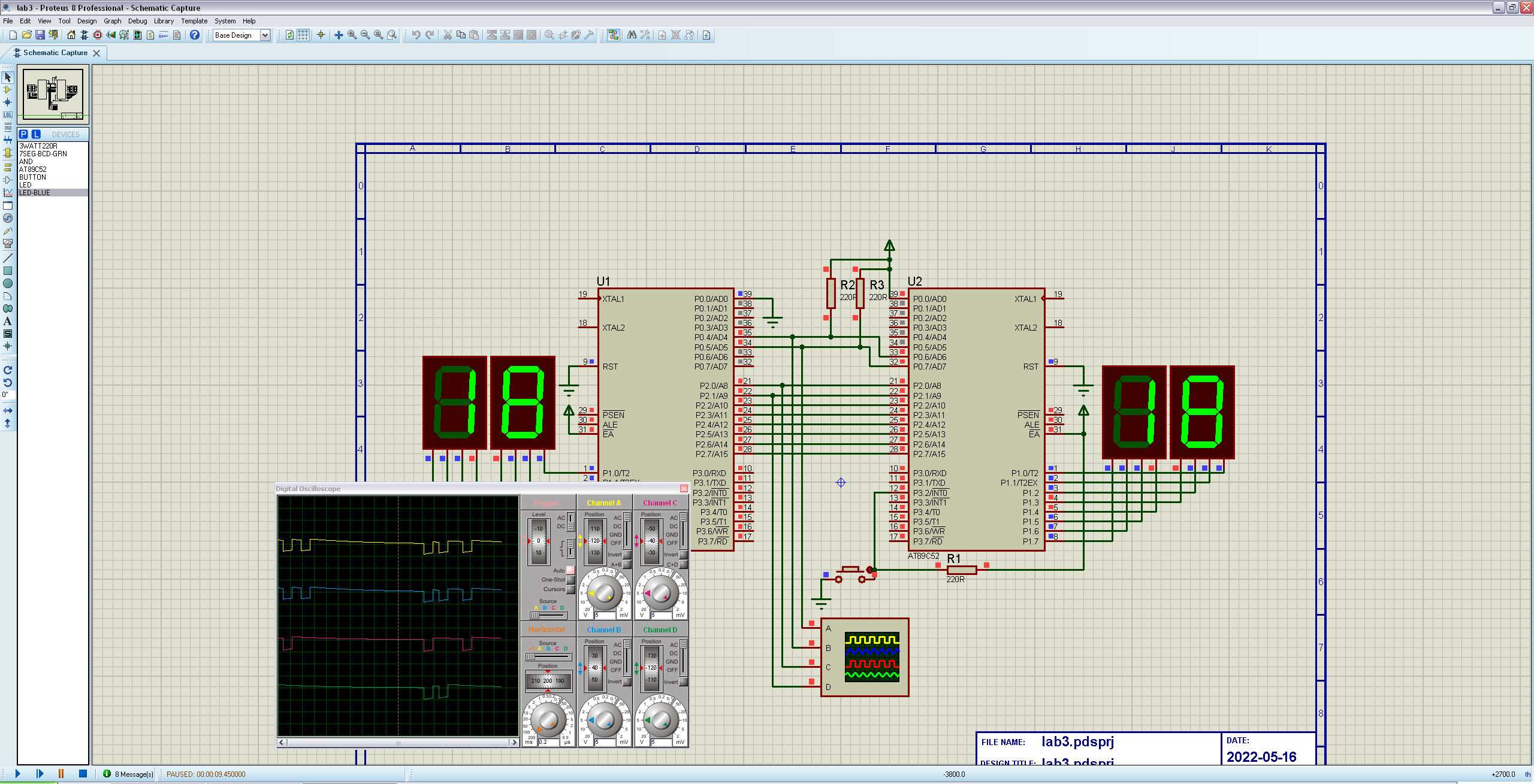
Pierwsze wciśnięcie, sygnały stanowiące początek i koniec nadawania (niebieskie) oraz początek i koniec odbierania (żółte). Zielony i czerwony pokazują wartości dwóch najmniej znaczących bitów, w tym przypadku 01 – wartość 1.



Trzecie wciśnięcie. Zielony i czerwony pokazują wartości dwóch najmniej znaczących bitów, w tym przypadku 11 – wartość 3.



18-te wciśnięcie – Na podstawie poprzednich widać zmiany wartości: 00 -> 01 -> 10. Reszta zdzielenia 18 przez 4 wynosi 2, czyli binarnie dokładnie 10.



### Podsumowanie

W celu stworzenia skutecznej transmisji równoległej należy zapewnić nie tylko połączenie do transportu danych, ale również połączenie do informowania o samym fakcie przesyłania. W tym celu w ramach powyższego układu należało stworzyć transmisję danych (w tym przypadku dla portu P2) oraz 2 linie komunikacyjne (górna w kierunku master->slave, dolna w kierunku slave->master). Oba mikrokontrolery działają na takiej zasadzie, że wysyłają komunikat o rozpoczęciu, a potem o zakończeniu swojego działania. Ilość możliwych wartości do wysłania zależy od ilości połączeń, wyraża się to wzorem 2n, gdzie n to ilość połączeń.

W porównaniu do transmisji szeregowej, jest ona:

1. Bardziej obciążająca sprzętowo ze względu na konieczność zastosowania większej ilości linii tranmisyjnych
2. Mniej skalowalna, ponieważ rozmiar danej jest z góry ograniczony.
3. Łatwiejsza w implementacji, gdyż nie musimy martwić się o problemy z odpowiednim taktowaniem układu.
4. Szybsza, ponieważ dana wysyłana jest na raz w danym odcinku czasu.