POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Elektronika i telekomunikacja (EIT)

SPECJALNOŚĆ: Elektroniczne i komputerowe systemy automatyki (ESA)

PRACA DYPLOMOWA MAGISTERSKA

Automatyzacja i wizualizacja sterowania poziomem cieczy

Automation and visualisation of liquid level contol

AUTOR:
Karol Kozłowski

PROWADZĄCY PRACĘ:
dr inż. Antoni Izworski

OCENA PRACY:

WROCŁAW 2008

Spis treści

Rozdział 1

Wstęp

1.1 Wprowadzenie

Jednym z najczęściej wykonywanych pomiarów w automatyce przemysłowej są pomiary poziomu w szczególności cieczy i materiałów sypkich. Znajdują one zastosowanie w zakładach energetycznych, elektrociepłowniczych, chemicznych, spożywczych itp. Pomiaru poziomu można dokonywać na wiele różnych sposobów, wybrane z nich zostały praedstawione w rozdziale 2. Regulacji dokonuje się za pomocą autonomicznych regulatorów lub też z wykożystaniem sterowników PLC.

1.2 Cel pracy

Celem pracy jest opracowanie układu mikroprocesorowego do pomiaru poziomu cieczy. Urządzenie będzie obsługiwać wejście analogowe w standardzie prądowym $4-20mA$ oraz wyjście w standardzie częstotliwościowym $1000-5000Hz$. Układ zrealizowany zostanie na mikrokontrolerze z przetwornikami analogowo-cyfrowymi. W pracy opisane zostaną róznież najpopularniesze sposoby pomiaru poziomu cieczy oraz najczęściej spotykane algorytmy regulacji jego poziomu. Opracowny zostanie również projekt graficzny i informatyczny stacji operatorskiej współpracującej ze sterownikiem S7-200 firmy Siemens, na którym zaimplementowane zostaną algorytmy regulacji 2-stawnej i 3-stawnej. Program stacji operatorskiej zostanie przygotowany w środowisku wizualizacyjnym Wonderware InTouch.

Ponadto w pracy zostaną opisane zagadnienia dotyczące sposobów identyfikacji obiektów oraz zasady doboru nastaw regulatorów ciągłych.

1.3 Zakres pracy

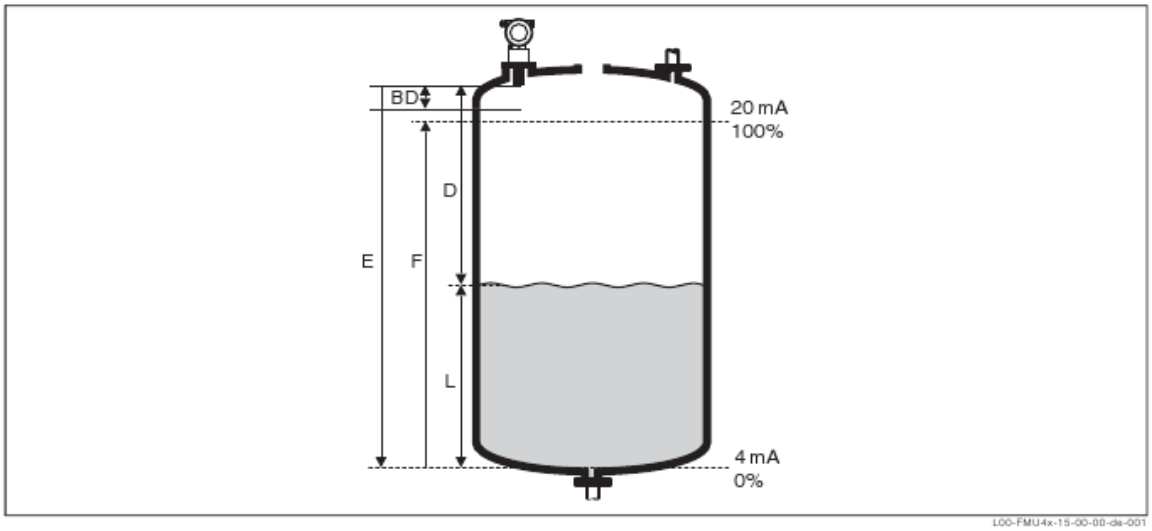
Zakres pracy obejmuje opis podstawowych metod pomiaru poziomu cieczy, takich jak ultradźwiękowa, elektroniczna, pojemnościowa, przewodnościowa, różnicy ciśnień, hydrostatyczna czy radiometryczna. Uwzględniono również przegląd najbardziej popularnych algorytmów sterowania poziomem jak np. algorytmy sterowania dyskretnego (dwustawne i trójstawne) oraz algorytmy regulacji ciągłej - PID. Dla tych ostatnich podano metody wyznaczania parametrów obiektu (identyfikacja) oraz zasady doboru nastaw regulatorów ciągłych. W dalszej częsci pracy przedstawiono opis budowy mikrokontrolerów ATmega, gdzie uwzględniono takie elementy architektury jak porty wejścia/wyjścia, przerwania, układy licznikowe oraz sposoby programowania (flashowania). Praca zawiera również

szczegółowy opis programowania (kodowania) mikrokontrolerów AVR ATmega szczegółowo opisujący takie aspekty jak instalacja niezbędnego oprogramowania, tworzenie i konfiguracja projektu w środowisku AVRStudio, kodowanie, kompilacja oraz symulacja napisanego programu. Kolejna część pracy opisuje budowę wykonanego przetwornika pomiarowego a w szczególności jego architekturę, budowę obwodów zasilania, wejściowcyh i wyjściowych. Następnie przedstawiony jest sposób realizacji układu, gdzie zawarte są schematy elektyczne i wzory płytek obwodów drukowanych. Podane są również wyznaczone laboratoryjnie charakterystyki przetwarzania wejścia i wyjścia układu. Zamieszczono również rozległy opis implementacji algorytmów kontroli w sterowniku PLC a konkretnie Siemens S7-200, uwzględnia on także instalację oprogramowania i podstawową konfigurację sprzętową. W tej części opisane zostały implementacje algorytmów sterowania dwu- i trójstawnego ze szczegółowym opisem konfiguracji szybkich liczników. Na końcu przedstawiony został projekt stacji operatorskiej wykonanej w środowisku Wonderware InTouch 9.5 uwględniający zarówno instalację oprogramowania wizualizacji, jak i serwera wymiany danych ze sterownikiem (TopServer). Znajduje się tam również szczegółowy opis tworzenia podstawowych elementów wizualizacji - jak wskaźniki czy suwaki, łącznie z przypisaniem ich do zmiennych zewnętrznych (sterownika).

Do niniejszej pracy załączona zostanie płyta CD z wersją elektroniczną pracy, omawianym oprogramowaniem oraz wykonanymi aplikacjami (wizualizacją, programem steronika, programem mikrokontrolera.

Rozdział 2

Pomiar poziomu cieczy



E: Wysokość zbiornika; F: Pełny zakres pomiaru; D: Odległość od membrany czujnika do poziomu produktu; L: Poziom produktu; BD: Strefa martwa

Rysunek 2.1 Pomiar poziomu cieczy [1]

Pomiar poziomu (zarówno cieczy jak i materiałów sypkich) najczęściej sprowadza się do pomiaru długości lub odległości. W zależności od mierzonego medium zakresy pomiarowe mogą wynosić od kilku centymetrów (np. w samochodowych zbiornikach paliwa) do kilkudziesięciu metrów (np. w silosach zbożowych). W przemyśle najczęściej dokonuje się pomiarów poziomu cieczy, past, szlamów, gazów ciekłych, drobnoziarnistych materiałów sypkich i kruszyw, zgromadzonych w zbiornikach stacjonarnych i przenośnych, magazynowych i buforowych. Pomiary tego typu wykorzystywane są praktycznie w każdej gałęzi przemysłu, dlatego też środowisko w jakim są one wykonywane jest bardzo zróżnicowane. Pomiarów dokonuje się przy temperaturach zarówno bardzo niskich (bliskich zeru bezwzględnemu) jak i wysokich (kilkaset stopni Celsjusza). Ciśnienie panujące w zbiornikach jest również bardzo zróżnicowane.

Istnieje wiele różnych metod pomiaru, które są uzależnione najczęściej od charakteru mierzonego medium. Najczęściej stosowanymi metodami są ultradźwiękowa, elektromechaniczna, pojemnościowa, przewodnościowa, różnicy ciśnień, hydrostatyczna radiometryczna, optyczna i inne.

2.1 Metoda ultradźwiękowa

Rysunek 2.2 Pomiar ultradźwiękowy [1]

Jest to jedna z częściej spotykanych metod pomiaru. Nadajnik czujnika emituje krótkie impulsu ultradźwiękowe, które po odbiciu od powierzchni produktu wracają do anteny, pracującej jednocześnie jako odbiornik. Zasada działania urządzenia opiera się na pomiarze czasu przelotu t fali akustycznej pomiędzy czujnikiem a powierzchnią medium. Czujnik wykorzystuje zmierzony czas t oraz informację o prędkości dźwięku c do wyliczenia odległości D zgodnie z zależnością:

$$D = c \cdot \frac{t}{2} \tag{2.1}$$

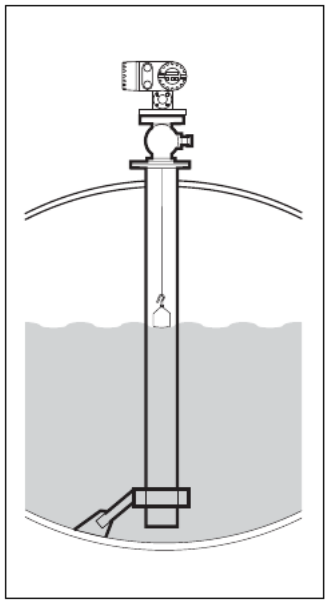
gdzie c - prędkość dźwięku

Informacja o wysokości zbiornika E pozwala na wyliczenie poziomu L z równania 2.2:

$$L = E - D \tag{2.2}$$

Czujniki tego typu często posiadają wbudowany czujnik temperatury służący do kompensacji zmian prędkości rozchodzenia się fali ultradźwiękowej przy zmianach temperatury.

2.2 Metoda elektromechaniczna

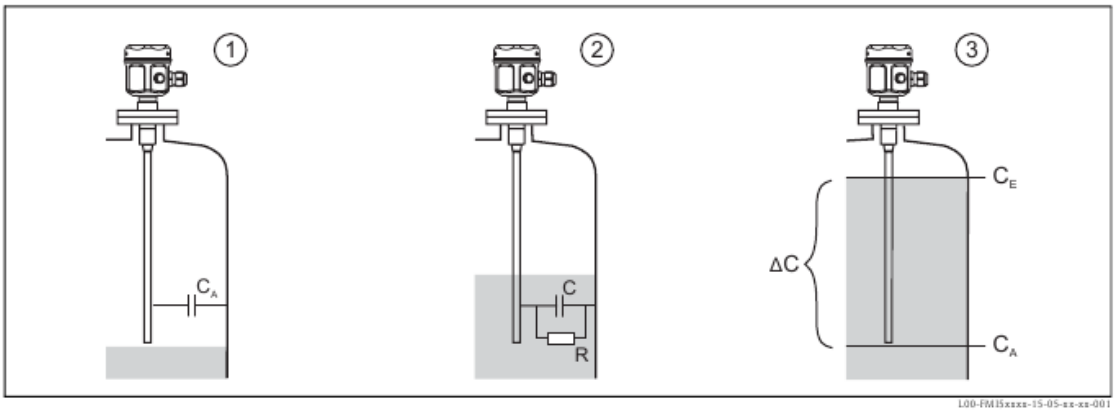


Rysunek 2.3 Pomiar elektromechaniczny [1]

Zasada działania pływakowego przetwornika poziomu oparta jest na pomiarze przemieszczenia czujnika wypieranego przez ciecz. Mały czujnik pływakowy jest dokładnie pozycjonowany w cieczy za pomocą serwomechanizmu elektromagnetycznego. Jest on zawieszony na lince pomiarowej, nawiniętej na precyzyjnie rowkowany bęben, umieszczony w obudowie. Bęben jest sterowany przez serwomechanizm ze sprzężeniem magnetycznym, całkowicie odizolowany od obudowy bębna. Magnesy zewnętrzne znajdują się na bębnie, natomiast magnesy wewnętrzne - na elektrycznym silniku napędowym. Obrót magnesów wewnętrznych, na skutek przyciągania magnetycznego, powoduje również obracanie się magnesów zewnętrznych, a tym samym ruch obrotowy całego zespołu bębna. Masa pływaka zawieszonego na lince wytwarza moment obrotowy oddziaływający na magnesy zewnętrzne, powodując tym samym zmianę strumienia magnetycznego. Zmiany strumienia, generowane pomiędzy elementami zespołu bębna, wykrywane są przez przetwornik elektromagnetyczny o unikatowej konstrukcji, znajdujący się na magnesie wewnętrznym. Utrzymanie sygnału sterującego, generowanego przez zmiany strumienia magnetycznego, na wymaganym poziomie, tj. zgodnym z wprowadzoną nastawą jest zapewnione przez właściwe kontrolowanie silnika napędowego.

W czasie opuszczania czujnika pływakowego, po zetknięciu z cieczą jego masa ulega zmniejszeniu ze względu na siłę wyporu cieczy. W efekcie, moment obrotowy mechanizmu ze sprzężeniem magnetycznym ulega zmianie, która mierzona jest przez 5 par czujników Halla z kompensacją temperaturową. Sygnał odzwierciedlający pozycję pływaka jest przesyłany do układu sterowania silnikiem. Podczas, gdy poziom cieczy podnosi się i opada, pozycja pływaka jest regulowana linką przez silnik napędowy. Poziom cieczy jest wyznaczany z dokładnością ok $1mm$, na podstawie dokładnej analizy ruchu obrotowego bębna z nawiniętą linką pomiarową.

2.3 Metoda pojemnościowa



R: Przewodność cieczy
C: Pojemność cieczy
C_A: Pojemność początkowa (sonda odkryta)
C_E: Pojemność końcowa (sonda zakryta): Zmiana pojemności
ΔC: Zmiana pojemności

Rysunek 2.4 Pomiar pojemnościowy [1]

Zasada pomiaru metodą pojemnościową bazuje na wyznaczaniu zmiany pojemności kondensatora, którego jedną elektrodę stanowi sonda, natomiast drugą ściana zbiornika lub rura osłonowa (materiał przewodzący). Zmiana poziomu cieczy powoduje zmianę pojemności utworzonego w ten sposób kondensatora. Podczas, gdy sonda znajduje się w powietrzu, mierzona jest określona pojemność początkowa. Po napełnieniu zbiornika cieczą, pojemność kondensatora wzrasta proporcjonalnie do stopnia zakrycia sondy.

Dla cieczy o przewodności większej od $100\frac{\mu s}{cm}$, pomiar jest niezależny od wartości stałej dielektrycznej (DK) medium. W efekcie, zmiany wartości DK nie wpływają na wskazanie wartości mierzonej. Ponadto, w przypadku sond z częścią nieaktywną, konstrukcja przyrządu pozwala wyeliminować wpływ osadu i kondensacji przy przyłączu technologicznym.

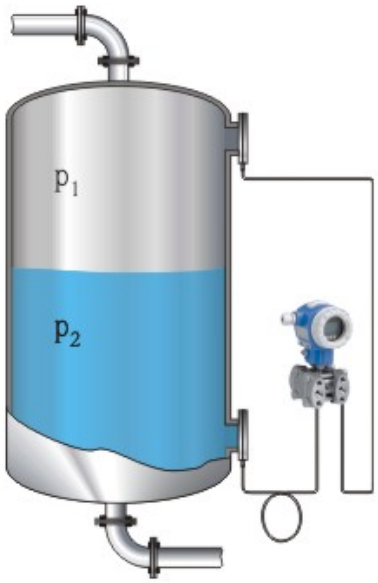
2.4 Metoda przewodnościowa

Rysunek 2.5 Pomiar przewodnościowy [1]

W pustym zbiorniku między czujnikami prętowymi występuje różnica potencjałów. Po napełnieniu zbiornika, między elektrodą odniesienia, i na przykład, czujnikiem prętowym maksymalnego poziomu, popłynie prąd i nastąpi przełączenie wyjścia sygnalizatora. Natychmiast po odkryciu przez ciecz tego czujnika następuje przełączenie do stanu początkowego. W przypadku regulacji dwustanowej, przełączenie urządzenia do stanu początkowego następuje dopiero po odsłonięciu obu czujników - stanu maksymalnego i minimalnego.

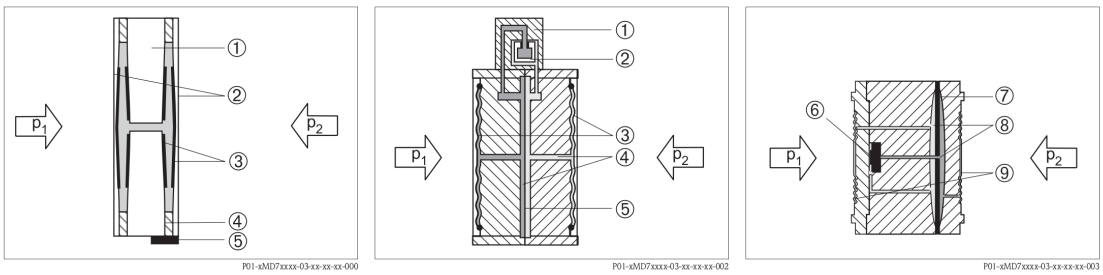
Stosowanie napięcia przemiennego zapobiega korozji czujników prętowych lub ich zniszczeniu na skutek elektrolizy. Materiał, z którego wykonane są ścianki zbiornika nie jest istotny, ponieważ system został zaprojektowany w postaci zamkniętego obwodu bezpotencjałowego obejmującego czujniki prętowe i moduł elektroniki. Nie ma niebezpieczeństwa, jeśli w czasie pracy czujniki prętowe zostaną dotknięte.

2.5 Metoda różnicy ciśnień



Rysunek 2.6 Pomiar różnicy ciśnień [1]

Ciśnienie procesowe działające na membranę czujnika, powoduje jej ugięcie. Zmiana odległości między bardzo precyzyjnie naniesionymi złotymi elektrodami, powoduje zmianę pojemności po obydwu stronach. Na tej podstawie wyznaczany jest stosunek pomiędzy nadciśnieniem a ciśnieniem hydrostatycznym (aby pomiar był dokładny, nie powinien być większy niż 4 : 1). Metoda ta ma przewagę nad metodą hydrostatyczną gdyż na wynik pomiaru nie wpływa nadciśnienie w zbiorniku. Jest to główny powód, dla którego metodę tę stosuje się w zbiornikach ciśnieniowych.



Czujnik ceramiczny

1 Podłoże ceramiczne
2 Membrana
3 Elektrody
4 Szkło frytowe utrzymujące membranę na podłożu ceramicznym
5 Czujnik temperatury

Czujnik krzemowy10 mbar i 30 mbar

1 Element pomiarowy
2 Membrana silikonowa
3 Membrana separująca
4 Olej wypełniający
5 Wbudowane zabezpieczenie przeciwprzeciążeniowe

Czujnik krzemowy od 100 mbar

6 Element pomiarowy
7 Membrana przeciążeniowa / membrana środkowa
8 Olej wypełniający
9 Membrana separująca

Rysunek 2.7 Spotykane konstrukcje przetworników ciśnień [1]

2.6 Metoda hydrostatyczna

Rysunek 2.8 Pomiar hydrostatyczny [1]

Ciężar słupa cieczy powoduje powstanie ciśnienia hydrostatycznego działającego na membranę czujnika. Zakładając stałą gęstość produktu ciśnienie hydrostatyczne jest liniową funkcją wysokości słupa cieczy:

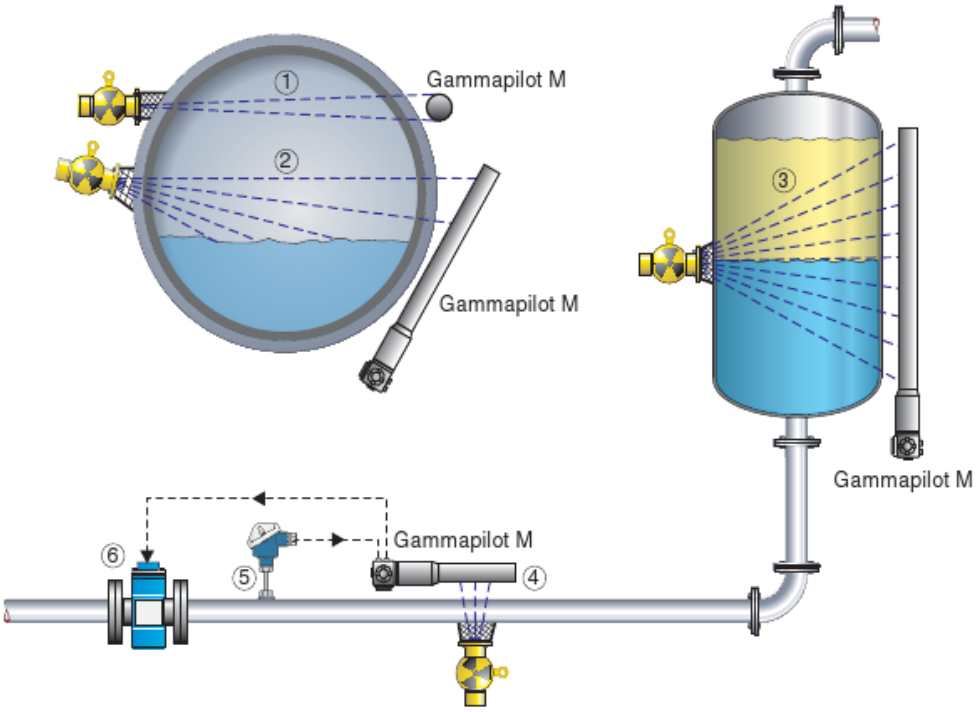
$$p_{hydrostat} = \varrho \cdot g \cdot h \tag{2.3}$$

ϱ = gęstość
g = stała grawitacji ($9.81 \frac{m}{s^2}$)
h = odległość pomiędzy powierzchnią cieczy, a środkiem membrany czujnika

Zaletami tej metody pomiaru są:

- prosta konstrukcja układu pomiarowego
- wewnętrzne elementy zbiornika, zapienienie i turbulencje powierzchni cieczy nie zakłócają pomiaru

2.7 Metoda radiometryczna



Rysunek 2.9 Pomiar radiometryczny [1]

Ta metoda pomiaru stosowana jest jedynie w ekstremalnych warunkach procesowych bądź tam, gdzie zastosowanie innych metod pomiaru jest niemożliwe.

Zasada pomiaru tą metodą polega na wykrywaniu absorpcji promieniowania γ przechodzącego przez mierzone medium. Źródło promieniowania jonizującego (w postaci izotopu cezu lub kobaltu) umieszcza się na ścianie zbiornika, fale przechodząc przez medium ulegają osłabieniu na skutek absorpcji. Zamontowany po przeciwległej stronie zbiornika detektor wykrywa wiązkę promieniowania wykorzystując efekt fotoluminescencji. Przykładowe techniki pomiaru tą metodą z wykorzystaniem czujnika Gammapilot M firmy Endress+Hauser znajdują się na rysunku 2.9.

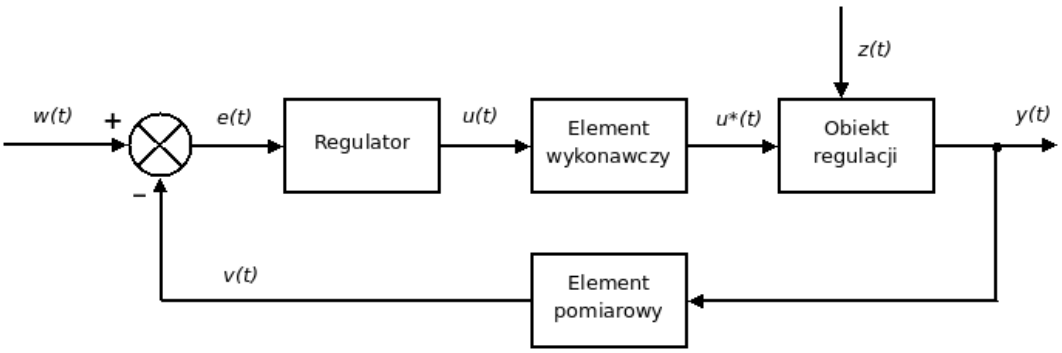
Jak podaje producent, czujniki tego typu można również wykorzystać do następujących celów[1]:

- pomiar poziomu, również w trybie kaskadowym (konfiguracja szeregowa przetworników) lub zwielokrotniania czułości (konfiguracja równoległa przetworników)
- sygnalizację poziomu (monitorowanie poziomu minimalnego lub maksymalnego)
- detekcja rozdziału faz
- pomiar gęstości
- pomiar koncentracji
- pomiar przepływu masowego z wykorzystaniem przepływomierza objętościowego

[1] Więcej informacji na stronie producenta: `http://www.pl.endress.com/gammapilot_m`

Rozdział 3

Regulacja poziomu cieczy



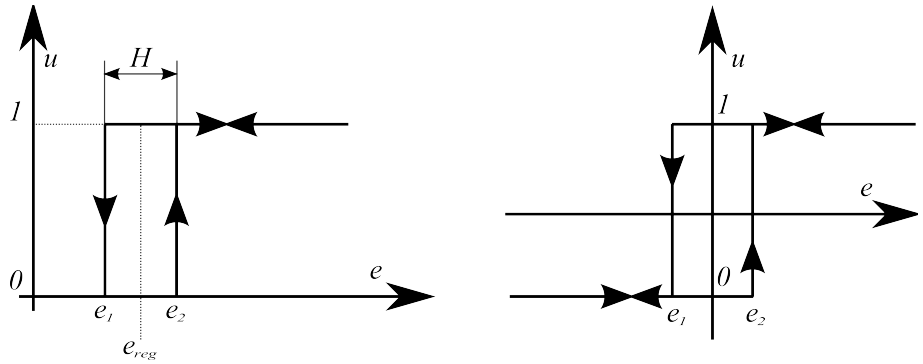
Rysunek 3.1 Układ Automatycznej Regulacji [3]

W automatyce spotykamy się z wieloma różnymi metodami regulacji wartości. Najczęściej stosowanymi sposobami są regulacja dyskretna (dwustawna bądź trójstawna) oraz ciągła regulacja PID[1]. Sygnały regulacji powstają w regulatorze w wyniku określenia uchybu regulacji, a następnie odpowiedniego uformowania wartości wyjściowej. W regulatorach o działaniu ciągłym wielkość wyjściowa jest funkcją ciągłą wielkości wejściowej. W regulatorach dyskretnych wartość wyjściowa przyjmuje jeden z n stanów, zależnie od przedziału, w którym znajduje się wartość wejściowa. Znane są również połączenia tych dwóch typów regulatorów, np.: regulator dwustawny z elementem inercyjnym drugiego rzędu w roli sprzężenia korekcyjnego [6] - regulator ten ma podobne właściwości do regulatora PD. Zastosowanie takiego sprężenia nie wpływa na wartość błędu w stanie ustalonym, jednak znacząco wpływa na jakość regulacji. Dobór typu regulatora zależy zwykle od jego zastosowania. Regulatory dyskretne stosuje się tam, gdzie wartość regulowana nie musi mieć wysokiej dokładności (np.: regulacja temperatury w pokoju), bądź tam gdzie wydajność urządzeń wykonawczych nie może być płynnie regulowana.

Przebiegi czasowe regulatorów omówionych w tym rozdziale pochodzą ze środowiska MATLAB/Simulink. Wszystkie pliki symulacyjne znajdują się na załączonej płycie CD w katalogu $\backslash projekty \backslash MATLAB \backslash$

[1] ang. Proportional-Integral-Derivative controller - regulator proporcjonalno-całkująco-różniczkujący

3.1 Sterowanie dwustawne



Rysunek 3.2 Charakterystyki statyczne (u - sygnał wyjściowy, e - sygnał wejściowy regulatora, $H = e_2 - e_1$ - strefa histerezy) [6]

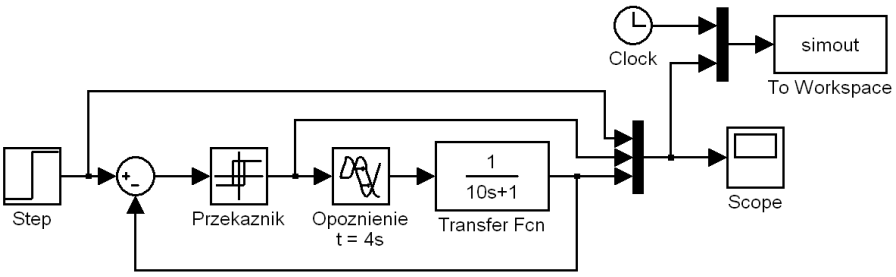
Regulatory posiadające nieliniową i nieciągłą charakterystykę przedstawioną na rysunku 3.2 nazywane są regulatorami dwustawnymi lub dwupołożeniowymi. Sygnał wyjściowy takich urządzeń może przyjmować tylko dwa stany logiczne - stan logicznego zera lub stan logicznej jedynki. Wprowadzenie takiego elementu do układu automatycznej regulacji powoduje, że cały układ staje się nieliniowy. Niesie to za sobą następujące konsekwencje:

- w układach nieliniowych nie występuje zasada superpozycji,
- charakter odpowiedzi zależy od wartości amplitudy wymuszenia, a częstotliwość sygnału odpowiedzi nie jest równa częstotliwości wymuszenia (mogą wystąpić składowe sygnału o częstotliwościach wyższych albo niższych niż częstotliwość sygnału wymuszającego),
- stabilność takiego układu zależy od warunków początkowych.

Do elementów dwustawnych zaliczamy przekaźniki (elektroniczne bądź elektromechaniczne), styczniki, tyrystory itp. Schemat blokowy układu regulacji dwupołożeniowej jest przedstawiony na rysunku 3.3. Przyjęto w nim model obiektu w postaci członu inercyjnego pierwszego rzędu z opóźnieniem (odpowiada to obiektom rzeczywistym o wysokim rzędzie inercji). Regulacja dwustawna stosowana jest do obiektów inercyjnych lub całkujących (z opóźnieniem) mających duże stałe czasowe (regulacja ciśnienia, poziomu cieczy czy temperatury). W zależności od zastosowania możemy wyróżnić różne typy regulatorów dwustawnych:

- *termostaty* - służące do regulacji temperatury,
- *presostaty* - służące do regulacji ciśnienia,
- *mobreye* - służące do regulacji poziomu cieczy.

Działanie układu regulacji polega na tym, że gdy uchyb regulacji $e > 0$, czyli $y_0 > y(t)$, to sygnał wyjściowy przyjmuje stan logicznej jedynki, natomiast gdy $e < 0$, czyli $y_0 < y(t)$, to sygnał wyjściowy przyjmuje stan zera logicznego (rys. 3.4). W układzie regulacji w stanie ustalonym oscylacje wielkości regulowanej $y(t)$ zachodzą wokół wartości średniej



Rysunek 3.3 Schemat blokowy układu dwustawnej regulacji automatycznej

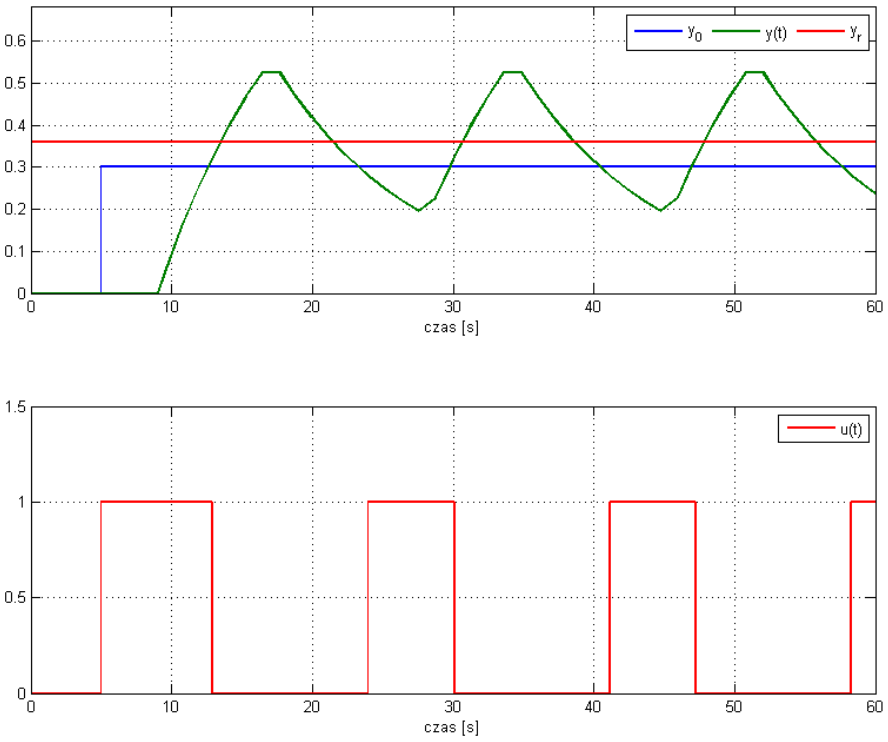
y_r (dla modelu obiektu w postaci inercji pierwszego rzędu z opóźnieniem). Przy czym wartość ta wyznaczana jest w następujący sposób:

$$y_r = \frac{y_{min} + y_{max}}{2} \tag{3.1}$$

W stanie ustalonym wartość średnia y_r może różnić się od wartości zadanej y_0 dając błąd ustalony:

$$e_{ust} = y_0 - y_r \tag{3.2}$$

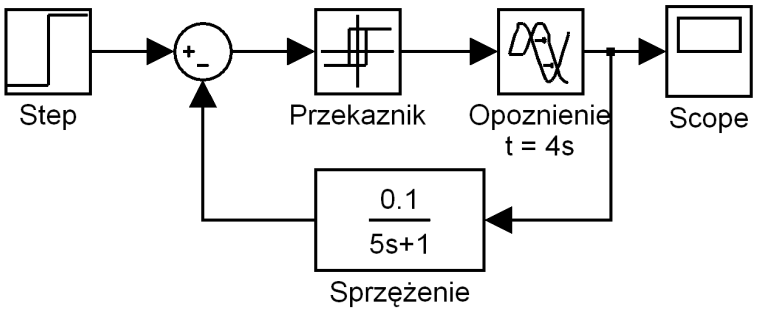
Błąd ten nie występuje w przypadku, gdy wartość zadana $y_0 = 0.5$, wtedy $y_r = y_0$ a czasy załączenia i wyłączenia są sobie równe.



Rysunek 3.4 Przebieg sygnału wielkości regulowanej $y(t)$ oraz sygnału wyjściowego $u(t)$ w układzie regulacji dwustawnej ($y_0 = 0.3$)

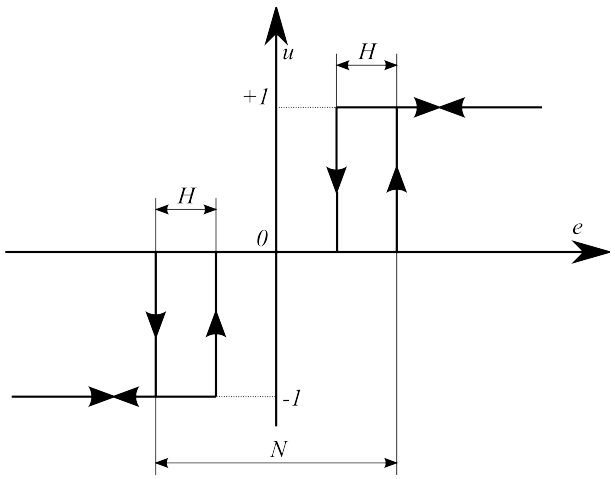
W celu poprawy jakości regulacji stosuje się regulację dwustawną z korekcją dynamiczną w postaci dodatkowego ujemnego sprzężenia zwrotnego wokół elementu przekaź-

nikowego (rys. 3.5). Zastosowanie korekcji dynamicznej prowadzi do zwiększenia częstotliwości łączeń, a przez to do zmniejszenia amplitudy oscylacji wielkości regulowanej. Jeżeli w roli sprzężenia zwrotnego zastosuje się obiekt inercyjny pierwszego rzędu taki regulator będzie miał właściwości zbliżone do regulatora ciągłego typu PD.



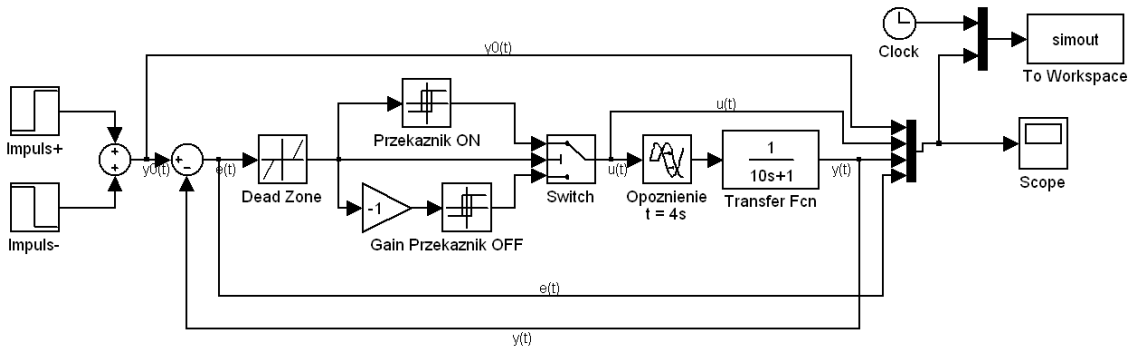
Rysunek 3.5 Regulator dwustawny z elementem inercyjnym pierwszego rzędu w roli sprzężenia korekcyjnego

3.2 Sterownie trójstawne



Rysunek 3.6 Charakterystyka statyczna obiektu trójstawnego (s - sygnał wyjściowy, e - sygnał wejściowy regulatora, H - strefa histerezy, N - strefa nieczułości) [6]

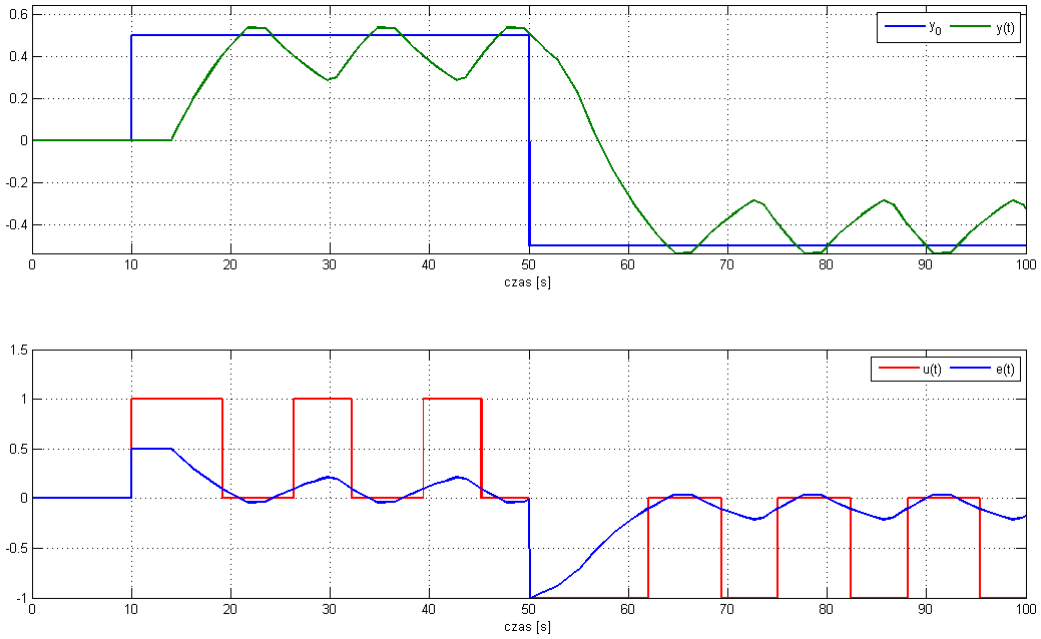
Regulatory trójstawne posiadają trój-stanową (-1, 0, $+1$) charakterystykę statyczną przedstawioną na rysunku 3.6. W porównaniu do regulatorów dwustawnych posiadają dodatkowy parametr (oprócz strefy histerezy H) - strefę nieczułości N. Dlatego dla tego typu układów jako parametr podaje się stosunek tych wartości - $\frac{H}{N}$. Regulatory trójpołożeniowe stosowane są najczęściej w układach, gdzie zachodzi zarówno nagrzewanie (odpowiada to stanowi 1) jak i chłodzenie (co odpowiada stanowi -1) - lub napełnianiu i opróżnianiu w przypadku sterowania poziomu cieczy lub ciśnienia. Często w celu poprawy działania regulatora trójstawnego stosuje się ujemne sprzężenie zwrotne zawierające element inercyjny kształtujący właściwości dynamiczne całego układu.



Rysunek 3.7 Schemat blokowy regulatora trójstawnego

Schemat blokowy regulatora trójstawnego prezentuje rysunek 3.7.

Istotne znaczenie regulatorów trójpołożeniowych wynika z ich możliwości sterowania silnikami nawrotnymi. Trzy stany na wyjściu regulatora odpowiadają kierunkowi obrotów w lewo, w prawo oraz zatrzymaniu silnika. Przebiegi czasowe regulator trójpołożeniowego przedstawione są na rysunku 3.8 [coś dopisać]



Rysunek 3.8 Charakterystyki dynamiczne obiektu trójstawnego

3.3 Regulacja ciągła - PID

Postać czasową idealnego regulatora proporcjonalno-całkująco-różniczkującego (PID) można zapisać w następujący sposób:

$$u(t) = k_p \left(e(t) + \frac{1}{T_i} \int_0^{\tau} e(\tau) d\tau + T_d \frac{de(t)}{dt} \right) \tag{3.3}$$

gdzie:

k_p - współczynnik wzmocnienia
T_i - czas całkowania
T_d - czas różniczkowania

Transmitancja operatorowa takiego regulatora ma następującą postać:

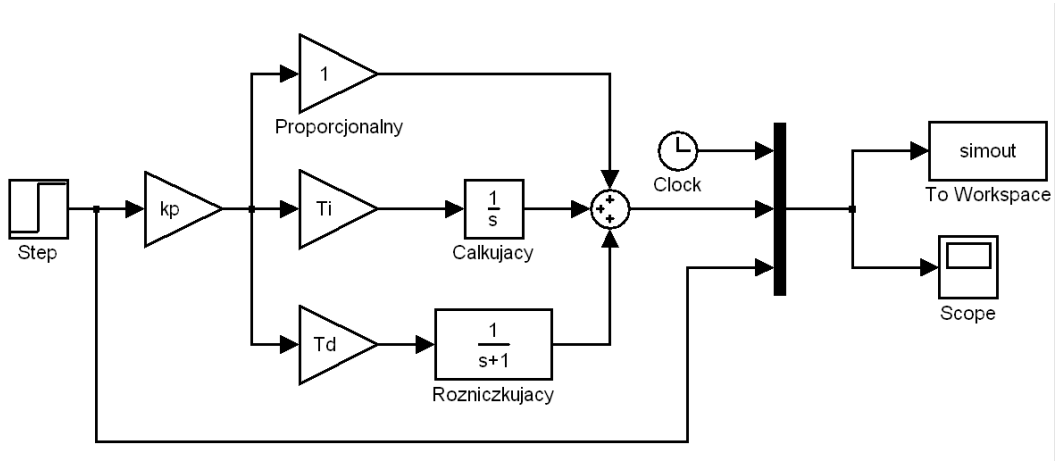
$$G_{PID}(s) = k_p \left(1 + \frac{1}{T_i s} + T_d s \right) \tag{3.4}$$

Jak wiadomo, niemożliwe jest fizyczne zrealizowanie idealnego elementu różniczkującego, dlatego też transmitancja rzeczywistego regulatora PID (rysunek 3.9) ma postać:

$$G_{PID}(s) = k_p \left(1 + \frac{1}{T_i s} + \frac{T_d s}{1 + Ts} \right) \tag{3.5}$$

gdzie:

T - stała czasowa inercji regulatora



Rysunek 3.9 Schemat blokowy regulatora PID

Do zmiany odpowiedzi regulatora służą parametry: k_p, T_i, T_d zwane nastawami regulatora. W zależności od przyjętych nastaw regulatora możemy uzyskać obiekty o właściwościach P, I, PI, PD oraz PID - odpowiednio regulator proporcjonalny, całkowy, proporcjonalno różniczkowy i proporcjonalno-całkowo-różniczkowy. "Regulatory o właściwościach wyłącznie D nie mogą spełniać zadań regulacji automatycznej, a regulatory ID zwykle nie są przydatne do regulacji automatycznej." [9]

Zakres proporcjonalności k_p regulatora jest to odwrotność współczynnika wzmocnienia proporcjonalnego wyrażona w procentach:

$$k_p = \frac{100\%}{k_p} \tag{3.6}$$

przy czym k_p to iloraz składowej proporcjonalnej wielkości wyjściowej regulatora do przyrostu odchyłki regulacji. Parametr k_p można określić następującą zależnością:

$$k_p = \frac{\Delta I_{we}}{\Delta I_{wy}} 100\% = \frac{I_{we2}}{I_{wy2} - I_{wy1}} 100\% \tag{3.7}$$

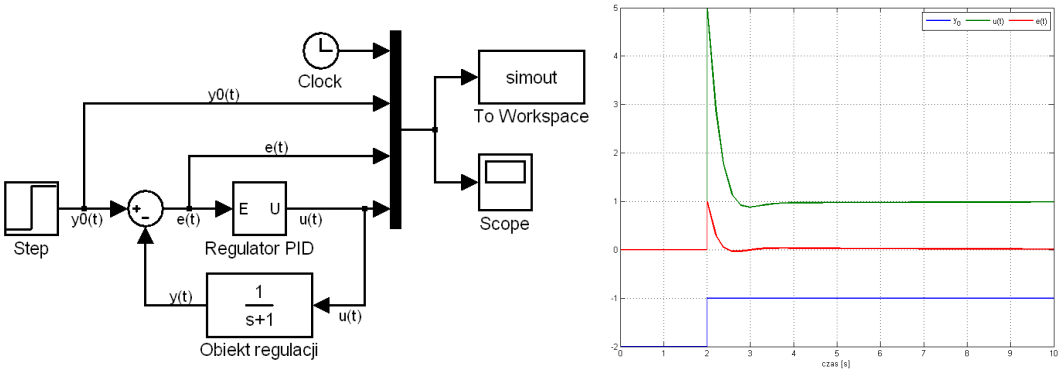
Gdzie:

- I_{we2} wartość wielkości wejściowej przy zerowej nastawie wartości zadanej i wartości wielkości wyjściowej w granicach $50\% \pm 20\%$
- I_{wy1} zmierzona wartość sygnału wyjściowego w momencie ustalenia się sygnału wyjściowego po zmianie sygnału wejściowego z I_{we2} na I_{we1}
- I_{wy2} zmierzona wartość sygnału wyjściowego po czasie zawierającym się w granicach $0,1T_i - 2T_i$ od momentu podania na wejście sygnału I_{we2}, w chwili zmiany wartości tego sygnału z I_{we2} na I_{we1}.

Czas zdwojenia T_i (zwany stałą całkowania) określa czas potrzebny na to, by przy wymuszeniu skokowym na wejściu regulatora uzyskać na jego wyjściu sygnał dwukrotnie większy od tego, który wynika ze współczynnika k_p

Czas wyprzedzenia T_d , to czas, po upływie którego od podania na wejście regulatora PD sygnału narastającego liniowo, sygnał na wyjściu regulatora osiągnął dwukrotną wartość tej, która wynika z działania różniczkowego.

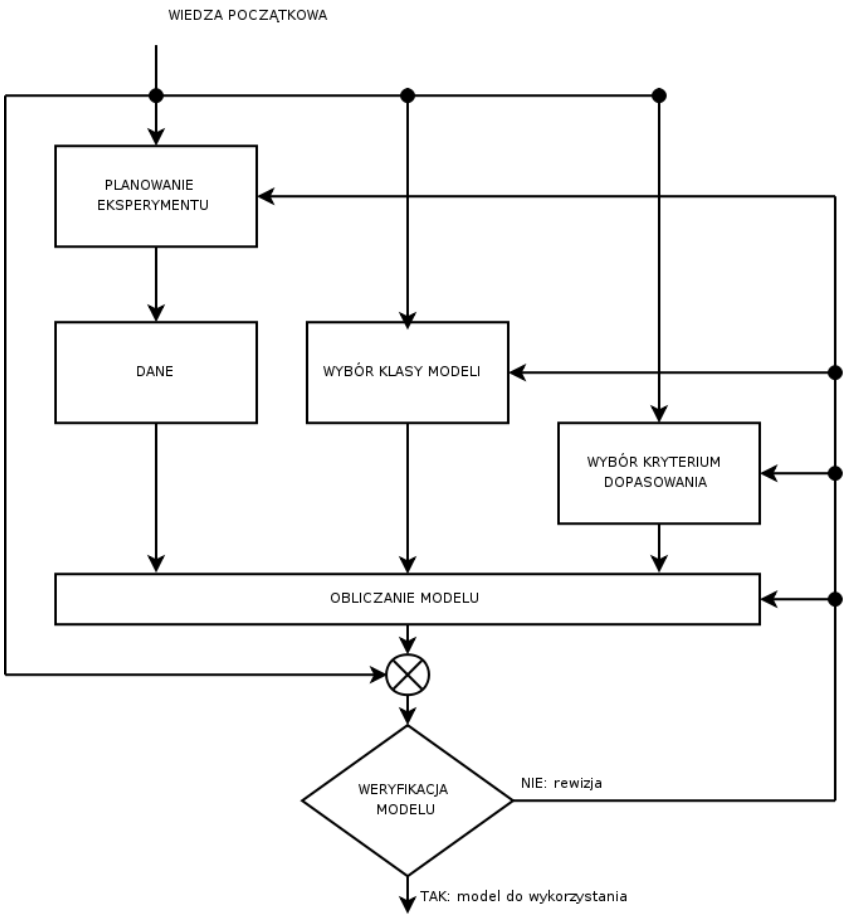
Regulację PID wykonuje się najczęściej w zamkniętej pętli sprzężenia zwrotnego. Model i odpowiedź układu na skok jednostkowy przedstawione są na rysunku 3.10.



(a) Model (b) Symulacja

Rysunek 3.10 Układ automatycznej regulacji PID

3.3.1 Identyfikacja obiektu



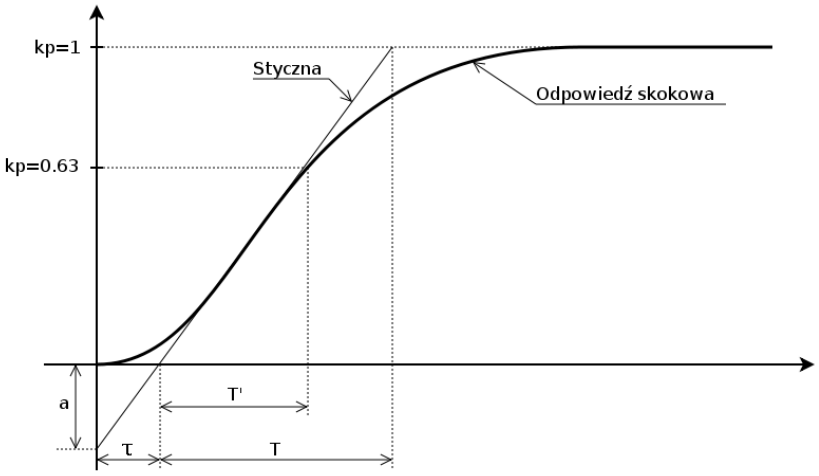
Rysunek 3.11 Schemat blokowy eksperymentu identyfikacji [10]

W celu uzyskania dobrej jakości regulacji w układach sterowania konieczna jest znajomość danego obiektu, czyli opis zależności wejściowo-wyjściowych. Opis taki ma zazwyczaj postać przybliżonego modelu matematycznego, danego na przykład w postaci równań różniczkowych lub różnicowych bądź w postaci odpowiednich transmitancji operatorowych.

Identyfikacja ma na celu uzyskanie możliwie dokładnego matematycznego modelu obiektu sterowania na podstawie posiadanej wiedzy początkowej oraz pomiarów sygnałów wejściowych i wyjściowych uzyskanych, jeżeli to możliwe, w trakcie specjalnie zaplanowanego eksperymentu identyfikacji. Ogólny schemat blokowy eksperymentu przedstawiony jest na rysunku 3.11. Eksperyment taki przeprowadza się zwykle na układzie otwartym, jest to jednak możliwe tylko wtedy, gdy układ jest stabilny. Identyfikacja jest możliwa również w układzie zamkniętym[2], wykorzystuje się wówczas układy samonastrajające[3], które na bieżąco dopasowują parametry modelu.

[2]stosuje się ją w przypadku obiektu niestabilnego, bądź sterowania adaptacyjnego
[3]ang. self-tuning systems

Identyfikację obiektów sterowania najczęściej przeprowadza się poprzez odczyt odpowiednich wartości z ich charakterystyki skokowej (patrz rysunek 3.12).



Rysunek 3.12 Odczyt parametrów identyfikacji [6]

Jest to dobre i (przy zachowaniu precyzji obliczeń) dokładne rozwiązanie, ponieważ większość obiektów automatyki można przedstawić jako obiekt inercyjny pierwszego rzędu z opóźnieniem o transmitancji:

$$K(s) = \frac{k_p \cdot e^{-s \cdot \tau}}{1 + s \cdot T_z} \tag{3.8}$$

Gdzie:

$K(s)$	transmitancja obiektu
k_p	stała proporcjonalności
τ	wartość odczytana z wykresu
T_z	wartość T lub T' odczytana z wykresu (należy przyjąć tą, która daje lepsze przybliżenie)

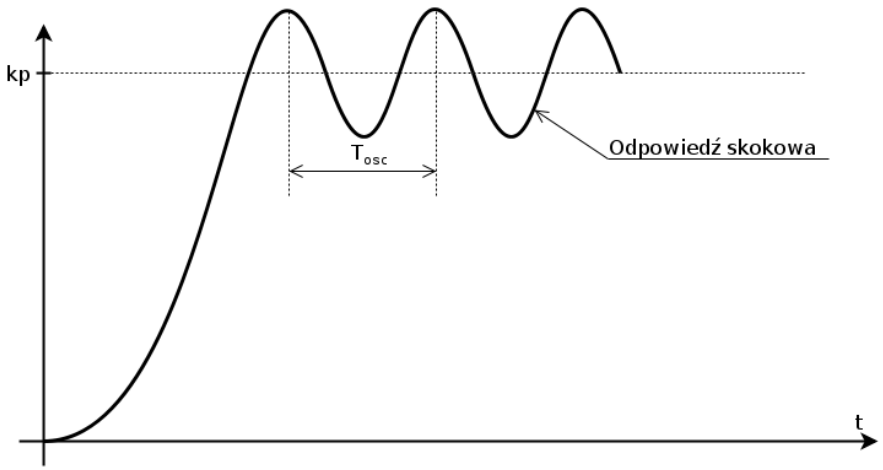
Czasem korzysta się również z transmitancji postaci:

$$K(s) = \frac{k_p}{(1 + s \cdot T_s)^n} \tag{3.9}$$

Gdzie:

$K(s)$	transmitancja obiektu
k_p	stała proporcjonalności
T_s	wartość T lub T' odczytana z wykresu (należy przyjąć tą, która daje lepsze przybliżenie)
n	rząd obiektu

Innym sposobem wyznaczenia parametrów obiektu jest wyznaczenie wzmocnienia krytycznego ($k_{p,kryt}$) oraz czasu oscylacji (T_{osc}) z wykresu w stanie ustalonym (patrz rysunek 3.13). Doświadczenie przeprowadzamy w układzie zamkniętym. Regulator ustawiamy tylko na działanie proporcjonalne i stopniowo zwiększamy parametr k_p od 0 do wartości krytycznej $k_{p,kryt}$, przy której można zaobserwować niegasnące oscylacje na wyjściu obiektu (układ znajduje się na granicy stabilności). Z otrzymanego wykresu odczytujemy czas oscylacji T_{osc}



Rysunek 3.13 Odczyt parametrów identyfikacji

3.3.2 Dobór nastaw regulatora

Podczas doboru nastaw należy mieć na uwadze:

- **zakres nastaw** - przedział zmienności parametrów k_p, T_i, T_d
- **ograniczenie nastaw** - niemożliwość nastawienia dowolnych wartości parametrów, pomimo iż leżą one w dopuszczalnych zakresach
- **zależność nastaw (interakcja)** - niemożność znalezienia pojedynczych elementów w strukturze regulatora, którymi można oddzielnie nastawiać wartości poszczególnych parametrów.

Należy również uwzględnić objawy wynikające z nieprawidłowego doboru nastaw. Ich zestawienie znajduje się w tabeli 3.1.

Tabela. 3.1 Objawy nieprawidłowego doboru nastaw regulatora

Objawy	Przyczyna
Trwałe oscylacje wielkości regulowanej	k_p i T_d za duże, T_i za małe
Przeregulowanie przy rozruchu	k_p za duże, T_i oraz T_d za małe
Przebieg przy rozruchu aperiodyczny, silnie tłumiony	k_p za małe, T_i za duże
Oscylacje zanikające	k_p za małe, T_i/T_d za małe

Nastawy według Zieglera - Nicholsa

Algorytm doboru nastaw według Zieglera-Nicholsa jest jednym z najczęściej opisywanych w literaturze. Jest to zestaw opracowanych doświadczalnie i sugerowanych ustawień regulatorów ciągłych. Ich wielkości zależą od wartości parametrów odczytanych z charakterystyk transmitancji obiektu. W wyniku eksperymentów powstały dwie metody Zieglera-Nicholsa, rozwijane i modyfikowane w latach późniejszych. [11]

Tabela. 3.2 Nastawy według I metody Zieglera-Nicholsa [11]

Typ Regulatora	k_p	T_i	T_d
P	$\frac{T}{\tau}$	∞	0
PI	$\frac{0{,}9T}{\tau}$	$\frac{\tau}{0{,}3}$	0
PID	$\frac{1{,}2T}{\tau}$	2τ	$0,5\tau$

Tabela. 3.3 Nastawy według II metody Zieglera-Nicholsa [11]

Typ Regulatora	k_p	T_i	T_d
P	$\frac{k_{p,kryt}}{2}$	∞	0
PI	$0,45k_{p,kryt}$	$\frac{T_{osc}}{1,2}$	0
PID	$0,6k_{p,kryt}$	$\frac{T_{osc}}{2}$	$\frac{T_{osc}}{}$

Nastawy według Cohen-Coona

Tabela. 3.4 Nastawy według Cohen-Coona [13]

Typ Regulatora	k_p	T_i	T_d
P	$\frac{1}{k_p}\left(\frac{T}{\tau}+0,35\right)$	∞	0
PI	$\frac{0,9}{k_p}\left(\frac{T}{\tau}+0,92\right)$	$\frac{3,3\cdot T+0,3\cdot\tau}{T+2,2\cdot\tau}\tau$	0
PID	$\frac{1,24}{k_p}\left(\frac{T}{\tau}+0,13\right)$	∞	0
PD	$\frac{1,35}{k_p}\left(\frac{T}{\tau}+0,18\right)$	$\frac{2,5\cdot T+0,5\cdot\tau}{T+0,61\cdot\tau}\tau$	$\frac{0,37\cdot T}{T+0,19\cdot\tau}\tau$

Nastawy według Chiena, Hronesa i Reswicka (tzw. CHR)

Tabela. 3.5 Wybór typu regulatora według Chiena, Hronesa i Reswicka [13]

Warunek	Typ Regulatora
$R > 10$	P
$10 < R < 7.5$	PI
$7.5 < R < 3$	PID
$R < 3$	regulator wyższego rzędu

Gdzie $R = \frac{T}{\tau}$.

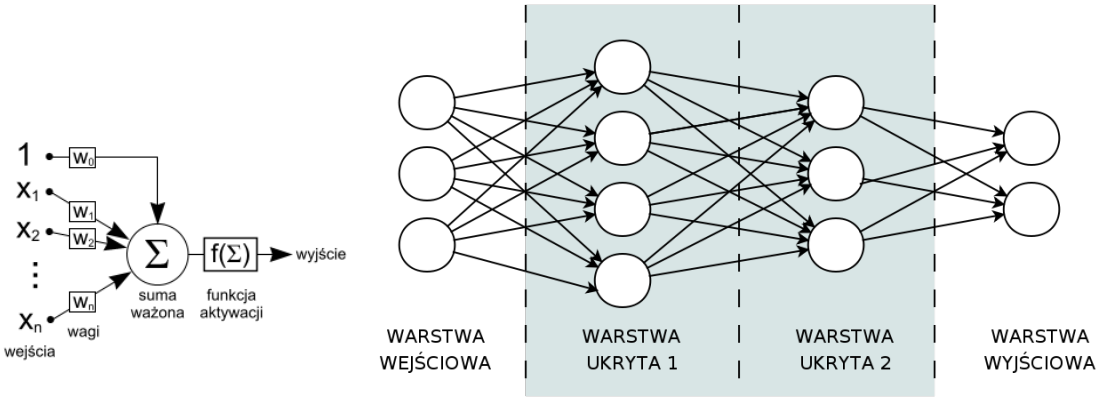
Tabela. 3.6 Nastawy według Chiena, Hronesa i Reswicka [13]

Przeregulowanie	0%			20%		
Typ Regulatora	k_p	T_i	T_d	k_p	T_i	T_d
P	$\frac{0,3}{a}$	∞	0	$\frac{0,7}{a}$	∞	0
PI	$\frac{0,35}{a}$	$1,2\cdot T$	0	$\frac{0,6}{a}$	T	0
PID	$\frac{0,6}{a}$	T	$\frac{\tau}{2}$	$\frac{0,95}{a}$	$1,4\cdot T$	$0,47\cdot\tau$

3.4 Inne sposoby regulacji

Wiele współczesnych urządzeń buduje się z wykorzystaniem tzw. sztucznej inteligencji pozwalającej w przybliżony sposób modelować reguły, naśladujące zachowania człowieka. W układach automatyki spotykamy się z adaptacyjnymi układami sterowania, które na bieżąco przestrajają regulatory dostosowując je do zmiennych parametrów obiektu. Są to jednak dość proste urządzenia w porównaniu z metodami, którymi dysponuje współczesna nauka, takimi jak obliczenia ewolucyjne, sieci neuronowe czy logika rozmyta[4].

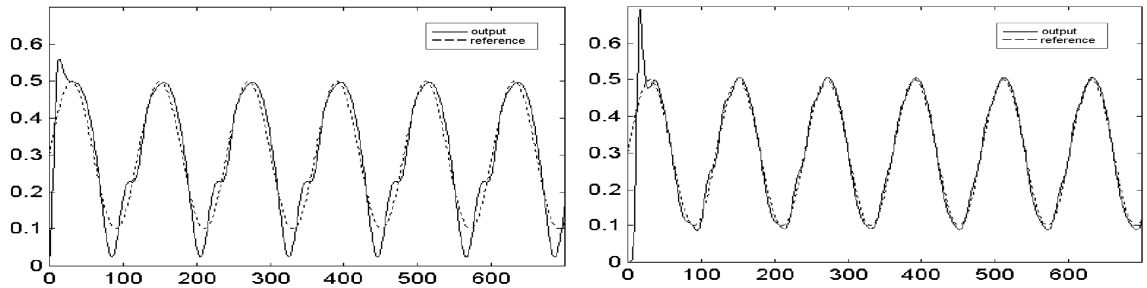
3.4.1 Regulatory neuronowe



(a) Neuron (b) Uproszczona struktura sieci neuronowej

Rysunek 3.14 Sztuczne sieci neuronowe [3]

Sieci neuronowe to struktury matematyczne realizujące obliczenia poprzez warstwy elementów zwanych neuronami. Inspiracją do ich struktury (patrz rysunek 3.14) była budowa naturalnych układów nerwowych. Posiadają one możliwość uczenia się, w wyniku czego mogą być stosowane jako układy przewidujące, w systemach z wieloma wejściami. Zastosowanie sieci neuronowej ma duży wpływ na poprawę parametrów regulacji, co można zaobserwować na rysunku 3.15



(a) Regulator PID (b) Regulator PID z siecią neuronową

Rysunek 3.15 Wpływ wykorzystania sieci neuronowej na poprawę parametrów regulacji. [15]

[4]ang. Fuzzy Logic

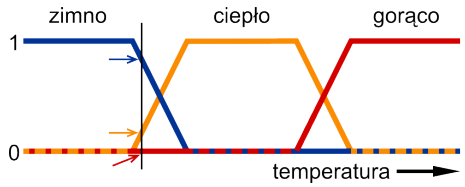
3.4.2 Regulatory rozmyte

Logika rozmyta to logika wielowartościową będąca uogólnieniem logiki dwuwartościowej. Pomiędzy dwa stany 0 i 1 logiki klasycznej wprowadza szereg stanów pośrednich, które określają stopień przynależności do elementu do zbioru.

"Logika rozmyta okazała się bardzo przydatna w zastosowaniach inżynierskich, czyli tam, gdzie klasyczna logika klasyfikująca jedynie według kryterium prawda/fałsz nie potrafi skutecznie poradzić sobie z wieloma niejednoznacznościami i sprzecznościami. Znajduje wiele zastosowań, między innymi w elektronicznych systemach sterowania (maszynami, pojazdami i automatami), zadaniach eksploracji danych czy też w budowie systemów ekspertowych.

Metody logiki rozmytej wraz z algorytmami ewolucyjnymi i sieciami neuronowymi stanowią nowoczesne narzędzia do budowy inteligentnych systemów mających zdolności uogólniania wiedzy." [3]

Przykład zastosowania logiki rozmytej przedstawiony jest na rysunku 3.16.



Rysunek 3.16 Przykład zastosowania logiki rozmytej [3]

Rozdział 4

Budowa mikrokontrolerów AVR ATmega

4.1 Architektura

Mikrokontrolery serii AVR zaprojektowane zostały z myślą o językach wysokiego poziomu. Ich budowa konsultowana była ze specjalistami w dziedzinie implementacji języka C. Dzięki temu otrzymano wygodny dla użytkownika, elastyczny i przede wszystkim wydajny rdzeń, pomimo iż skonstruowany jest on w architekturze 8 bitowej.

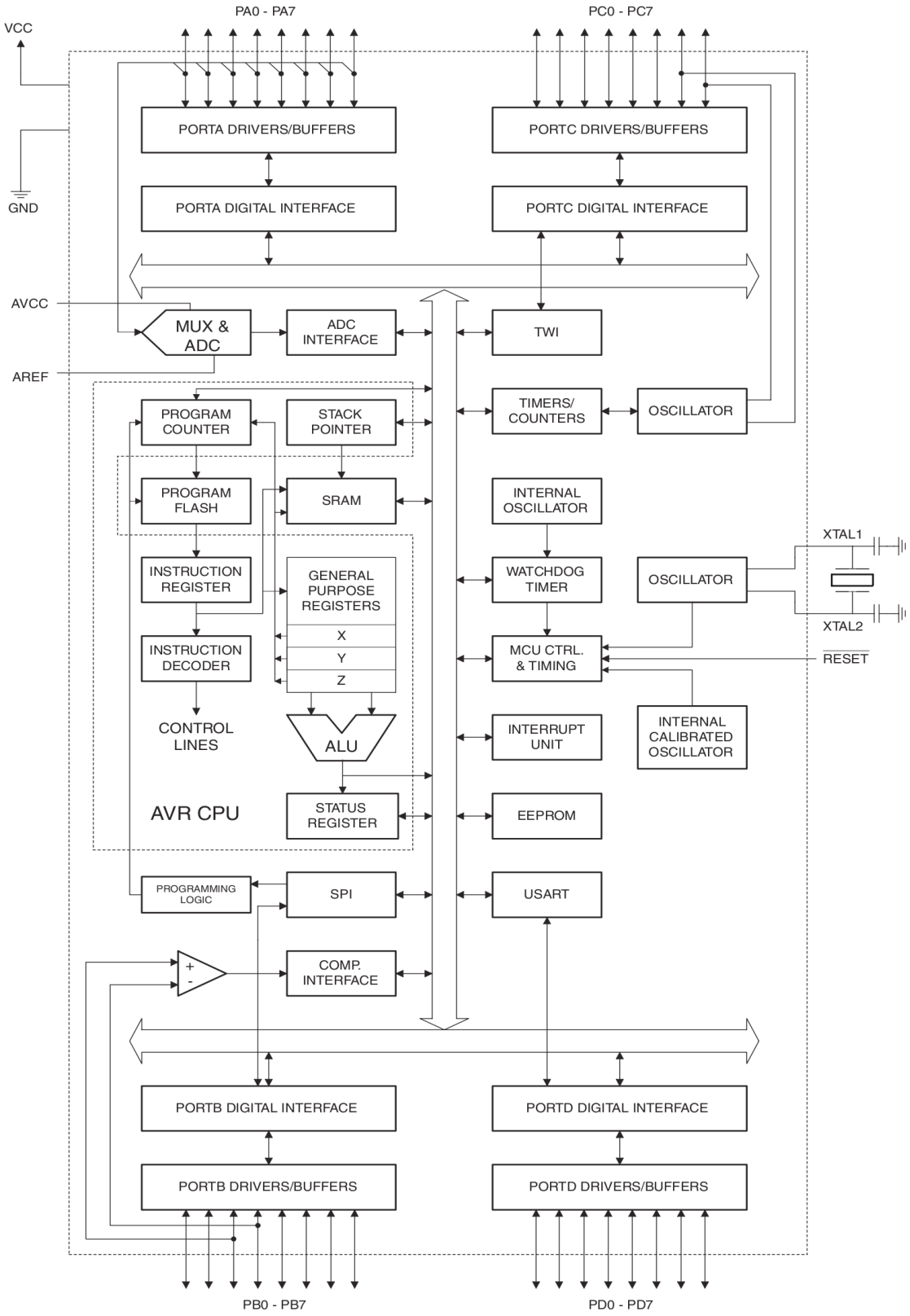
Układy AVR zbudowane są w oparciu o architekturę RISC[1]. Rdzeń mikrokontrolera dysponuje 131 instrukcjami, z których większość wykonywana jest w jednym cyklu zegarowym. Dzięki zastosowaniu układu sprzętowego operacje dzielenia wykonywane są jedynie w dwóch cyklach zegarowych.

Większość układów dysponuje zarówno interfejsem ISP[2] jak i JTAG pozwalającym na debugowanie układu w czasie jego pracy.

Do standardowych peryferiów każdego mikrokontrolera można zaliczyć:

- 8 i 16 bitowe liczniki pracujące w tybach: preskalera, komparatora
- Licznik czasu rzeczywistego z oddzielnym oscylatorem
- kanały PWM
- 8-kanałowy, 10-bitowy przetwornik analogowo cyfrowy (w tym kanały różnicowe)
- konparator analogowy
- interfejs SPI
- Watchdog

[1] ang. Reduced Instruction Set Computers - komputery o zredukowanej liczbie rozkazów
[2] ang. In-System Programming



Rysunek 4.1 Budowa mikrokontrolera ATmega16 [4]

4.2 Porty wejścia/wyjścia

Porty wejścia-wyjścia są podstawowym elementem komunikacyjnym mikrokontrolera. W przypadku układów AVR są one bardzo rozbudowane lecz ich programowanie nie sprawia większych problemów. Charakteryzują się one następującymi właściwościami:

- dwukierunkowość
- trójstanowość (patrz tabela 4.1)
- opcja wewnętrznego podciągania (należy ustawić flagę PUD w SFIOR[3])
- duża wydajność prądowa stopni wyjściowych
- symetryczna charakterystyka wyjściowa
- możliwość zmian linii na poziomie bitowym

Każdy mikrokontroler ATmega posiada kilka osobnych portów wejścia-wyjścia składających się z conajmniej ośmiu konfigurowanych niezależnie linii. Większość lini jest współdzielona z innymi peryferiami dlatego należy uważnie projektować układy. Każdy z portów dysponuje trzema rejestrami:

- PORT - Rejestr danych
- DDR (ang. Data Direcion Register) - Rejestr kierunkowy
- PIN - Rejestr wejściowy portu

Tabela. 4.1 Konfiguracje linii wyjściowych [5]

DDRxn	PORTxn	PUD	Tryb	Opis
0	0	x	wejście	Stan wysokiej impedancji (Hi-Z)
0	1	0	wejście	Podciągnięcie do zasilania
0	1	1	wejście	Stan wysokiej impedancji (Hi-Z)
1	0	x	wyjście	Wyjście w stanie niskim (Sink)
1	1	x	wyjście	Wyjście w stanie wysokim (Source)

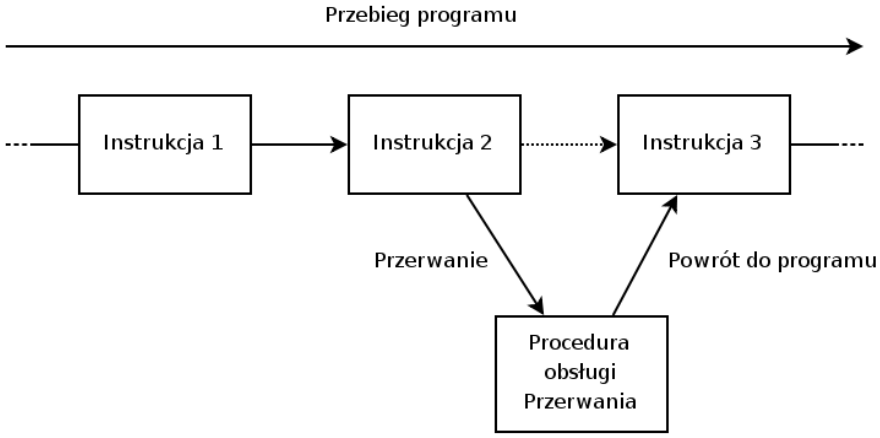
Gdzie n oznacza port (np. A) a x oznacza linię (np. 0).

Dokładne informacje o portach wejścia-wyjścia można znaleźć w dokumentacji konkretnego układu. Specyfikacje techniczne dla układów ATmega8, ATmega16 oraz ATmega32 znajdują się na załączonej płycie CD w katalogu $\backslash dokumenty \backslash AVR \backslash$

[3]ang. Special Function I/O Register

4.3 Przerwania

Przerwanie jest to sygnał generowany przez układy wewnęrzne (przerwanie wewnętrzne), zewnętrzne (przerwanie zewnętrzne) lub sam program (przerwanie programowe) powodujący wstrzymanie aktualnie wykonywanego programu i przejście do procedury obsługi danego przerwania. Po zakończeniu wykonywania procedury obsługi wykonywanie programu jest wznawiane. Miejsce w pamięci, gdzie znajduje się procedura obsługi przerwania jest zdefiniowane sprzętowo i nazywa się wektorem przerwania. Listę wektorół przerwań można znaleźć w dokumentacji mikrokontrolera.



Rysunek 4.2 Obsługa przerwania

Ponieważ wiele peryferiów ma możliwość generowania przerwań, wszystkie z nich są domyślnie zablokowane. Odblokowanie ich następuje poprzez wywołanie odpowiedniego kodu poprzez program mikrokontrolera. Każde z przerwań może być maskowane przez kasowanie bitów w odpowiednich rejestrach i w rejestrze statusu. Aby zapobiec nakładaniu się przrwań posiadają one priorytety.

Rejestry systemu obsługi przerwań:

- TIFR - Znaczniki przerwania z liczników/czasomierzy
- TIMSK - Maska przerwań liczników/czasomierzy
- GIMSK - Globalna maska przerwań

4.4 Układy licznikowe

Układy licznikowe to prawdopodobnie najczęściej wykorzystywane układy peryferyjne mikrokontrolerów "Najczęściej służą one do odmierzania czasu, zliczania zdarzeń, generacji przebiegów o zmiennej częstotliwości i wypełnieniu." [5].

Rodzina mikrokontolerów AVR jest dobrze wyposażona w tego typu układy. Najprostrze z nich wyposażone są w conajmniej 2 układy licznikowe. Firma atmel przyjęła stałe oznaczenie dla liczników znajdujących się w urządzeniach, liczniki 8-bitowe oznaczone są cyframi 0 i 2, natomiast 16-bitowe - 1 i 3. W dokumentacji technicznej są one nazywane układami *Timer/Counter*.

"Każdy z układów licznikowych ma minimum kilka rejestrów funkcyjnych, którw służą do nastawy parametrów pracy i informują o aktualnym stanie danego licznika. Ostatni człon nazwy takiego rejestru tworzy najczęściej cyfre, określająca numer przyporządkowanego mu licznika - nie jest to jednak regułą. Podobnie jest w przypadku zawartych w rejestrach znaczników - one zawsz są oznaczone cyfrą informującą o tym, którego licznika dotyczą." [5]

Podstawowym rejestrem przyporządkowanym każdemu licznikowi jest **TCNTn**, gdzie **n** oznacza numer licznika. "Zawiera on aktualną wartość zliczoną przez licznik i może być rejesterm 8- lub 16- bitowym, zależnie od typu licznika. W drugim przypadku TCNTn tworzą rejestry **TCNTnH** (MSB[4]) i **TCNTnL** (LSB[5]), z których pierwszy jest rejestrem buforowanym. Cecha ta powoduje, że oba rejestry są zapisywane i odczytywane zawsze jednocześnie, mimo 8-bitowej architektury mikrokontrolera. [...]

Niemalże wszystkie liczniki umożliwiają porównywanie wartości zliczanej z wartością ustaloną. Do przechowywania tej ostatniej służy rejestr **OCRn** lub rejestry **OCRnx** (zany licznik może mieć ich bowiem kilka, np. OCR1A, OCR1B, OCR1C) Podobnie jak w przypadku głównego rejestru licznikowego, również OCRnx mogą mieć postać 8- i 16-bitową (w drugim przypwadku podzielone są one na **OCRnxH** i **OCRnxL**).Mechanizm buforowy i tutaj ma zastosowanie, ale odnosi się tylko do operacji zapisu.

16-bitowe układy licznikowe mają funkcję przechwytywania aktualnej wartości licznika w skutek wykrycia pewnych zdarzeń wewnętrznych. Do przetrzymywania wartości "przechowywanej" służą rejestry **ICRnH** i **ICRnL**. Tworzą one kolejny, 16-bitowy rejestr buforowany.

Rejestrami, które służą do do nastawy trybu pracy liczników, są **TCCRn** (lub **TCCRnx** - niektóre liczniki wymagają bowiem większej ich liczby). Przechowywane są w nich bity **CSn** (zwykle trzy dla jednego modułu licznikowego, np. CS00, CS01, CS02), określające źródło sygnału taktującego przyporządkowany im licznik (ich znaczenie jest zależne od typu mikrokontrolera i rodzaju licznika). Bity o oznaczeniu **WGMn** (zwykle dwa lub cztery dla jednego modułu np. WGM10, WGM11, WGM12 i WGM13) określają natomiast tryb pracy danego licznika (normalny, CTC, PWM). Znaczniki **COMn** (zwykle ich para, np COM00 i COM01) pozwalają wybrać sposób automatycznej generacji sygnałów wyjściowych, a ich znaczenie jest zależne od nastawy bitów WGMn." [5]

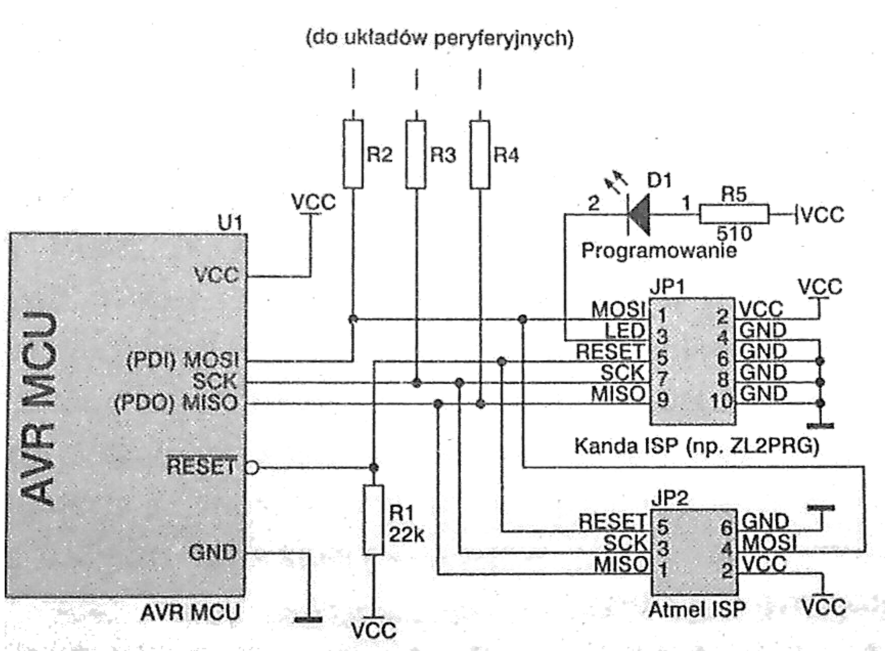
[4]Most Significant Bit - ang. najbardziej znaczący bit
[5]Least Significant Bit - ang. najmniej znaczący bit

4.5 Programowanie

Pamięć programu orazraz bity sterujące (***Fuse bits*** wszystkich układów serii AVR mogą być programowane conajmniej na dwa sposoby. "Programowanie to może odbywać się w urządzeniu, w którym mikrokontroler pracuje lub w specjalnym programatorze. Programowania w systemie ISP[6] dokonuje się najczęściej poprzez interfejs szeregowy **SPI** (czasem stosowany jest JTAG). [···]

Programatory SPI umożliwiają programowanie układu w urządzeniu docelowym (tryb ISP), co stanowi duże udogodnienie. Tryb ISP jest dostępnu, ponieważ interfejs ISP używa tylko trzech linii sygnałowych. Liniami tymi są: **SCK** (linia zegarowa), **MISO** (szeregowe wyjście), **MOSI** (szeregowe wejście). "[5]

Na rysunku 4.3 przedstawiono układ połączenia interfejsu programowania do układu. Ważne jest aby współdzielone linie portu nie zakłócały procesu programowania.



Rysunek 4.3 Sposób podłączenia programatora ISP do mikrokontrolera w układzie docelowym [5]

"Poza interfejsami programowania większość mikrokontrolerów ATmega wyposażono w tzw. interfejs uruchomieniowy[7], którym jest **JTAG** lub **debugWIRE**. Ten ostatni jest najnowszym wynalazkiem firmy Atmel i pozwala na transmisję jednoprzewodową."[5]

[6] ang. In System Programming
[7] ang. debug interface

Rozdział 5

Programowanie mikrokontrolerów AVR ATmega

Układy mikroprocesorowe AVR produkowane przez firmę Atmel są obecnie najpopularniejszą i najbardziej dostępną na rynku rodziną mikroprocesorów. Wynika to z prostoty ich programowania, łatwo dostępnej dokumentacji i darmowych narzędzi. "Rdzeń rodziny AVR zaprojektowano z myślą o językach wysokiego poziomu - konkretnie C. Przy pracach nad układem konsultowano się z ekspertami w dziedzinie implementacji tego języka, czego wynikiem jest elastyczny i wygodny dla programisty rdzeń, którego wydajności (pamiętając, że mamy do czynienia z układem ośmiobitowym) nie można wiele zarzucić." [5] Z tego też powodu programy przedstawione w niniejszym rozdziale napisane będą w języku C a jedynie niektóre przykłady podane będą w Assemblerze.

W dalszej części omówiona zostanie instalacja i konfiguracja podstawowych narzędzi służących do tworzenia oprogramowania dla mikrokontrolerów AVR. Istnieje wiele programów, które ułatwiają programowanie zarówno w języku C jak i assemblerze. W przypadku systemu operacyjnego Windows możemy skorzystać z generycznego produktu firmy Atmel jakim jest *AVR Studio*[1]. Dla platform Unix-owych (np.: Linux, BSD, Solaris) istnieje Open-Sourcowe narzędzie *KontrollerLab*[2].

Pokaźnym zbiorem informacji na temat tej serii mikrokontrolerów jest forum internetowe użytkowników mieszczące się pod adresem `http://www.avrfreaks.net`

Wszystkie użyte pliki instalacyjne znajdują się na załączonej płycie CD w katalogu *\instalki*. Na płycie znajdują się następujące wersje oprogramowania:

- AVR Studio 4.13 (marzec 2008) z poprawkami SP1 i SP2
- WinAVR 2007-12-21 - pakiet zawierający AVR-GCC i inne przydatne narzędzia
- PonyProg2000 2.07c - program obsługujący wiele programatorów (m.in. STK-200)

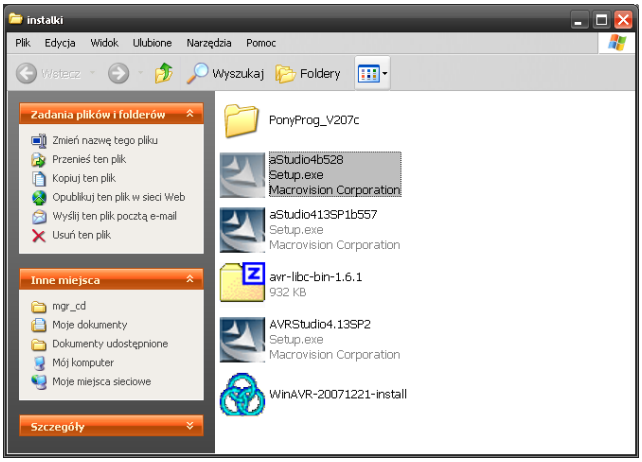
5.1 Instalacja AVR Studio 4

AVR Studio jest najpopularniejszym zintegrowanym środowiskiem programistycznym (ang. IDE - Integrated development environment) dla mikrokontrolerów AVR. Popularność jego wynika z faktu iż jest ono wydawane przez firmę Atmel - producenta mikrokontrolerów AVR.

[1] program dostępny na stronie `http://atmel.com/avrstudio`
[2] program dostępny na stronie `http://cadmaniac.org`

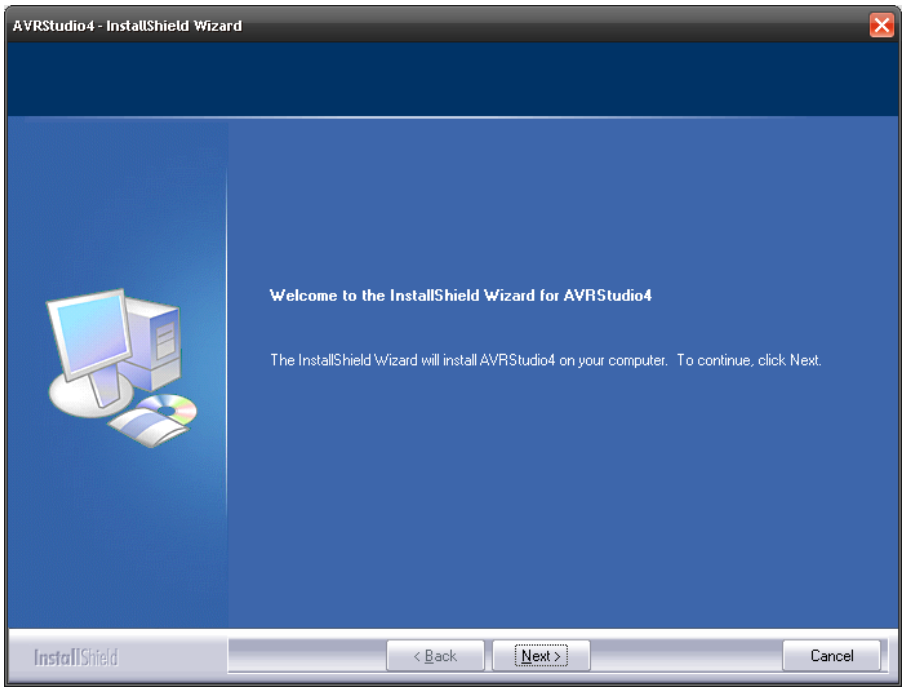
5.1.1 Instalacja programu

Aby rozpocząć instalację należy uruchomić plik *aStudio4b528.exe* z katalogu *\instalki* znajdującego się na załączonej płycie CD (patrz rysunek 5.1. Alternatywnie można ściągnąć nowszą wersję programu ze strony producenta (`http://atmel.com/avrstudio`).



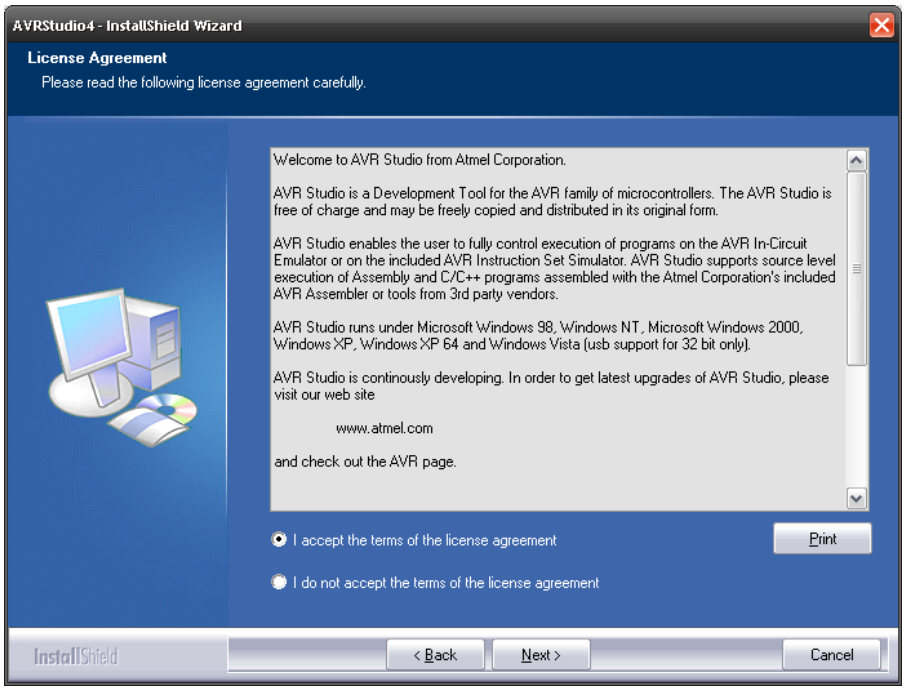
Rysunek 5.1 AVR Studio - Instalacja

Okno uruchmionego instalatora przedstawione jest na rysunku 5.2.



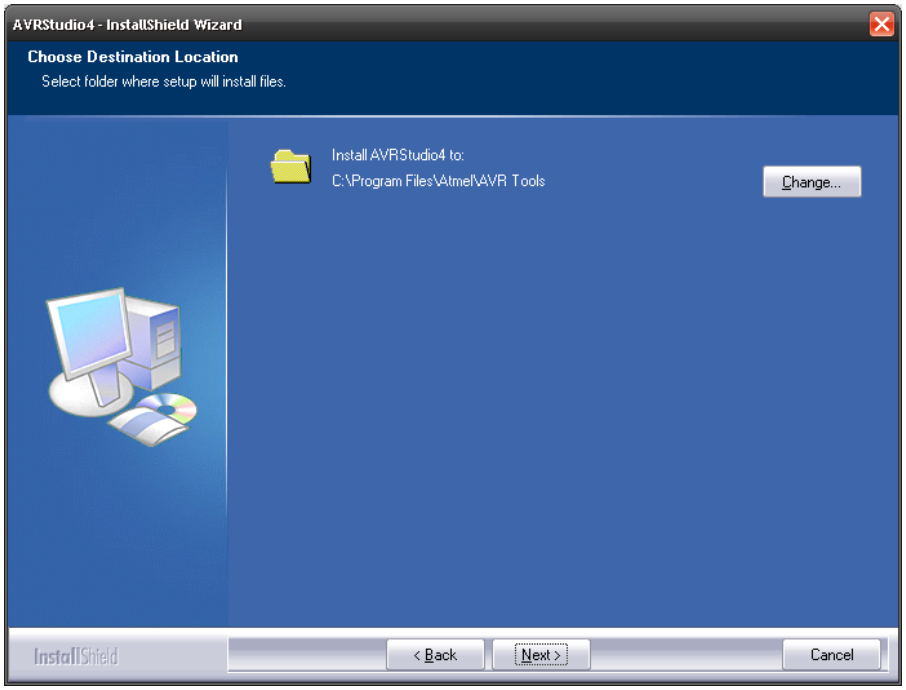
Rysunek 5.2 AVR Studio - Okno instalatora

Aby kontynuować instalację konieczna jest akceptacja licencji.



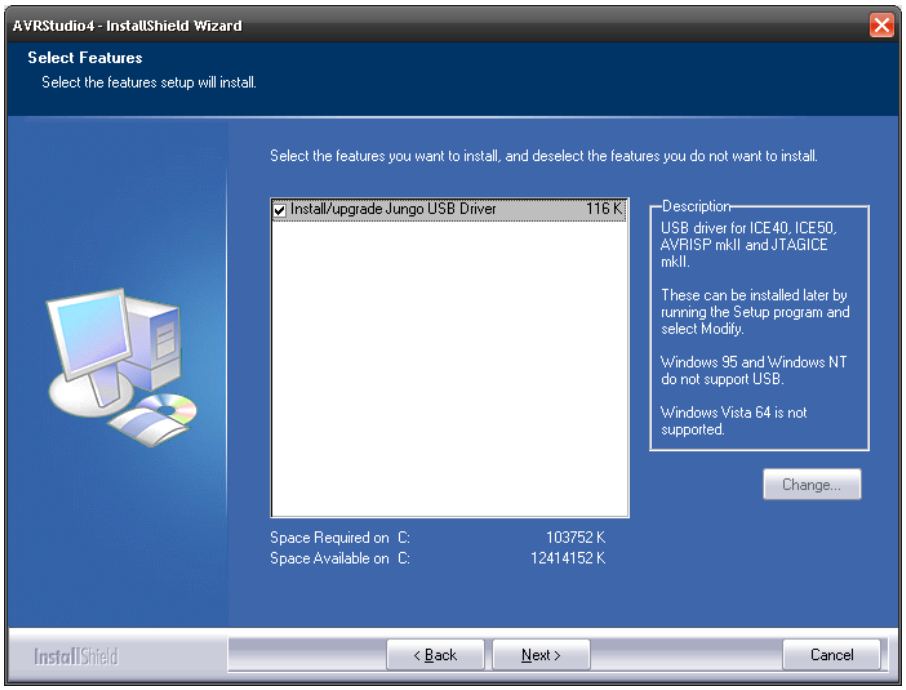
Rysunek 5.3 AVR Studio - Akceptacja licencji

Po akceptacji licencji należy wybrać katalog, do którego zostanie zainstalowana aplikacja. Domyślnie jest to *C:\Program Files\Atmel\AVR Tools*



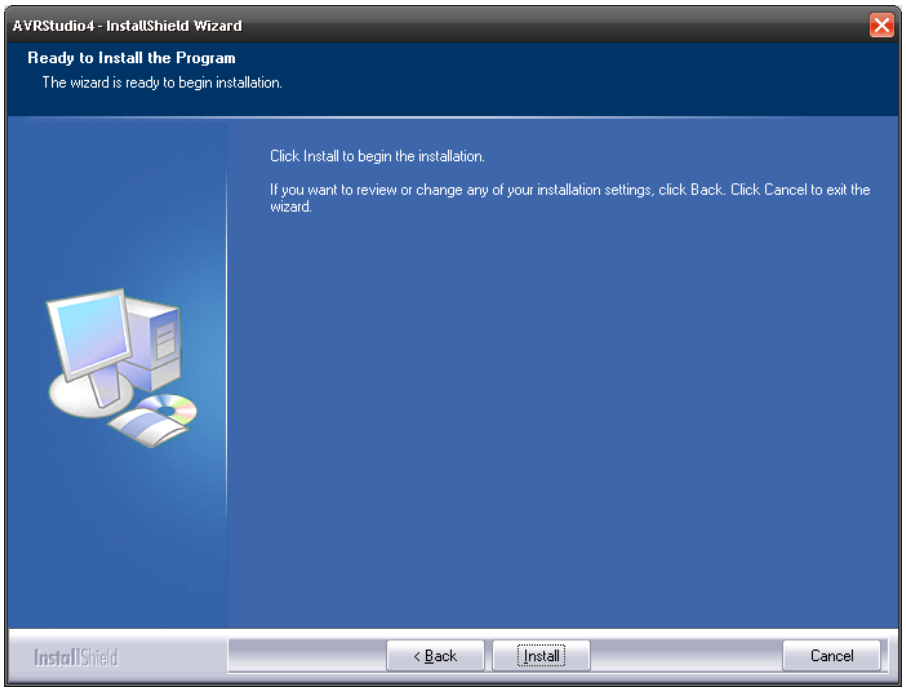
Rysunek 5.4 AVR Studio - Wybór katalogu instalacyjnego

W śród opcji instalacyjnych domyśnie zaznaczona jest aktualizacja sterownika USB do programatorów firmy Atmel. W przypadku korzystania z zestawu uruchmieniowego 300-K bądź innego programatora kompatybilnego z STK200 lub STK300 aktualizacja ta nie jest wymagana.

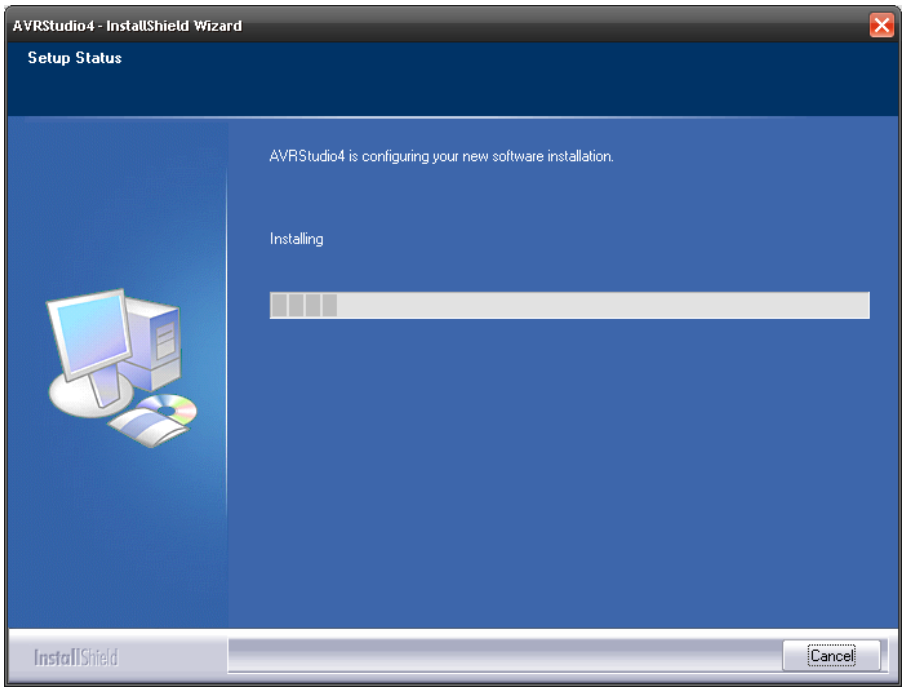


Rysunek 5.5 AVR Studio - Opcje instalacji

Możemy przystąpić do instalacji oprogramowania.

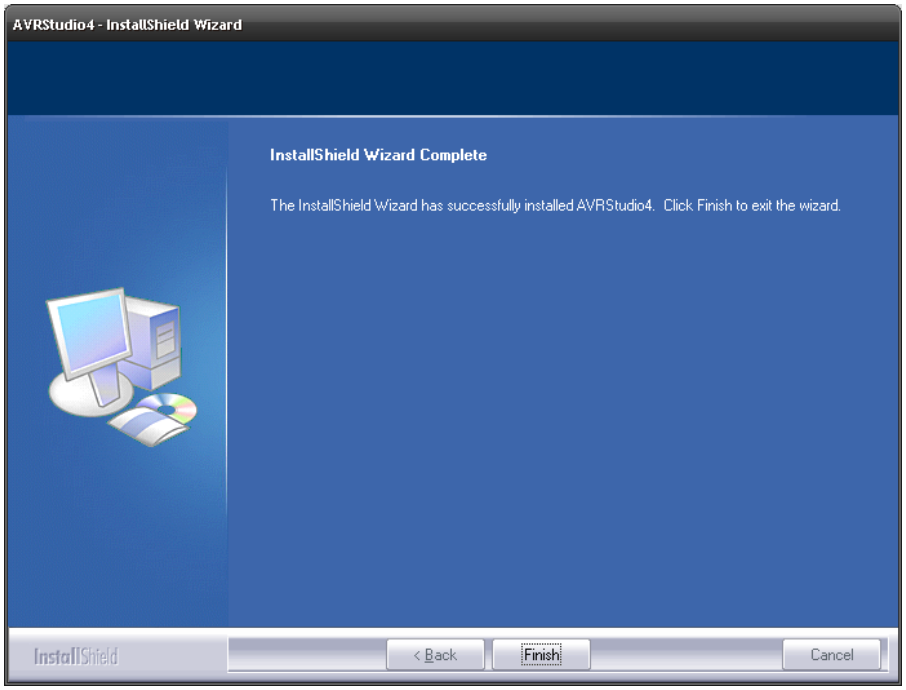


Rysunek 5.6 AVR Studio - Rozpoczęcie instalacji



Rysunek 5.7 AVR Studio - Proces instalacji

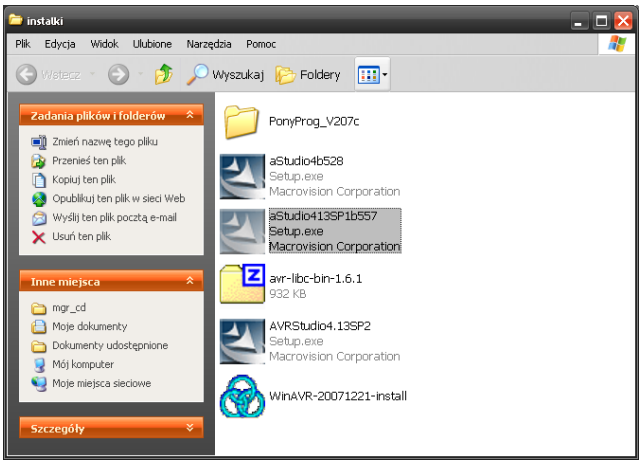
AVR Studio 4 zostało zainstalowane



Rysunek 5.8 AVR Studio - Zakończenie instalacji

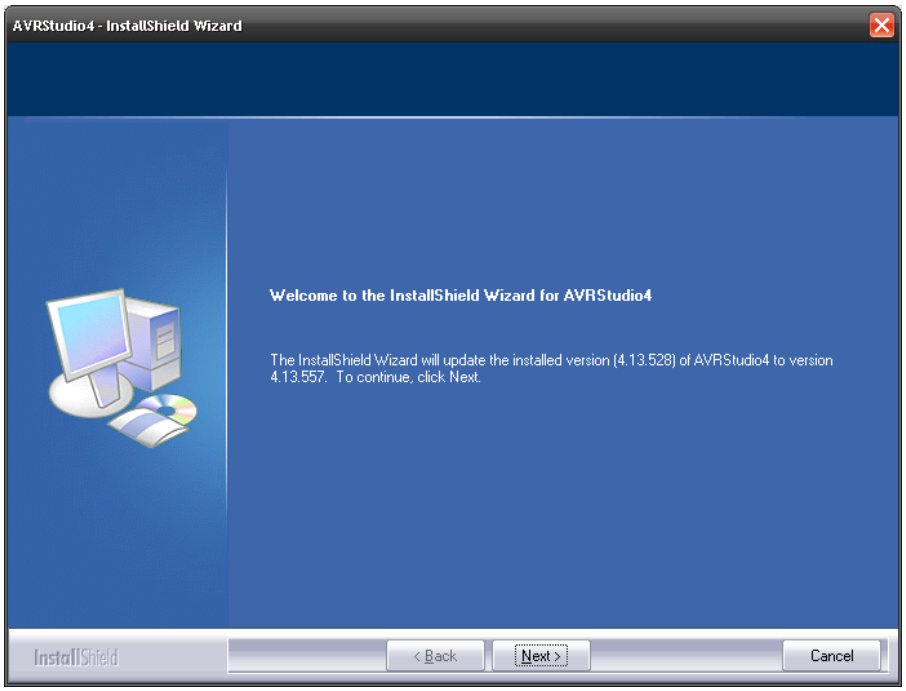
5.1.2 Instalacja poprawki SP1

Po zainstalowaniu podstawowej wersji AVR Studio konieczne jest zainstalowanie poprawki SP1 (ang. Service Pack)

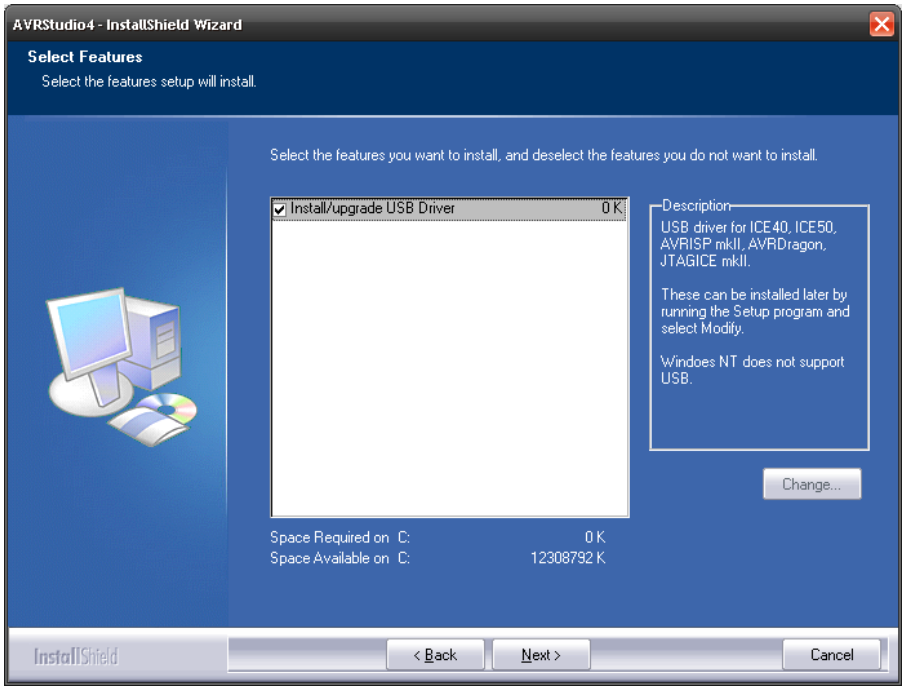


Rysunek 5.9 AVR Studio - Instalacja popraki SP1

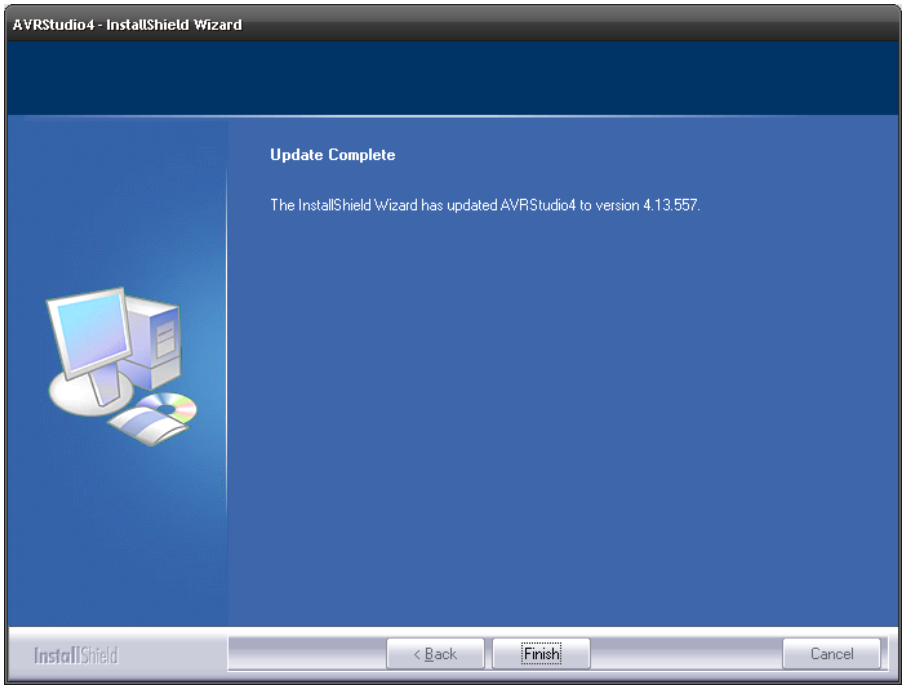
W celu instalacji poprawki SP1 dla AVR Studio 4 należy postępować analogicznie jak przy instalaji podstawowej wersji oprogramowania.



Rysunek 5.10 AVR Studio SP1 - Okno instalatora



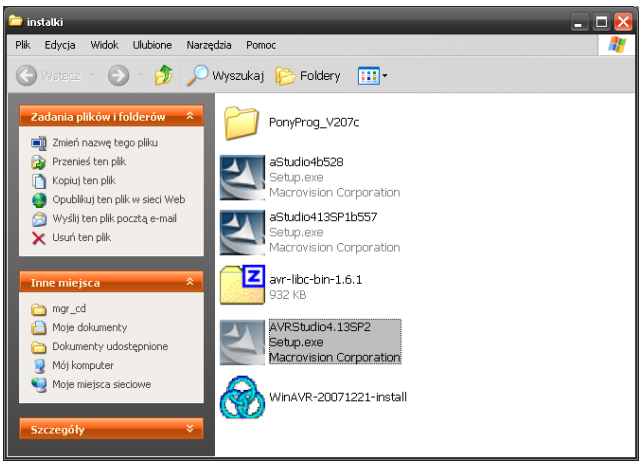
Rysunek 5.11 AVR Studio SP1 - Opcje aktualizacji



Rysunek 5.12 AVR Studio SP1 - Zakończenie aktualizacji

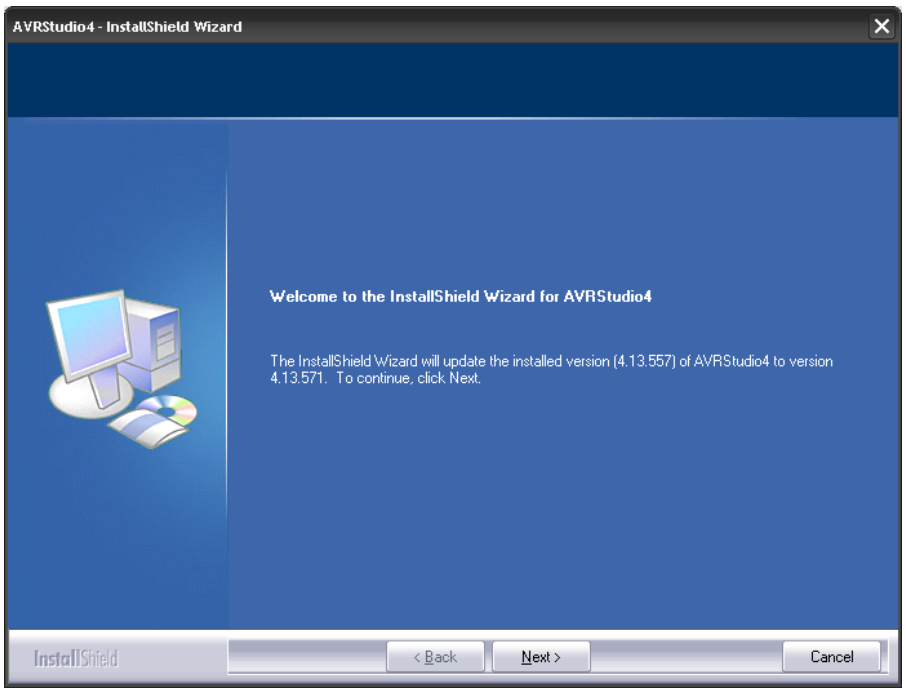
5.1.3 Instalacja poprawki SP2

Po zainstalowaniu aktualizacji SP1 należy przeprowadzić kolejną aktualizację - tym razem do wersji SP2.

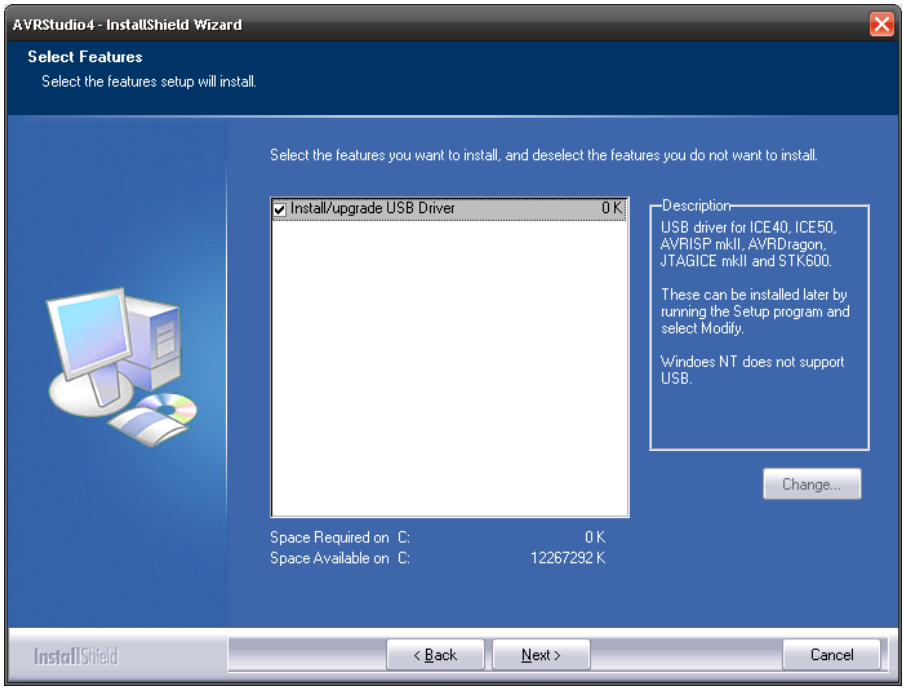


Rysunek 5.13 AVR Studio - Instalacja popraki SP1

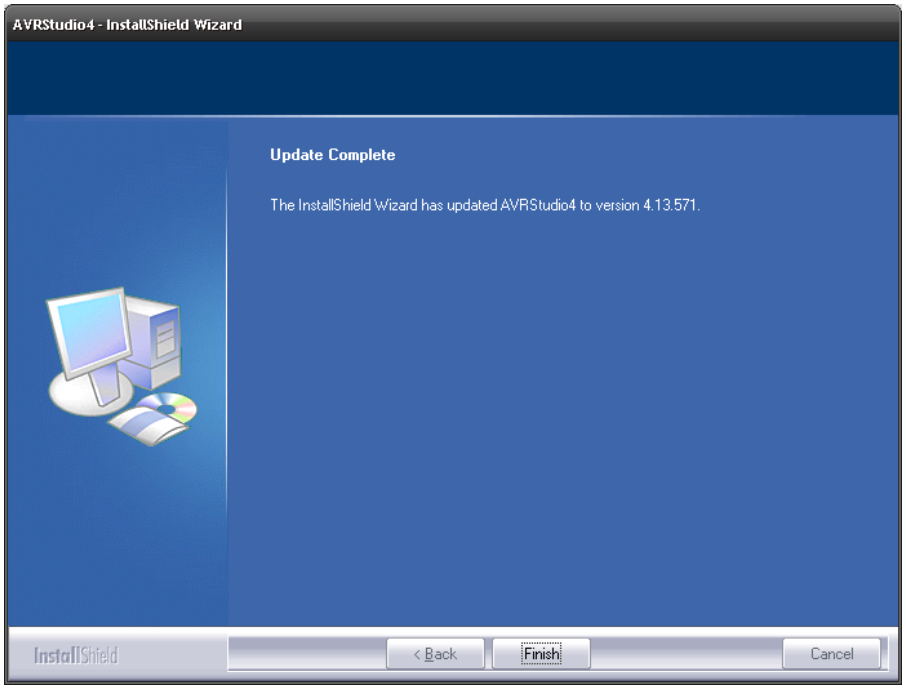
W celu instalacji poprawki SP2 dla AVR Studio 4 należy postępować identycznie jak przy instalaji poprawki SP1.



Rysunek 5.14 AVR Studio SP1 - Okno instalatora



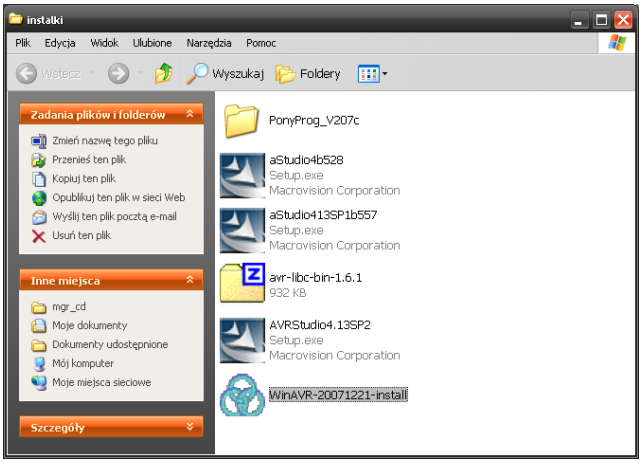
Rysunek 5.15 AVR Studio SP1 - Opcje aktualizacji



Rysunek 5.16 AVR Studio SP1 - Zakończenie aktualizacji

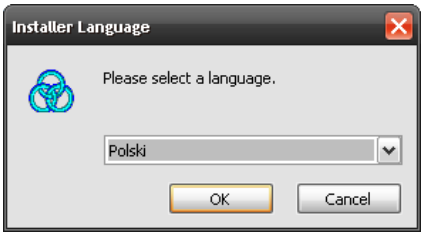
5.2 Instalacja WinAVR

AVR-GCC jest darmowym kompilatorem języka C (i assemblerem) dla platformy AVR utworzonym w ramach projektu GNU[3]. Zawarte w tym pakiecie oprogramowanie jest wydawane na licencji BSD bądź GNU GPL w wersji 2 lub nowszej, co umożliwia zastosowanie tego pakietu nawet do celów komercyjnych. Pełna treść tej licencji w języku angielskim znajduje się na załączonej płycie w pliku *gpl-2.0.txt*.



Rysunek 5.17 WinAVR - Instalacja

Po uruchomieniu instalator zapyta nas o wybór języka instalacji.



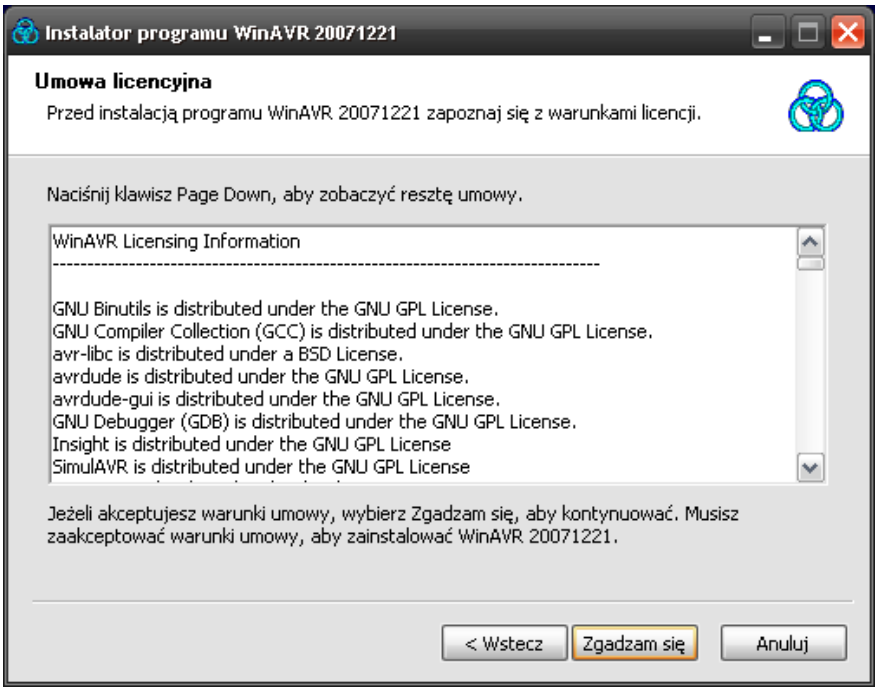
Rysunek 5.18 WinAVR - Wybór języka

[3]więcej informacji na stronie www.gnu.org

Po wybraniu języka zostanie toworzone główne okno instalacji.

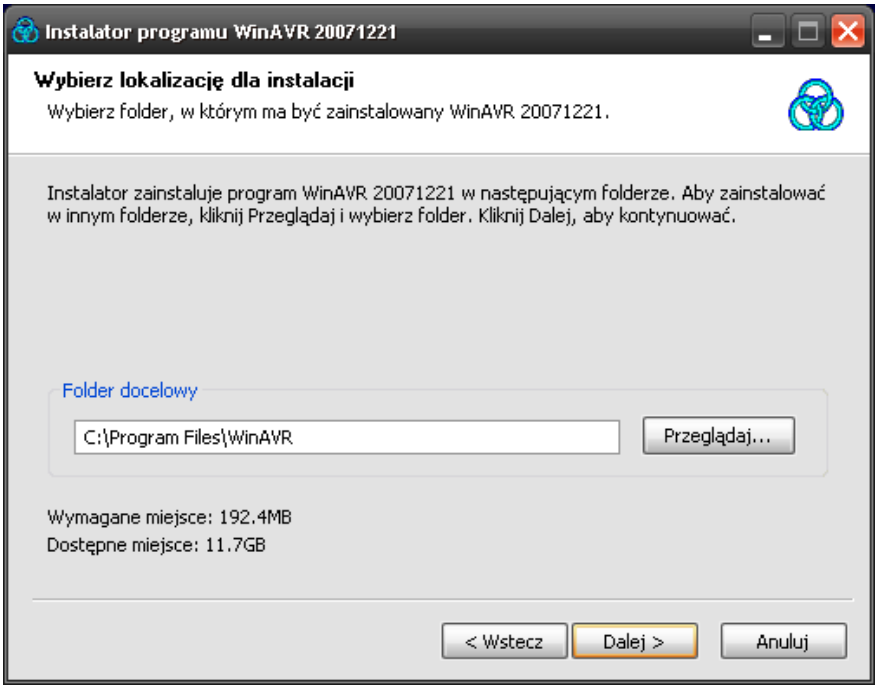
Rysunek 5.19 WinAVR - Okno instalatora

Aby kontynuować należy zaakceptować warunki licencji.



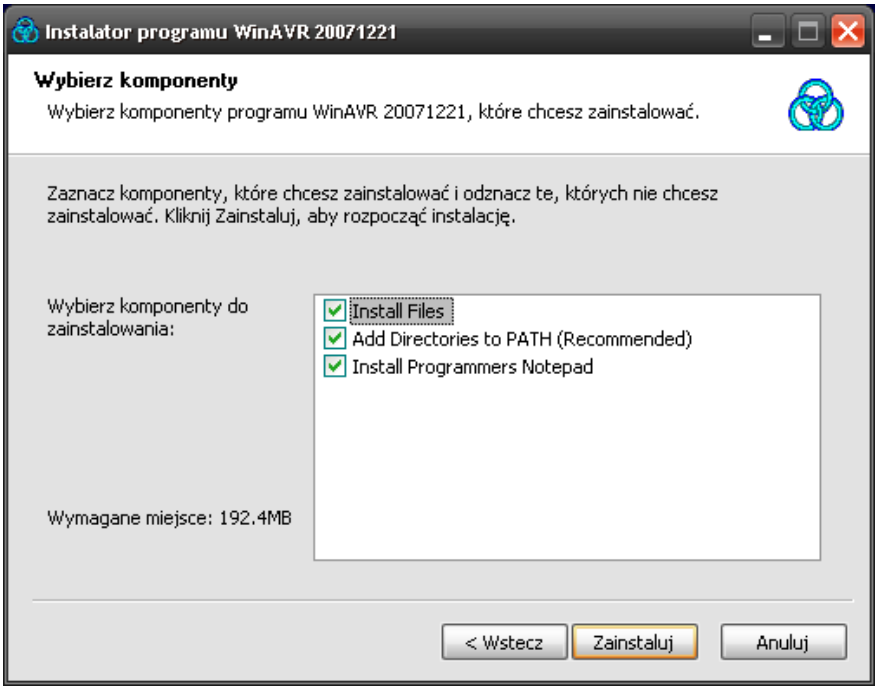
Rysunek 5.20 WinAVR - Akceptacja licencji

W oknie wyboru katalogu instalacyjnego należy wpisać *C:\Program Files\WinAVR*



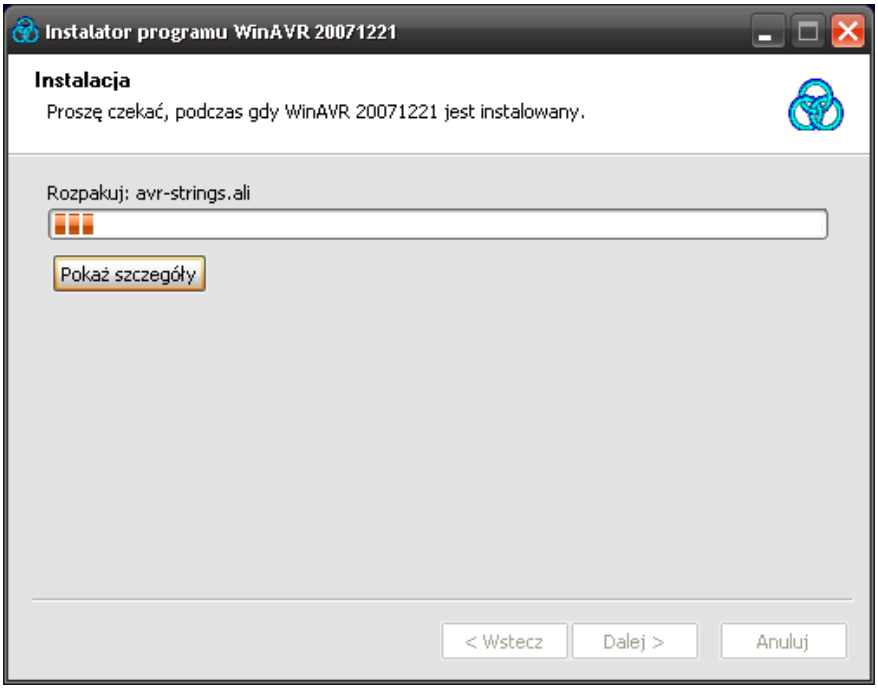
Rysunek 5.21 WinAVR - Wybór katalogu instalacyjnego

W oknie wyboru komponentów instalacji należy zaznaczyć wszystkie opcje.



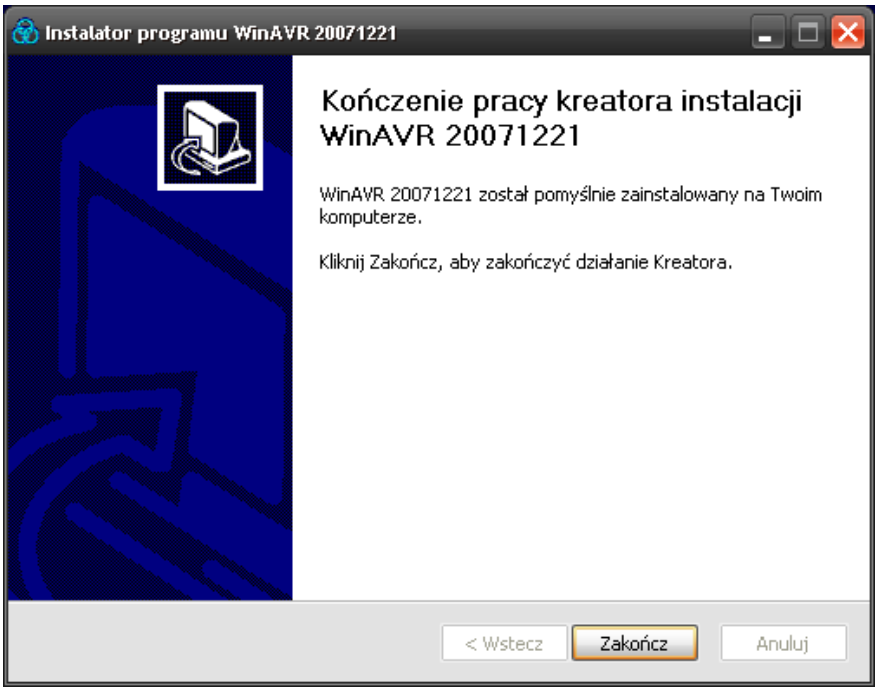
Rysunek 5.22 WinAVR - Opcje instalacji

Rozpocznie się instalacja oprogramowania.



Rysunek 5.23 WinAVR - Proces instalacji

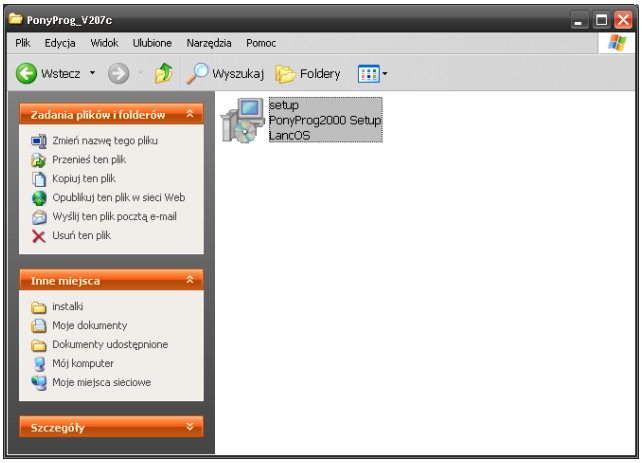
Koniec instalacji.



Rysunek 5.24 WinAVR - Zakończenie instalacji

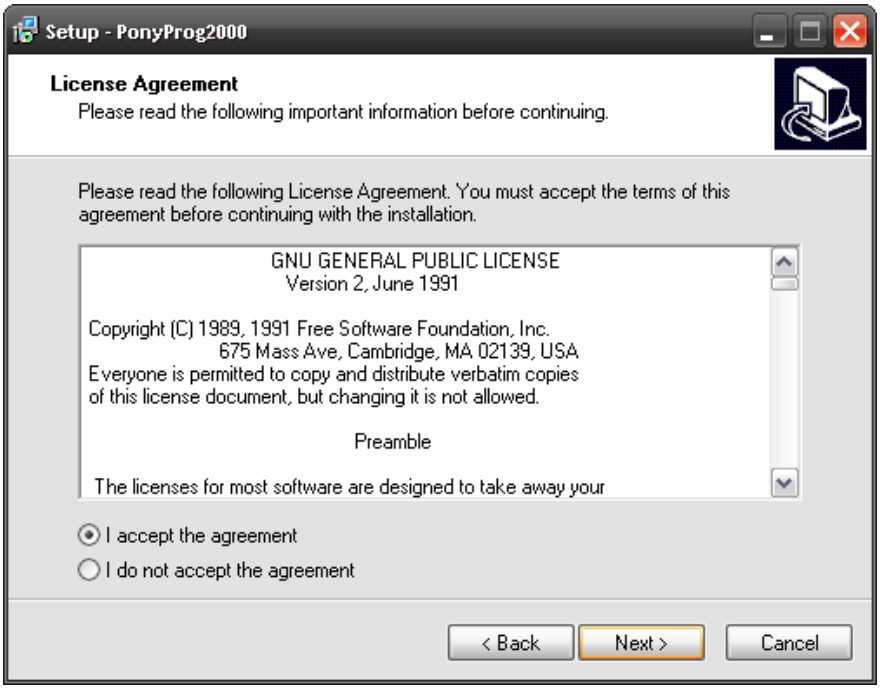
5.3 Instalacja PonyProg2000

PonyProg2000 to jeden z najpopularniejszych i darmowych programów do opsługi programatorów mikrokontrolerów AVR. Jest on ciągle aktualizowany i posiada wsparcie niemalże wszystkich układów z serii AVR. Program ten jest również wydany na licencji GPL.



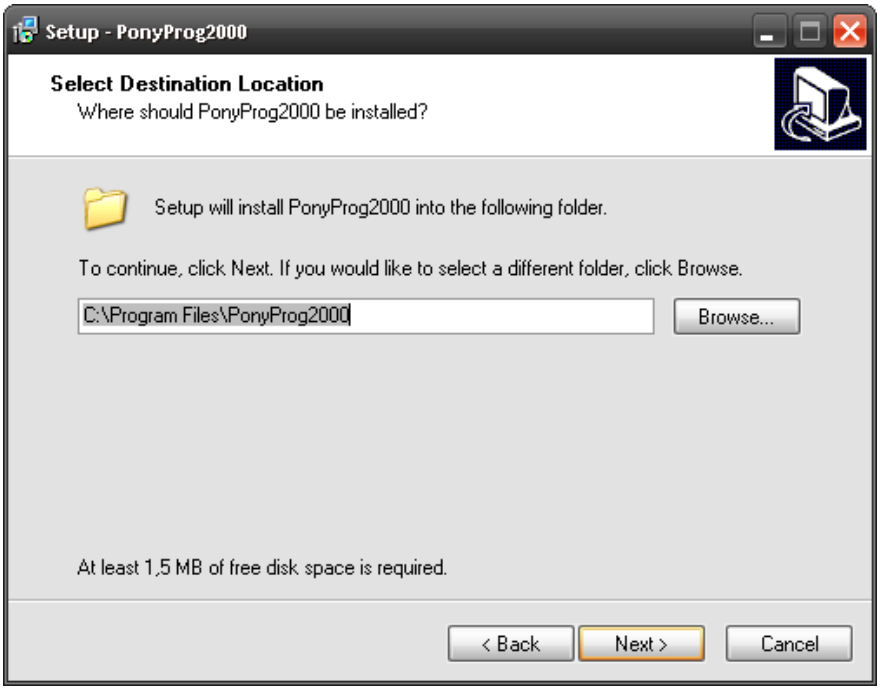
Rysunek 5.25 PonyProg2000 - Instalacja

Po uruchomieniu istalatora pokaże się treść licencji oprogramowania, którą należy zaakceptować.



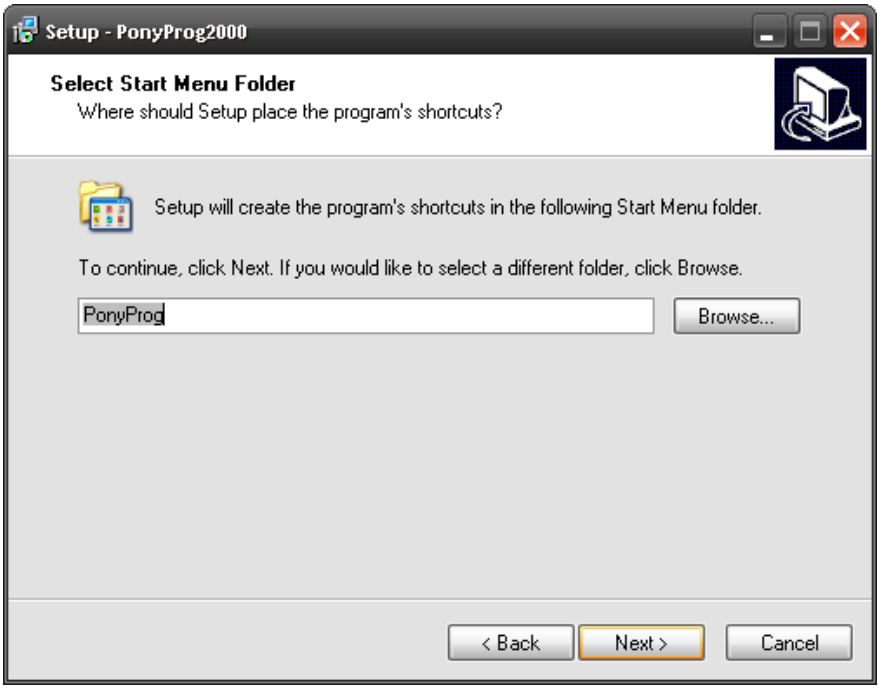
Rysunek 5.26 PonyProg2000 - Okno instalatora

Instalację zaleca się przeprowadzić do domyślnego katalogu: *C:\Program Files\PonyProg2000*



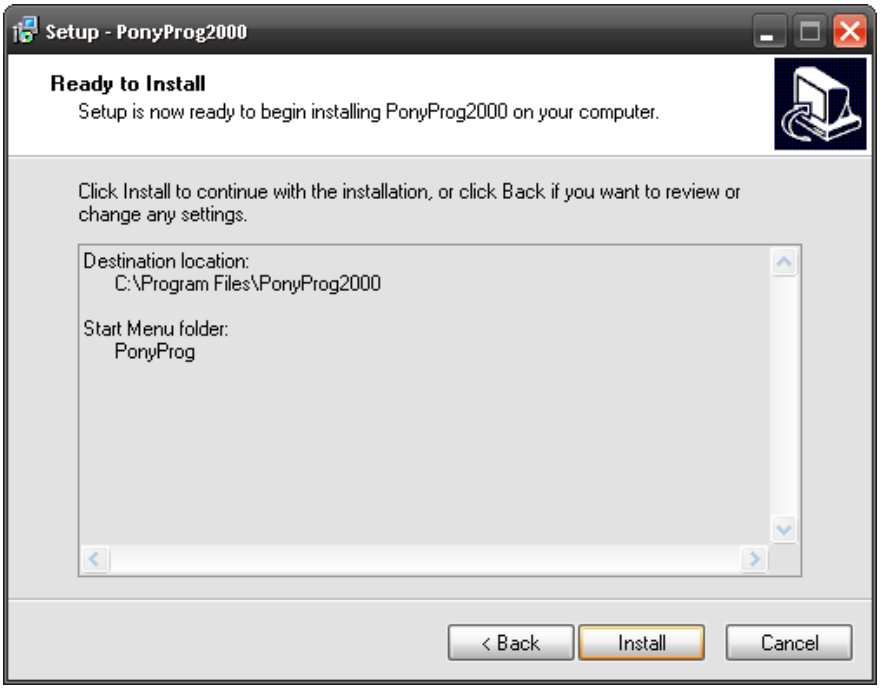
Rysunek 5.27 PonyProg2000 - Wybór katalogu instalacyjnego

Następnie instalator zapyta nas w jakim katalogu *Menu Start* chcemy utowrzyć sktóty do programu (zaleca się pozostawienie domyślnego: *PonyProg*)



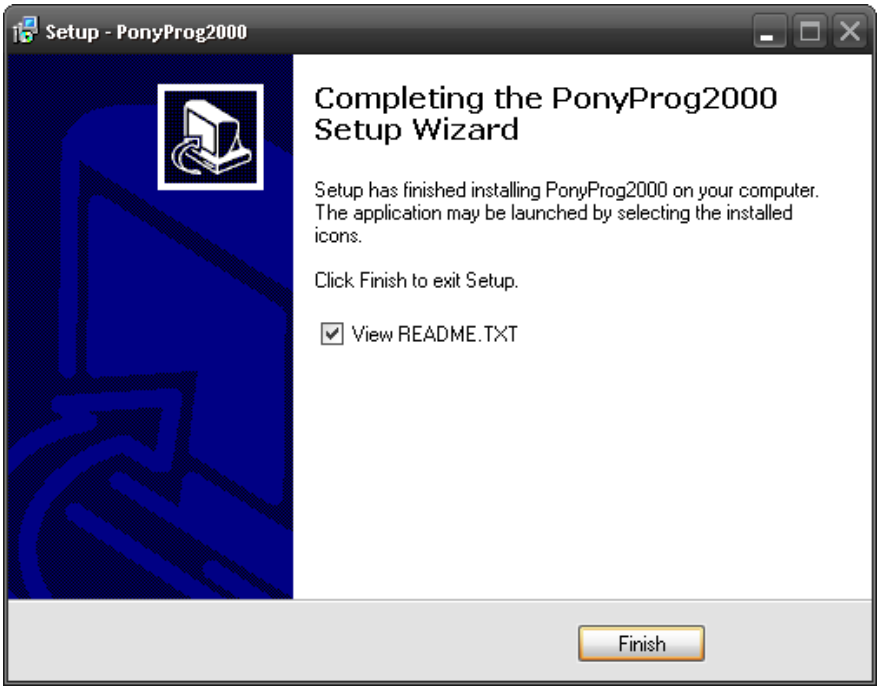
Rysunek 5.28 PonyProg2000 - Wybór katalogu skrótów Menu Start

Instalator zażąda potwierdzenia wybranych opcji instalacji. Po czym przystąpi do instalacji.



Rysunek 5.29 PonyProg2000 - Akceptacja konfiguracji i rozpoczęcie instalacji

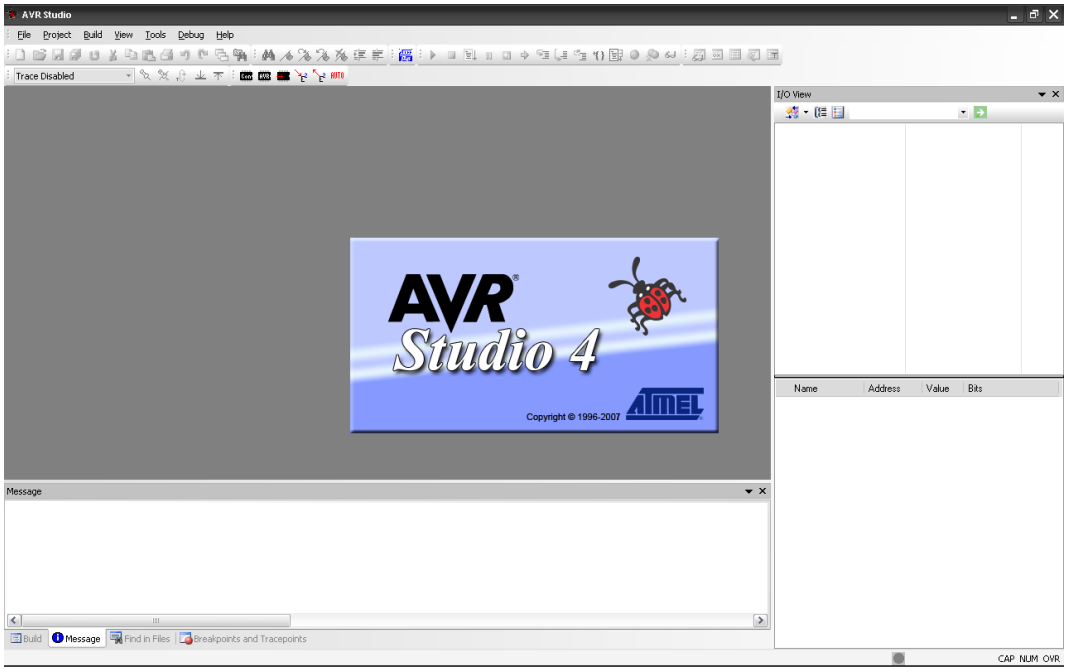
Po zainstalowaniu programu mamy możliwość obejrzenia pliku *readme*.



Rysunek 5.30 PonyProg2000 - Zakończenie instalacji

5.4 Symulacja prostego programu

W tej części zostanie opisany od podstaw sposób symulacji prostych programów na mikrokontrolery ATmega w środowisku AVR Studio. Najpierw zostanie opisana symulacja programu napisanego w języku C++ a następnie omówiony zostanie analogiczny przykład w Assemblerze.



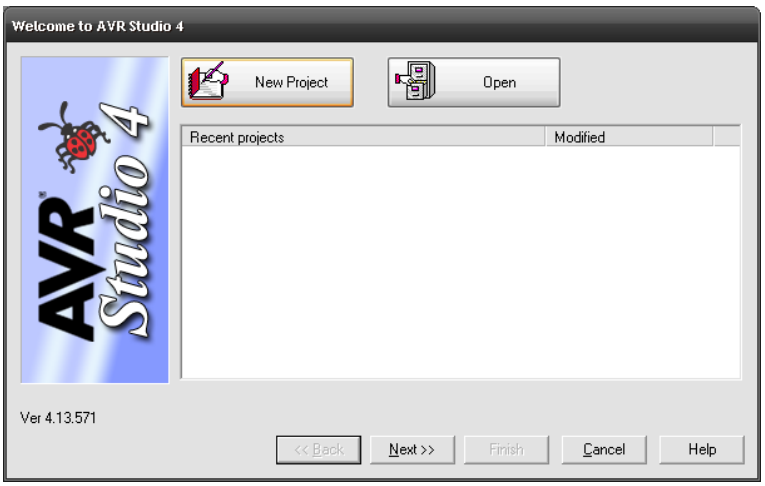
Rysunek 5.31 Uruchamianie AVR Studio 4

5.4.1 Program w C++

Ze względu na dostępność bibliotek *avrlibc* mikrokontrolery AVR są najczęściej programowane w języku C. Rozdział ten opisuje prosty program dla mikrokontorlera AVR napisany w tym właśnie języku - począwszy od stworzenia projektu do jego symulacji.

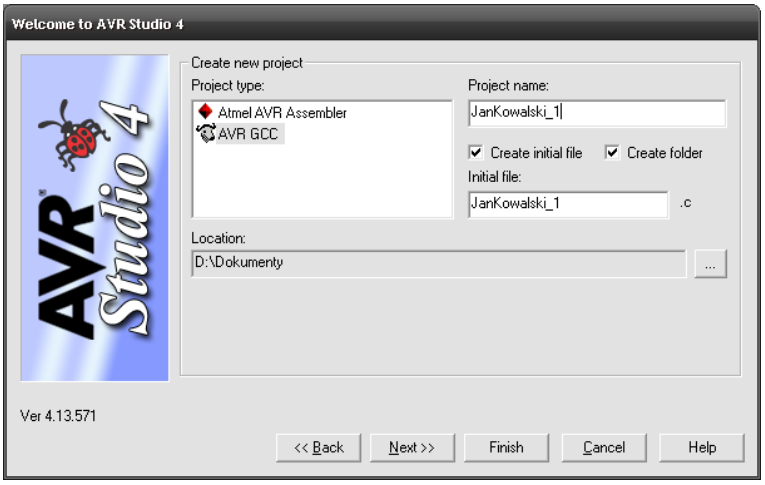
Tworzenie projektu

Pierwszą czynnością jaką należy zrobić po uruchmieniu AVR Studio jest utworzenie projektu. W tym celu z ekranu powitalnego wybieramy *New Project*.



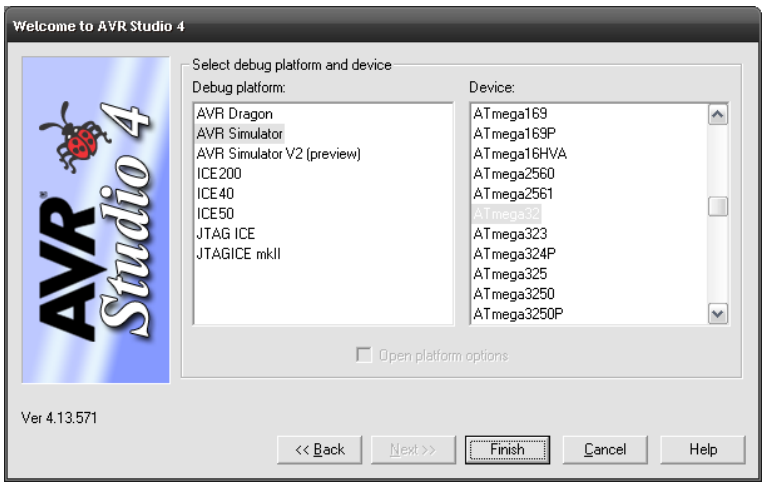
Rysunek 5.32 Utworzenie projektu w AVR Studio 4

Następnie w oknie tworzenia nowego projektu ustawiamy jego typ na: *AVR GCC* wpisujemy jego nazwę (np: *ImięNazwisko_1*) oraz wybieramy ścieżkę zapisu projektu. Patrz rysunek 5.33.



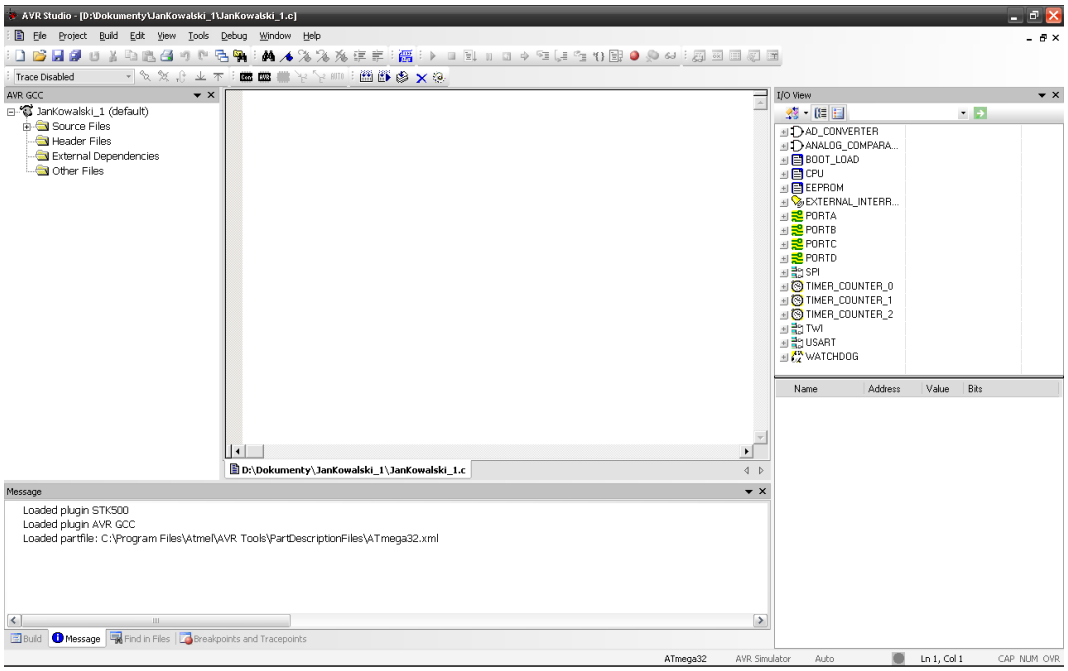
Rysunek 5.33 Konfiguracja nowego projektu w AVR Studio 4

Ostatnią czynnością jest wybór układu docelowego projektu - na tej podstawie przeprowadzana będzie symulacja.



Rysunek 5.34 Konfiguracja symulacji projektu

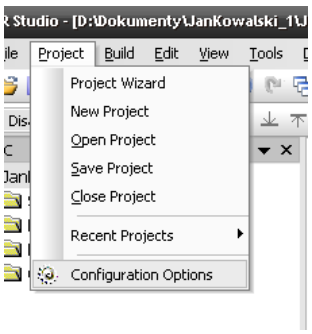
Po wykonaniu powyższych czynności otworzy nam się puste okno projektu.



Rysunek 5.35 Pusty projekt

Konfiguracja

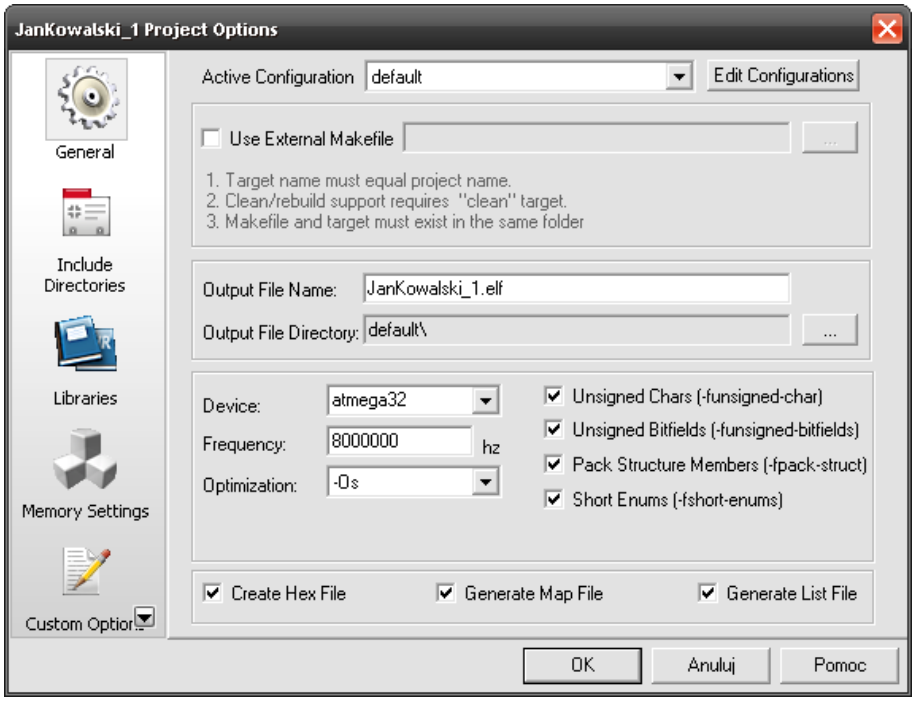
Po utowrzeniu projektu należy przeprowadzić jego konfigurację. W tym celu należy z menu *Project* wybrać opcję *Configuration Options*.



Rysunek 5.36 Menu projektu

W oknie konfiguracji projektu należy ustawić następujące opcje:

- **Device:** atmega32 *(typ układu)*
- **Frequency:** 8000000 *(częstotliwość taktowania)*
- **Optimization:** -Os *(stopień optymalizacji kompilatora)*



Rysunek 5.37 Ustawienia projektu

Programowanie

Poniżej znajuje się przykładowy program dla mikrokontrolera AVR napisany w języku C. Dokumentacja do bibliotek avrlibc znajuje się na oficjalnej stronie projektu[4]. Znaleźć ją można również na załączonym dyku CD w katalogu *\dokumenty\avrlibc\avr-libc-user-manual-1.6.1*

```
1 #include <avr/io.h>          /* biblioteka obslugi wejsc/wyjsc */
2 #include <util/delay.h>     /* bibliioteka procedur opozniajacych */
3
4 int main( void ) {
5
6   /* konfiguracja portu A */
7   PORTA = 0xff;             /* wyslij do portu 0xff */
8   DDRA = 0xff;              /* uzyj portu jako wyjscia */
9
10  /* nieskonczona petla programowa */
11  while(1) {
12
13    /* zaneguj stan wyjsc portu A */
14    PORTA = !PORTA;
15
16    /* procedura opozniajaca
17       zakomentowac w trakcie symulacji */
18    _delay_ms(100);
19
20  }
21
22 }
```

Skonfigurowany i gotowy do uruchmienia projekt zawierajacy powyższy kod można znaleźć na załączonej płycie cd w katalogu *\zrodla\demo1*

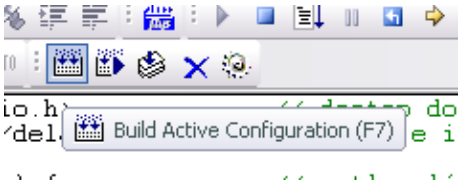
Dla poprawnego przebiegu symulacji powyższego kodu należy zakomentować linijkę

```
18    _delay_ms(100);
```

[4] http://www.nongnu.org/avr-libc/user-manual/

Kompilacja

Po wprowadzeniu kodu należy go skompilować poprzez wybranie ikony budowania projeku na pasku narzędzi (patrz rysunek 5.38), bądź przyciskając klawisz F7.



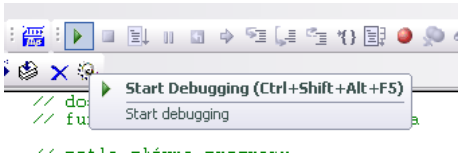
Rysunek 5.38 Kompilacja projektu

Jeżeli kod nie zawiera błędów powinien zostać skompilowany. Informację o błędach, ostrzeżeniach i informacjach kompilatora zawiera okno statusowe znajdujące się w dolnej części ekranu.

Rysunek 5.39 Okno statusowe

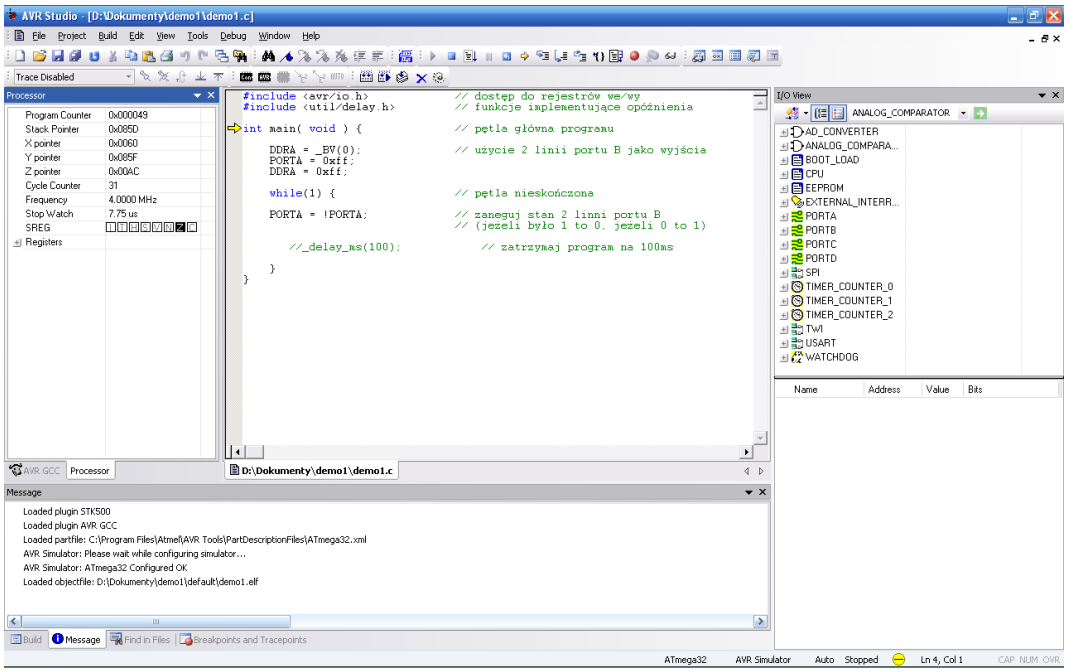
Symulacja/Debuggowanie

Aby przeprowadzić symulację napisanego programu. należy nacisnąć przycisk *"Start Debugging"* znajdujący się na głównym pasku narzędzi (patrz rysunek 5.40).



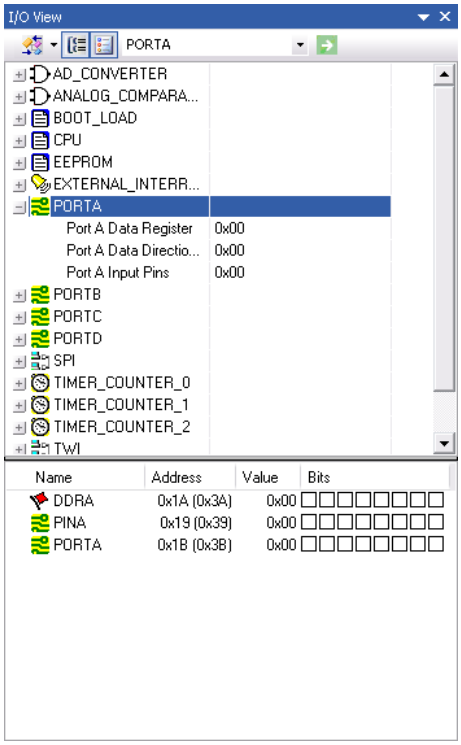
Rysunek 5.40 Uruchomienie Symulacji

Po przejściu w tryb symulacji zostanie wyróżniona główna funkcja programu (funkcja *main()*).



Rysunek 5.41 Interfejs Symulacji

W trybie symulacji, po prawej stronie ekranu widoczny jest podgląd rejestrów mikrokontrolera.



Rysunek 5.42 Okno podglądu urządzeń wewnętrznych

Aby sporuszać się po programie możemy użyć kliku operacji, są to m. in:

- **Step over** - wywołuje kolejną operację i przechodzi do następnej linijki.
- **Step into** - zagłąbia się w kod kolejnej operaji.
- **Go to cursor** - wykonuje program aż dojdzie do zaznaczonej linijki.
- **Breakpoint** - breakpointy ustawiane są na liniach kodu, jeżeli program jest uruchomiony i natrafi na breakpoint, jego praca zostaje wstrzymywana i aplikacja przechodzi w tryb debuggowania.

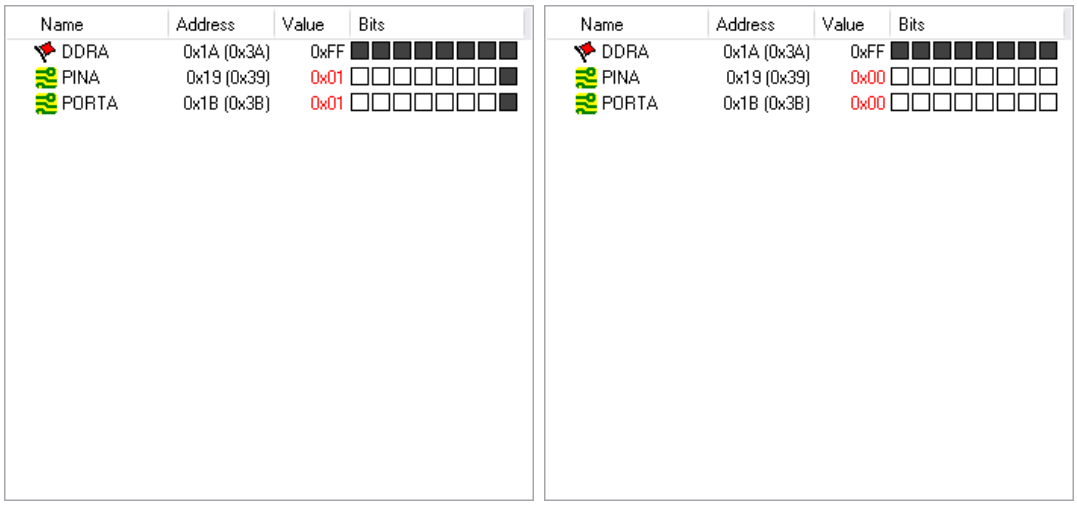
Rysunek 5.43 Przeprowadzanie symulacji 1

Na poniższym przykładzie przedstawiona jest sesja symulacji/debuggowania.

```
#include <avr/io.h>             // dostęp do rejestrów we/wy
#include <util/delay.h>         // funkcje implementujące opóźnienia

int main( void ) {              // pętla główna programu

    DDRA = _BV(0);              // użycie 2 linii portu B jako wyjścia
    PORTA = 0xff;
    DDRA = 0xff;

    while(1) {                  // pętla nieskończona

    PORTA = !PORTA;             // zaneguj stan 2 linni portu B
                                // (jeżeli było 1 to 0, jeżeli 0 to 1)

       //_delay_ms(100);            // zatrzymaj program na 100ms

    }
}
```

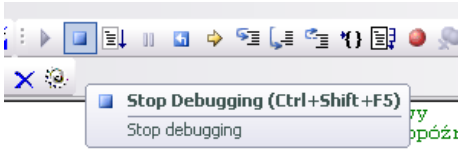
Rysunek 5.44 Przeprowadzanie symulacji 2

Wykonując symulację tego programu powinniśmy zaobserwować zmiany stanu portu wyjściowego A, tak jak pokazuje to rysunek 5.45.



Rysunek 5.45 Podgląd portu A1

Aby zatrzymać symulację/debuggowanie należy wcisnąć przycisk *"Stop Debugging"* z głównego paska narzędzi (patrz rysunek 5.46)



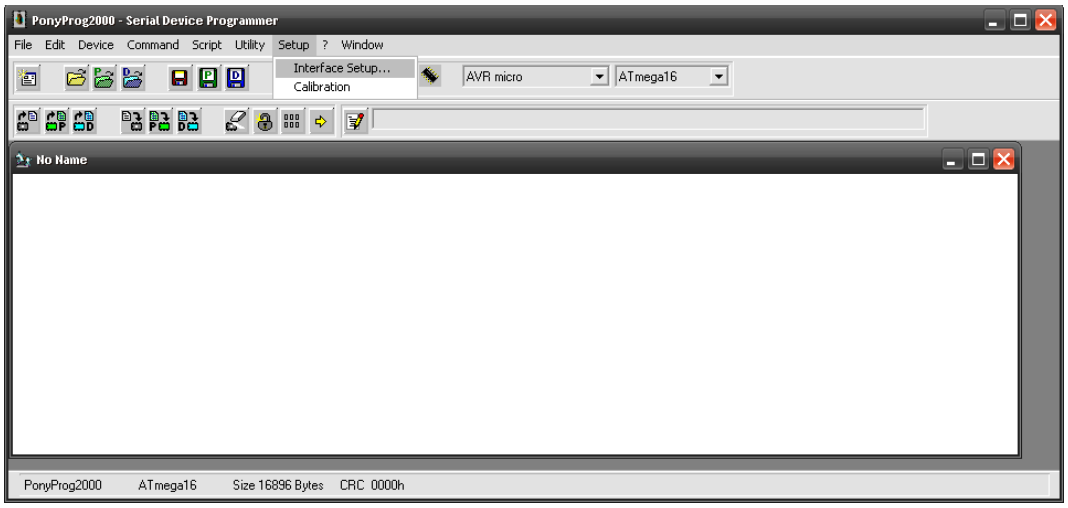
Rysunek 5.46 Zatrzymanie symulacji

Programowanie mikrokontrolera

Aby zaprogramować mikrokontroler należy uruchomić program *PonyProg2000*.

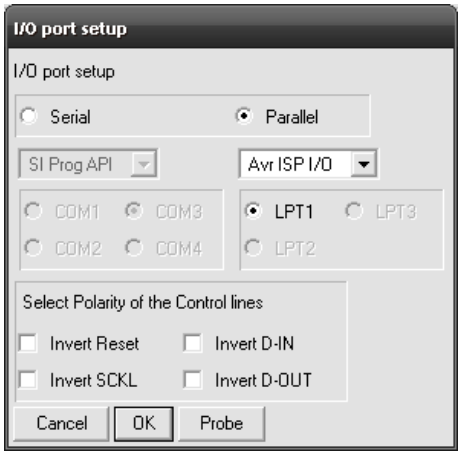
Rysunek 5.47 Uruchamianie programatora PonyProg2000

W głównym oknie programu wybieramy *Interface Setup...* z menu *Setup*



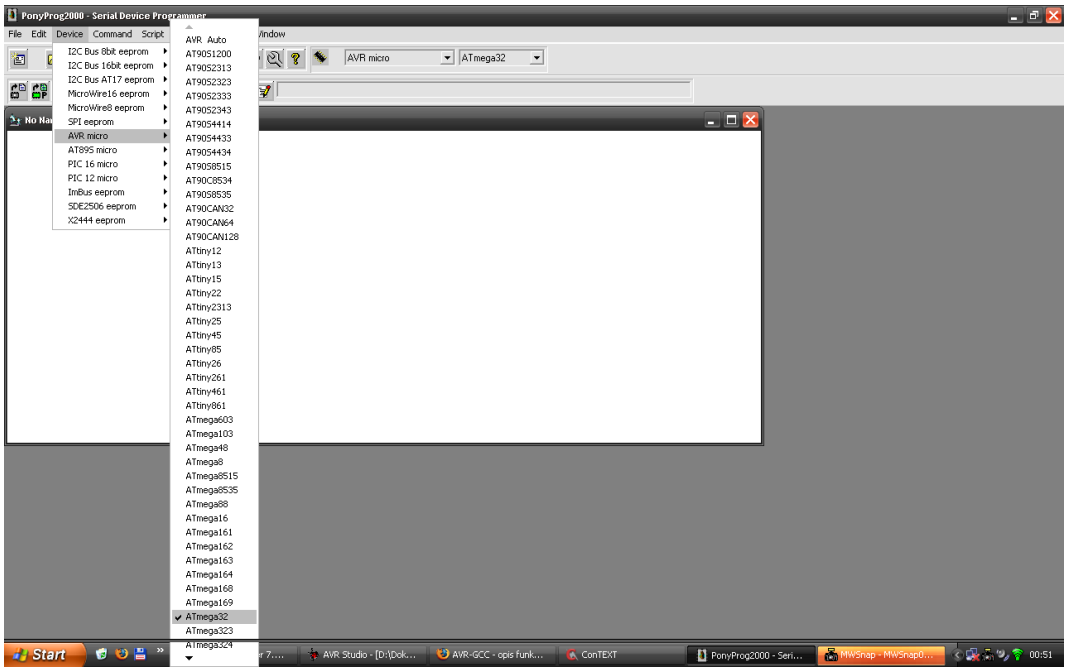
Rysunek 5.48 Główne okno programatora PonyProg2000

W nowo otworzonym oknie konfigurujemy ustawienia programatora (na rysunku 5.49 przedstawione są ustawienia dla programatora STK200)



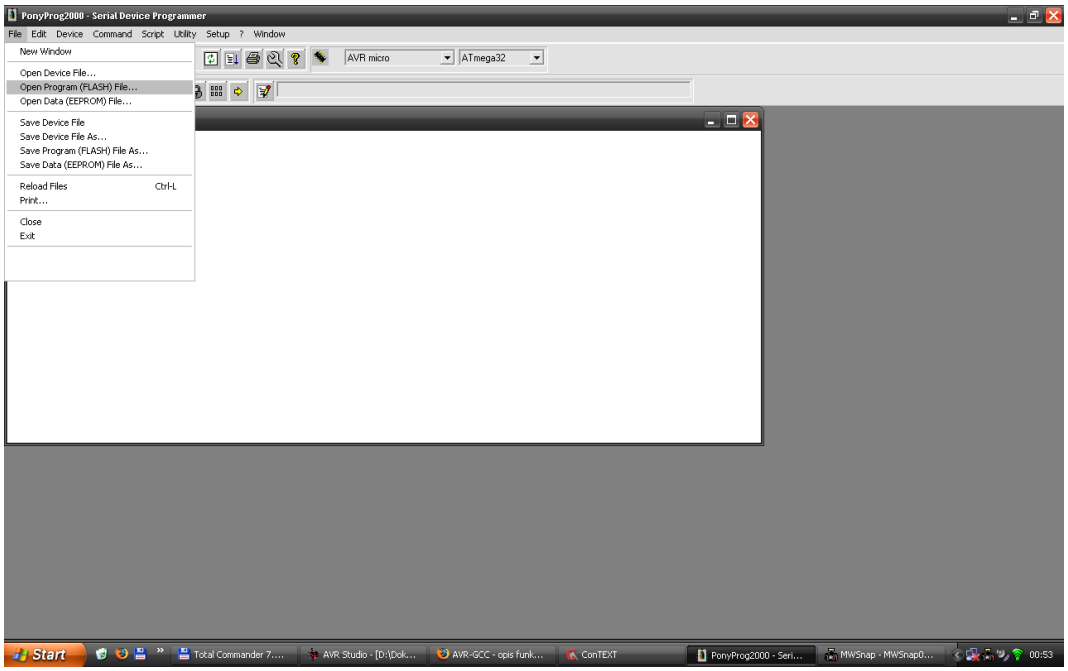
Rysunek 5.49 Konfiguracja programatora

Z menu *Device* wybieramy odpowiedni typ mikrokontrolera.



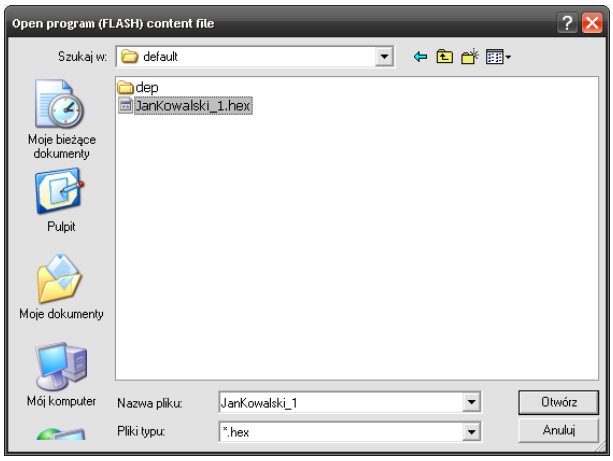
Rysunek 5.50 Wybór mikrokontrolera

Korzystając z opcji *Open Program (FLASH) File...* z menu *File* otwieramy plik do zaprogramowania.



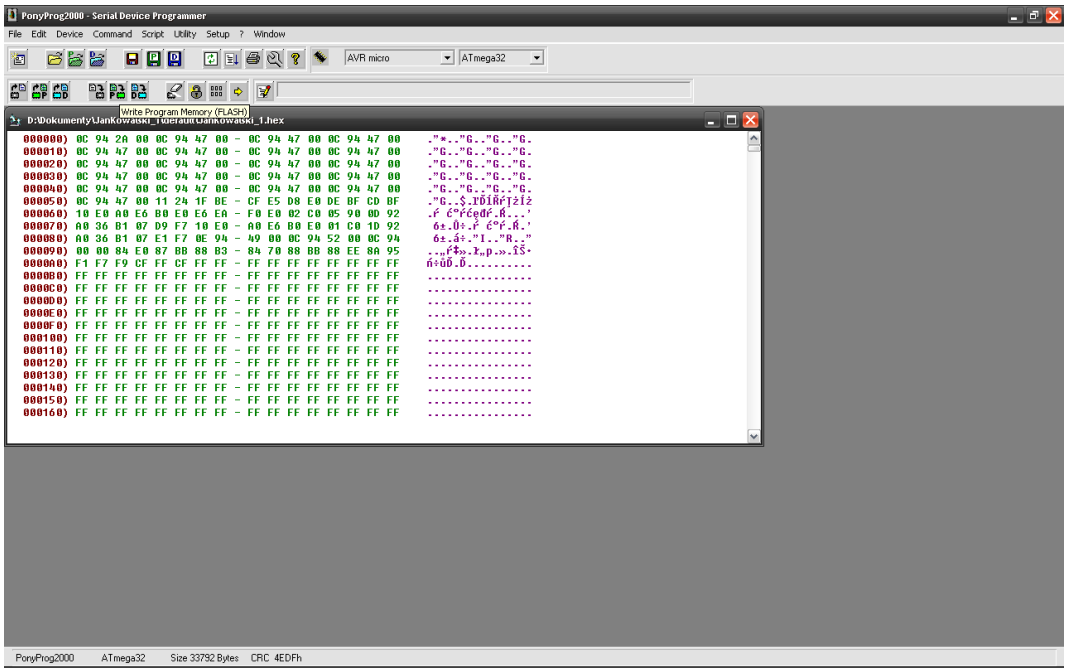
Rysunek 5.51 Otwarcie pliku 1

Przechodzimy do katalogu projektu AVRStudio i wybieramy plik o rozszerzeniu *.hex* i nazwie takiej jak projekt.



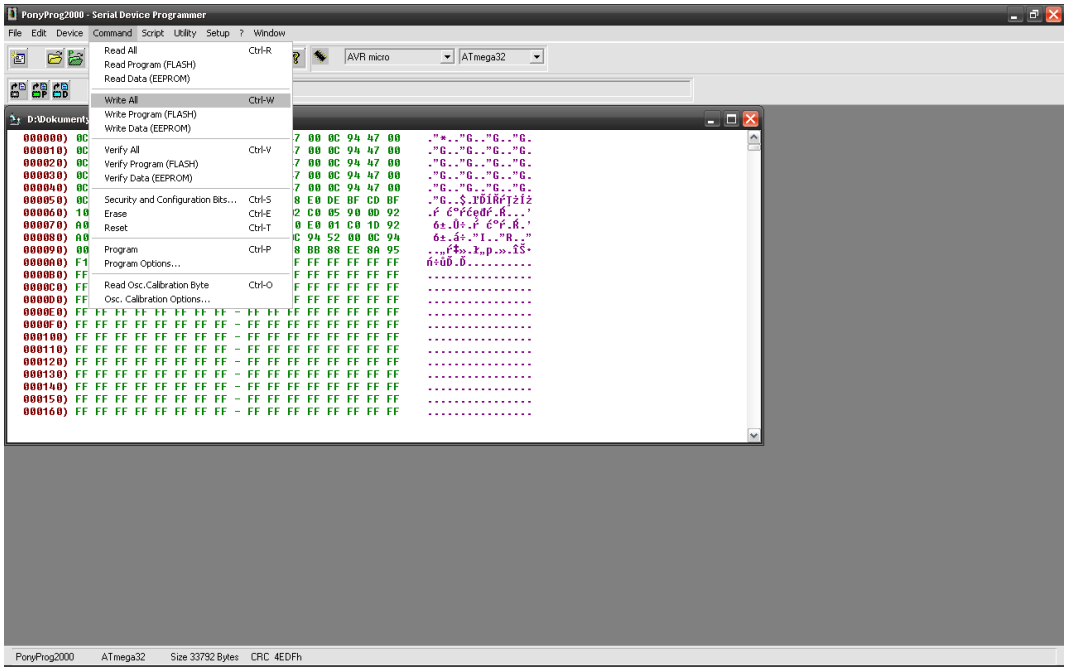
Rysunek 5.52 Otwarcie pliku 2

Zostanie otworzony program.



Rysunek 5.53 Otwarcie pliku 3

Aby skopiować go do mikrokontrolera wybieramy polecenie *Write All...* z menu *Command*.



Rysunek 5.54 Zapisywanie do urządzenia

Rozdział 6

Przetwornik pomiarowy

6.1 Opis układu

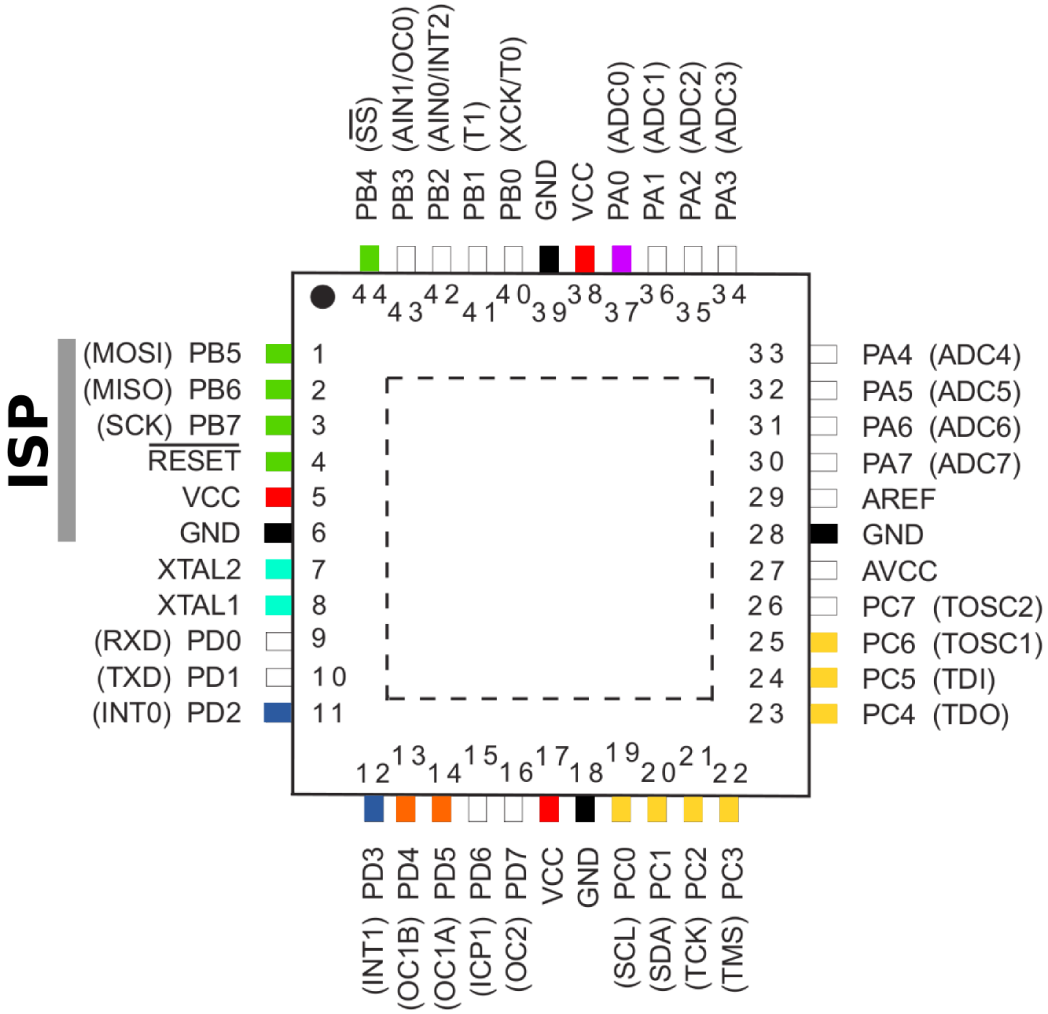
Głównym celem pracy jest skonstruowanie układu do przetwarzania wartości pomiarowej w standardzie prądowym $4-20mA$ do wartości w standardzie częstotliwościowym $1000-5000Hz$. Schemat działania układu przedstawiony jest na poniższym diagramie 6.1

Rysunek 6.1 Schemat działania układu

Sygnał przychodzący z czunika poziomu cieczy podany będzie na 10-bitowy przetwornik analogowy-cyfrowy znajdujący się wewnątrz mikrokontolera AVR. Sygnał przetwarzany jest do zmiennej wewnętrznej, która jest odpowiednio przetwarzna, tak aby na jej podstawie można było wygenerować sygnał wyjściowy (analogowy, PWM, częstotliwościowy).

6.2 Budowa układu

Układ oparty jest na mikroprocesorze AVR ATmega16 Do budowy układu wykorzystano większość z dostępnych w jego wnętrzu urządzeń. Rozmieszczenie połączeń do mikrokontrolera przedstawione jest na rysunku 6.2. Wykaz zastosowanych oznaczeń (kolorów) przedstawiony jest w tabeli 6.1.



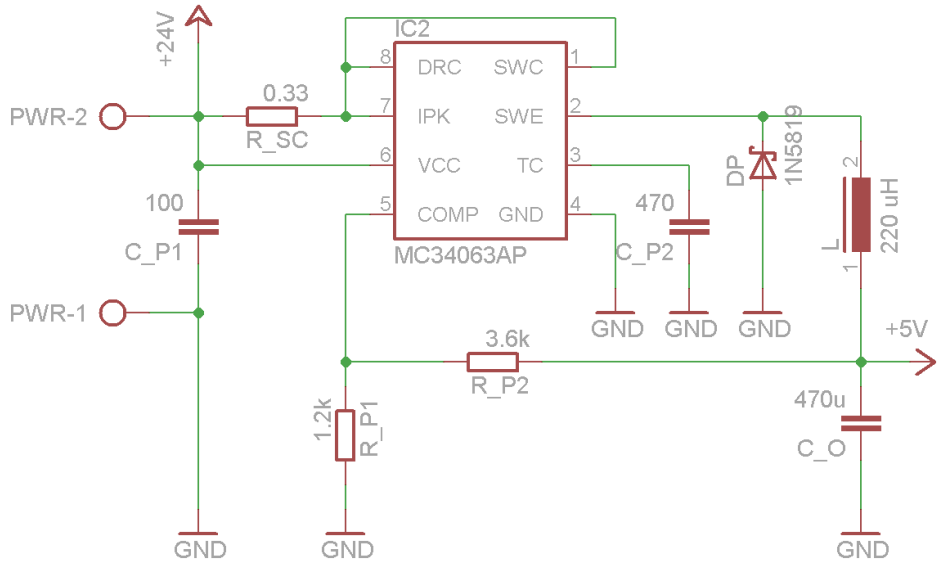
Rysunek 6.2 Wykaz połączeń

Tabela. 6.1 wykaz oznaczeń połączeń

kolor	opis
	Napięcie zasilania (VCC)
	Masa układu (GND)
	Wejścia cyfrowe
	Klawiatura
	Zewnętrzny oscylator kwarcowy
	Wejście analogowe
	Wyjście cyforowe / analogowe
	Wyświetlacz LCD

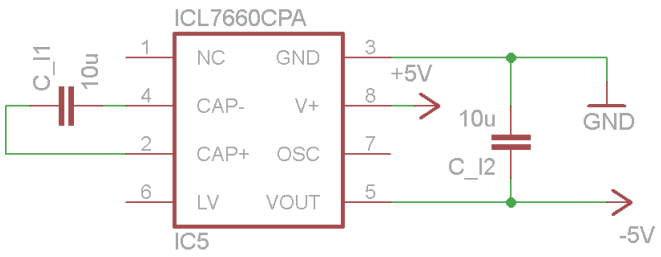
6.2.1 Obwód zasilania

Obwód zasilania układu składa się z dwóch elementów, pierwszy z nich stanowi przetwornica napięcia zasilającego (24/5V) jej schemat ideowy przedstawiony jest na rysunku 6.3. Zbudowana jest ona w oparciu o układ regulatora **MC34063A** firmy ON Semiconductors. Zakres napięć wejściowych może wachać się od 21 do 27 V. Przetwornica generuje napięcie 5V o maksymalnym natężeniu 500 mA (max 1.5A z wykorzystaniem dodatkowego tranzystora przełączającego).



Rysunek 6.3 Przetwornica zasilająca

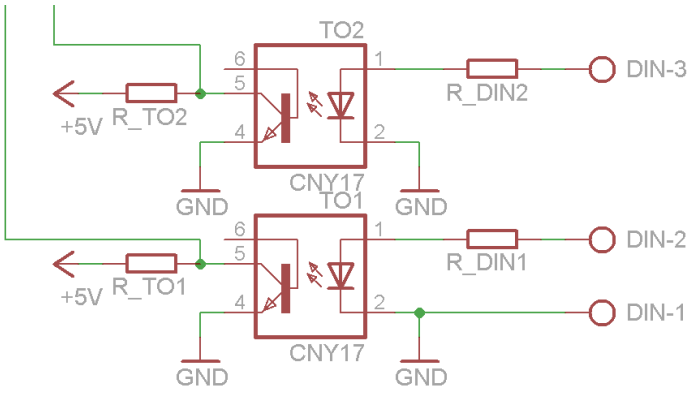
Drugim elementem obwodu zasilania jest inwerter napięcia użyty dla poprawy pracy wzmacniaczy operacyjnych zastosowanych w obwodach wyjściowych. Inwerterem napięcia jest układ scalony ICL7660S. Schemat ideowy układu inwertera napięcia przedstawiony jest na rysunku 6.4.



Rysunek 6.4 Inwerter napięcia

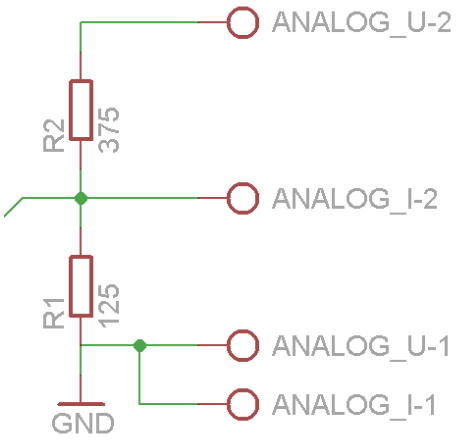
6.2.2 Obwody wejściowe

Obwód wejść cyforwych zbudowany jest w oparciu o optoizolację, która zapewnia również przełożenie napięcia. W roli transoptora zastosowano układ CNY17. Odpowiedni dobór rezystorów R_DIN i R_TO pozwolił na uzyskanie przełożenia napięcia ze standardu przemysłowego na napięcie, operacyjne mikrokontolera. Schemat ideowy obwodów wejść cyfrowych znajduje się na rysunku 6.5.



Rysunek 6.5 Obwód wejść cyfrowych

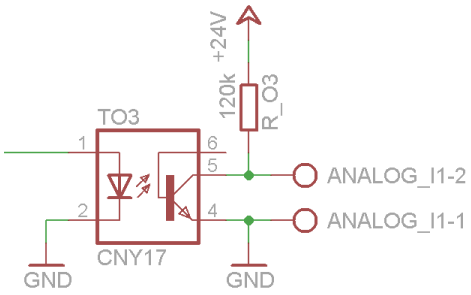
Obwód wejść analogowych zbudowany jest w oparciu o dzielnik napięciowy oraz napięciowy próbnik prądu. Schemat ideowy obwodów wejść analogowych znajduje się na rysunku 6.6.



Rysunek 6.6 Obwód wejść analogowych

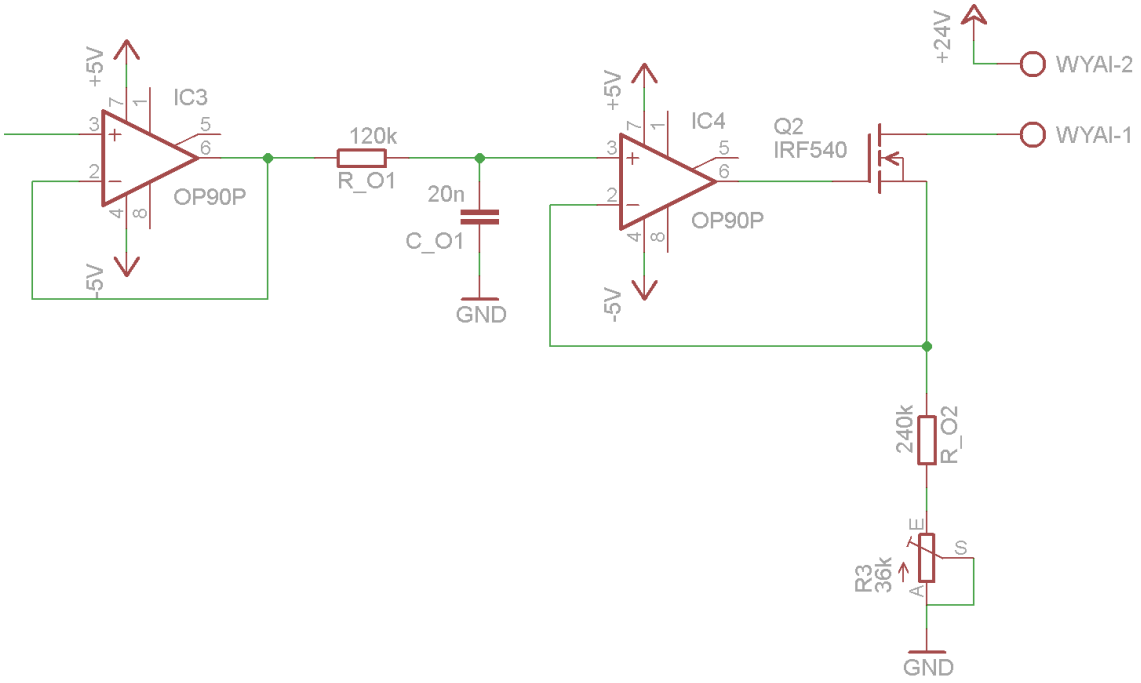
6.2.3 Obwody wyjściowe

Obwód wyjścia cyfrowego zbudowany jest w oparciu o transoptor CNY17 zapewniający przełożenie napięcia oraz izolację optyczną obwodów mikorokontorlera i obwodów roboczych. Schemat ideowy układu wyjścia cyforwego przedstawiony jest na rysunku 6.7.



Rysunek 6.7 Obwód wyjść cyfrowych

Budowa wyjścia analogowego oparta jest na generatorze PWM mikrokontrolera. Układ wyjściowy jest trójstopniowy. Pierwszy stopień stanowi wzmacniacz operacyjny w układzie wtórnika napięciowego dla stabilizacji wyjścia mikrokontrolera. Kolejnym stopniem jest dolnoprzepustowy filtr RC, służący do uśrednienia wartości generowanej przez modulator PWM. Trzeci stopień stanowi przetwornik napięcie na prąd zbudowany w oparciu o wzmacniacz operacyjny i tranzystor IRF540. Potencjometr R3 służy do korekcji przetwarzania napięcia na prąd. Schemat ideowy układy przedstawiony jesty na poniższym rysunku 6.8.



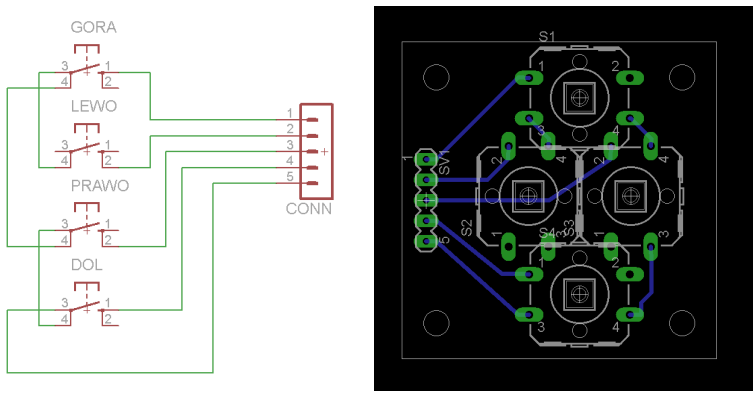
Rysunek 6.8 Obwód wyjść analogowych

6.3 Program

Kod programu przetwornika znajduje się w załcznikach do pracy.

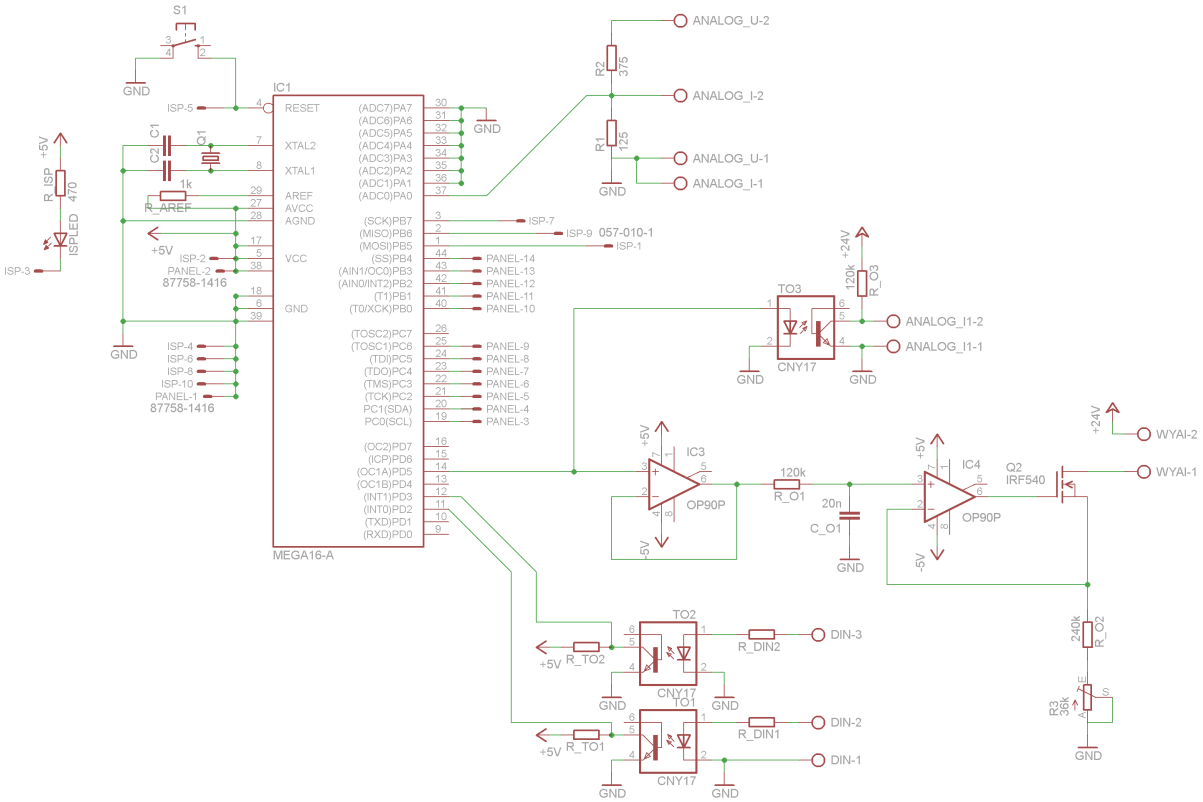
6.4 Realizacja układu

Realizacja układu sprowadzała się do opracowania wzorów płytek drukowanych panela użytkownika i płyty głownej urządzenia.

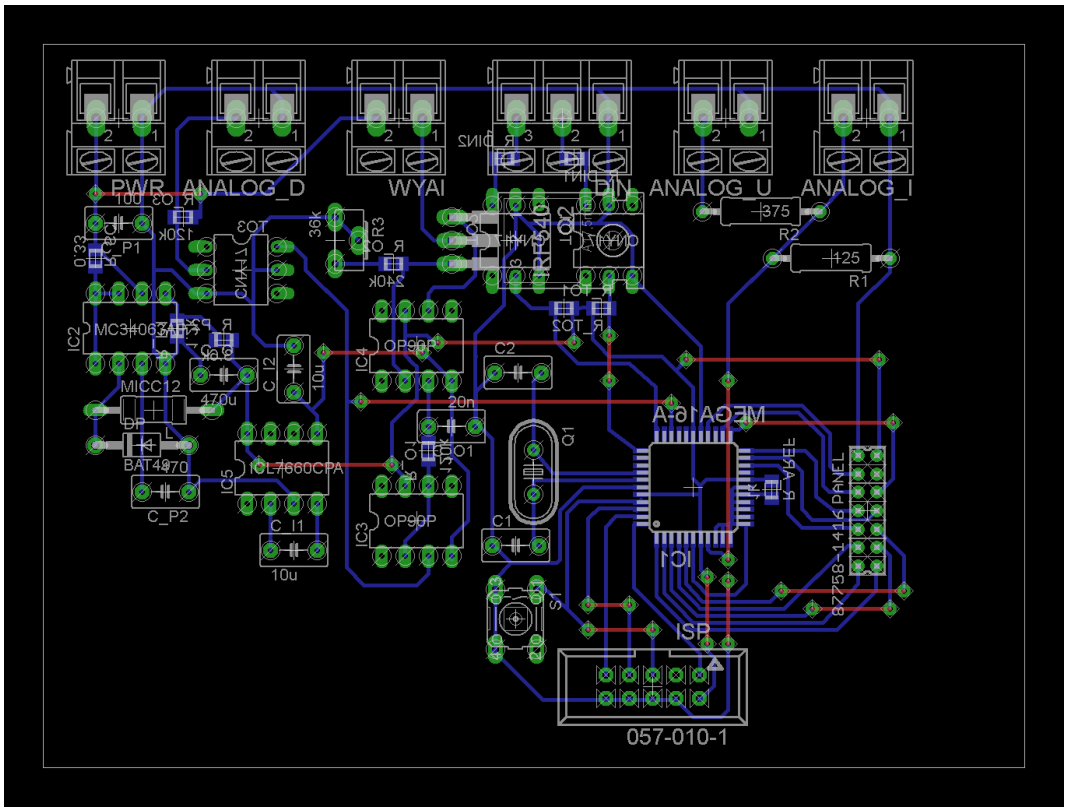


(a) Schemat Ideowy (b) Wzór płytki

Rysunek 6.9 Obwody panela



Rysunek 6.10 Schamat ideowy płyty głównej



Rysunek 6.11 Wzrór obwodu drukowanego płyty głownej

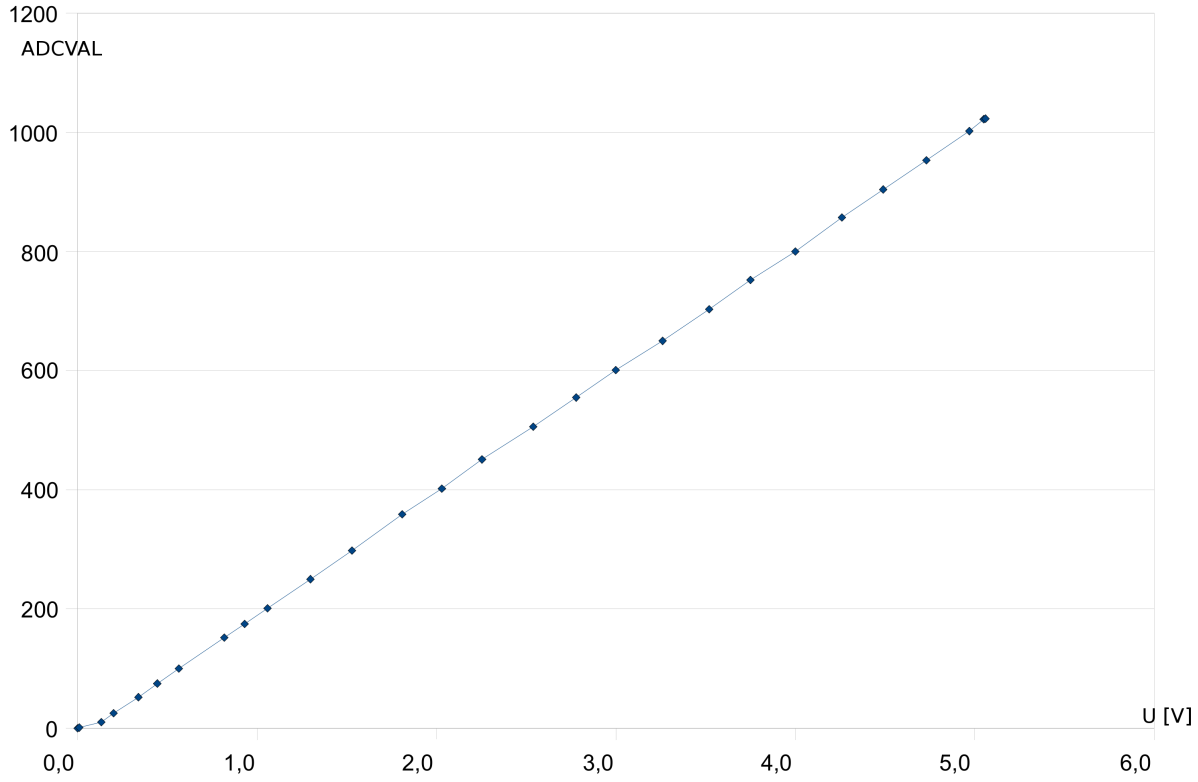
Do układu przygotowana została również obudowana szynę DIN.

Rysunek 6.12 Obuda układu

6.5 Charakterystyki układu

6.5.1 Badanie toru pomiarowego

W celu sprawdzenia poprawności przetwarzania przetwornika analogowo-cyfrowego przeprowadzono badanie toru pomiarowego. Na wejście przetwornika podawano napięcie i odczytywano wartość zmiennej wewnętrznej mikroprocesora reprezentującą mierzoną wielkość. Wyniki tego badania przedstawione są na wykresie 6.13.

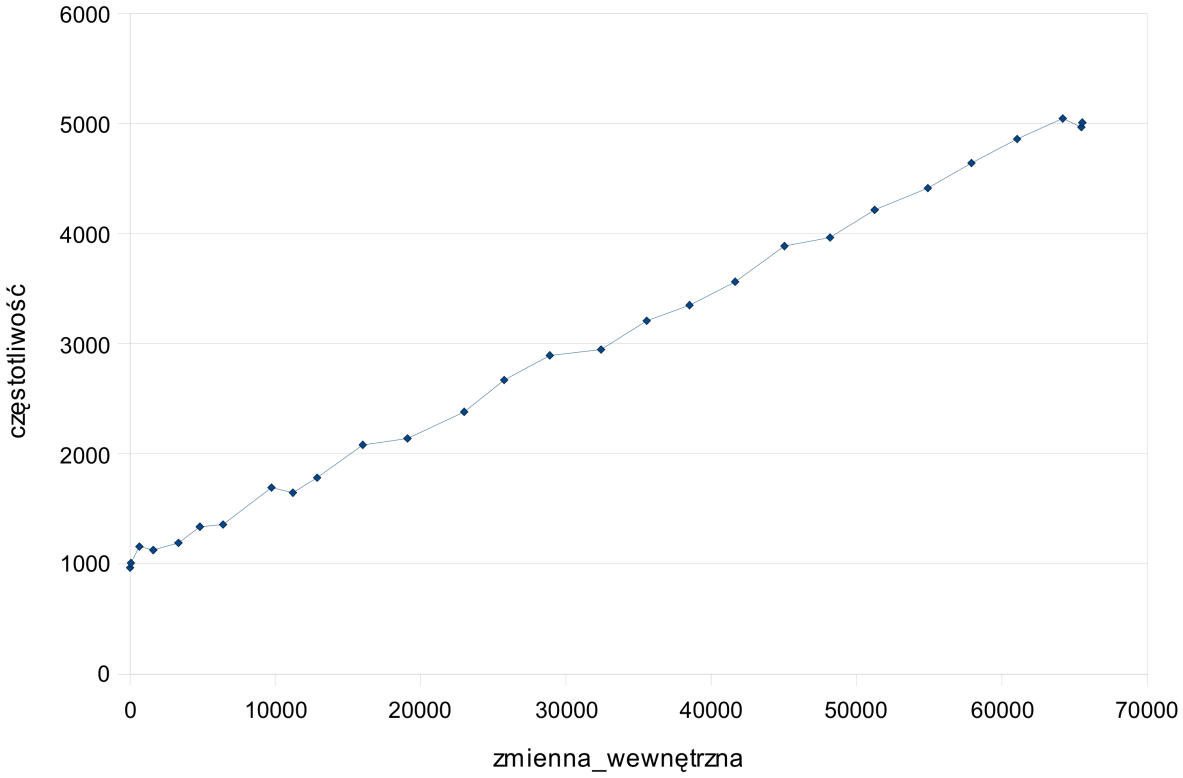


Rysunek 6.13 Charakterystyka toru pomiarowego

Uzyskana charakterystyka wykazuje liniowość przetwarzania w pełnym zakresie pomiarowym.

6.5.2 Badanie modulatora częstotliwości

W celu sprawdzenia poprawności przetwarzania toru wyjściowego, przeprowadzono badanie modulatora częstotliwości. Zmienną wewnętrzną mikroprocesora odpowiadającą za częstotliwość sygnału wyjściowego zmieniano w płenym jej zakresie i odczytywano częstotliwość sygnału wyjściowego. Wyniki tego badania przedstawione są na wykresie 6.14.



Rysunek 6.14 Charakterystyka toru pomiarowego

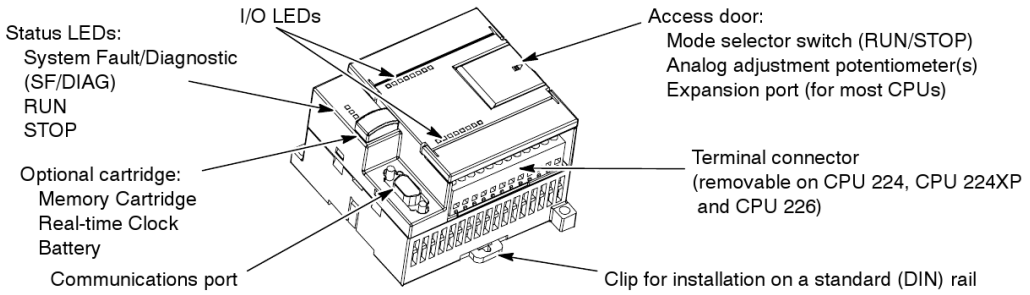
Uzyskana charakterystyka wykazuje liniowość przetwarzania w pełnym zakresie pomiarowym.

Rozdział 7

Sterowanie poziomem cieczy

Automatyzacja sterowania poziomem cieczy wykonana będzie na sterowniku Siemens SIMATIC S7-200. W sterowniku zostaną zaimplementowane algorytmy sterowania dwu- i trójstawnego.

7.1 Sterownik SIMATIC S7-200



Rysunek 7.1 Szkic sterownika[14]

Seria sterowników SIMATIC S7-200 jest rodziną urządzeń należących do tzw. grupy mikrosterowników. Ma ona zastosowanie w różnorodnych systemach automatyki. Programy dla sterowników z tej serii mogą zawierać logikę binarną, liczniki, funkcje czasowe, skomplikowane operacje matematyczne, a także obsługę komunikacji z innymi urządzeniami. Jednostki podstawowe sterowników nie zawierają modułów wejść ani wyjść analogowych (wyjątek stanowi *CPU 224XP* posiadający 2 wejścia i 1 wyjście analogowe), zawierają jednak szybkie liczniki za pomocą których można odczytywać wartości analogowe.

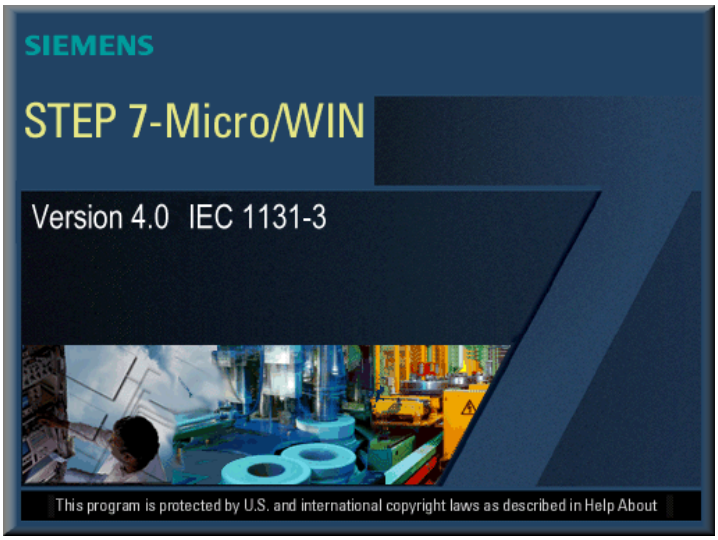
Komunikacja komputera ze sterownikiem może odbywać się w dwojaki sposób. Pierwszy z nich (najpopularniejszy i zarazem ekonomiczny) to połączenie poprzez kabel PPI Multi-Master. Łączy on port komunikacyjny sterownika z portem szeregowym komputera (RS232). Rzadziej spotykanym rozwiązaniem jest użycie protokołu MPI, ponieważ wymaga ono zastosowania specjalnej karty rozszerzeń (interfejsu MPI) dla komputera PC.

Mikrosterowniki S7-200 umożliwiają także obsługę procesów za pomocą paneli wyświetlających np. *TD 200* lub *TP 070*.

7.2 Oprogramowanie narzędziowe STEP 7-Micro/WIN

Programowanie sterowników serii SIMATIC odbywa się za pomocą pakietu narzędziowego STEP. Dla serii mikrosterowników S7-200 została przygotowana specjalna wersja tej aplikacji o nazwie STEP 7-Micro/WIN. Minimalne wymagania sprzętowe potrzebne do wydajnej pracy:

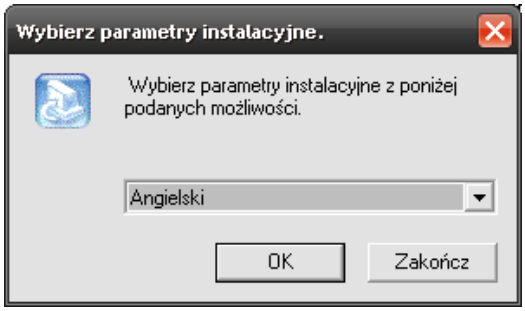
- system operacyjny Windows 2000 lub XP (do instalacji konieczne są uprawnienia administratora)
- co najmniej 100 MB wolnej przestrzeni dyskowej
- mysz



Rysunek 7.2 STEP 7-Micro/WIN

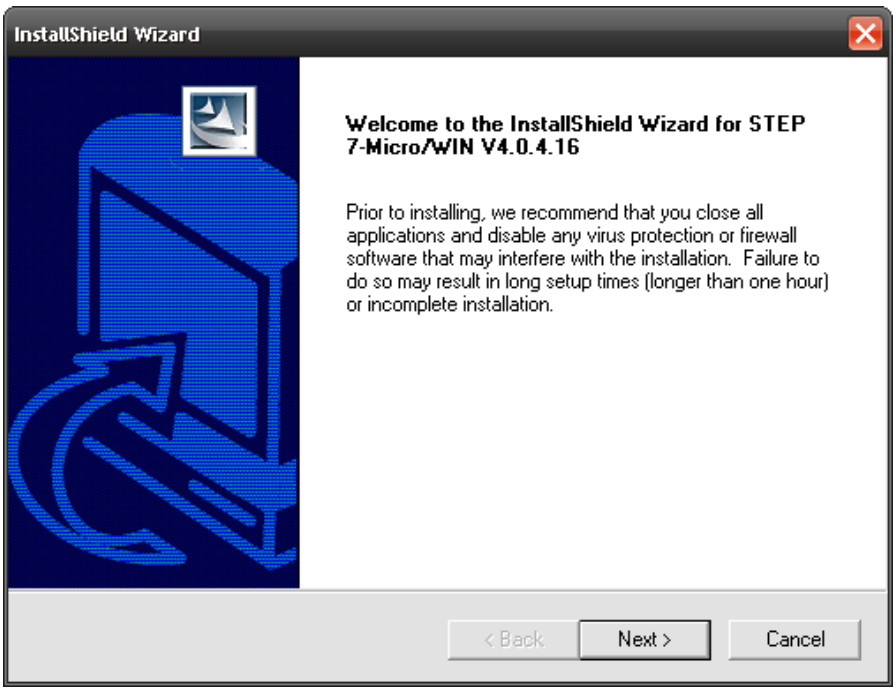
7.2.1 Instalacja

Aby zainstalować program STEP 7-Micro/WIN należy włożyć płytę instalacyjną do napędu CD. Kreator instalacji wystartuje automatycznie prosząc nas o wybór języka instalacji (rysunek 7.3).



Rysunek 7.3 Okno wyboru języka

Przed rozpoczęciem, instalator poprosi nas o zamknięcie wszystkich programów działających w "tle" oraz wyłączenie programów antywirusowych i firewalli, które mogły by przeszkodzić w procesie instalacji.

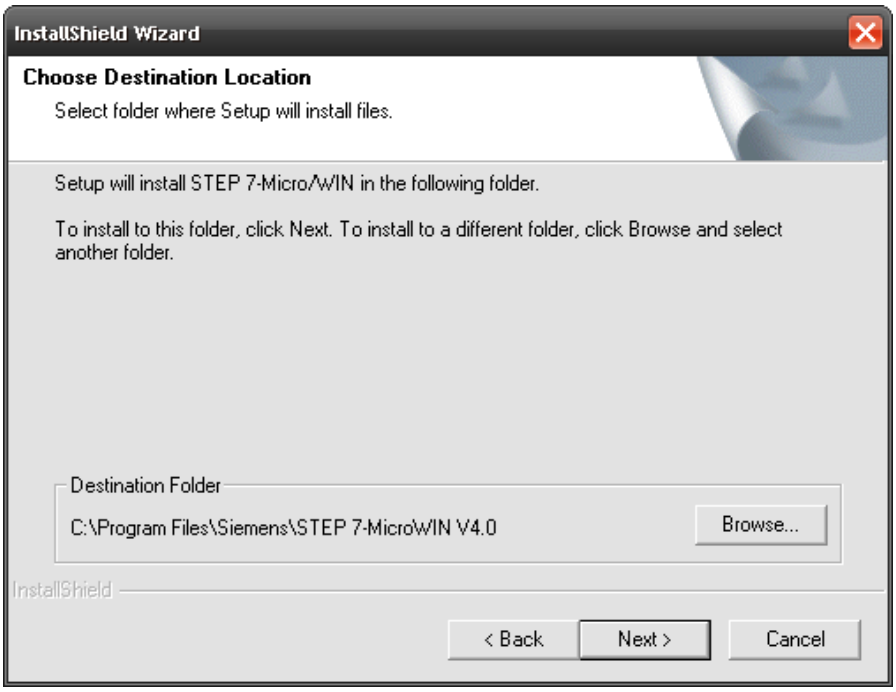


Rysunek 7.4 Okno Instalatora

Aby kontynuować należy przeczytać i zaakceptować licencję oprogramowania.

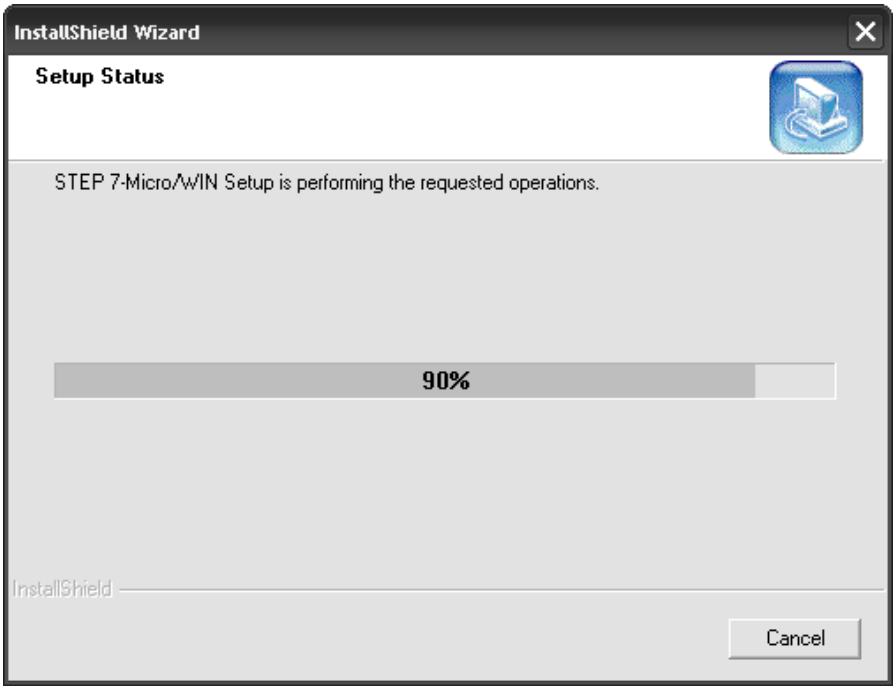
Rysunek 7.5 Akceptacja licencji

Następnie zostaniemy poproszeni o wybór folderu docelowego.



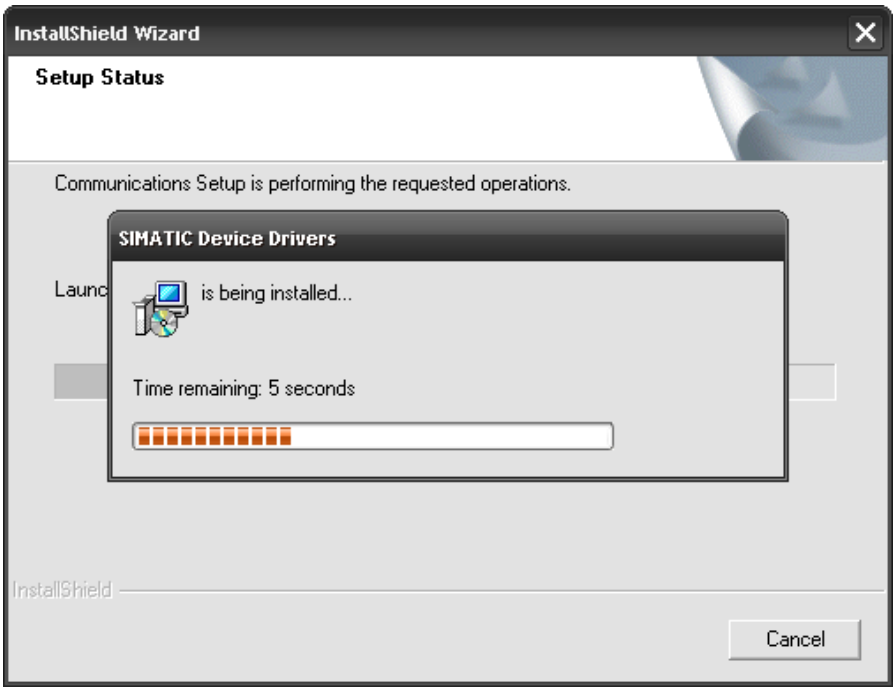
Rysunek 7.6 Wybór katalogu instalacji

Rozpocznie się instalacja oprogramowania.



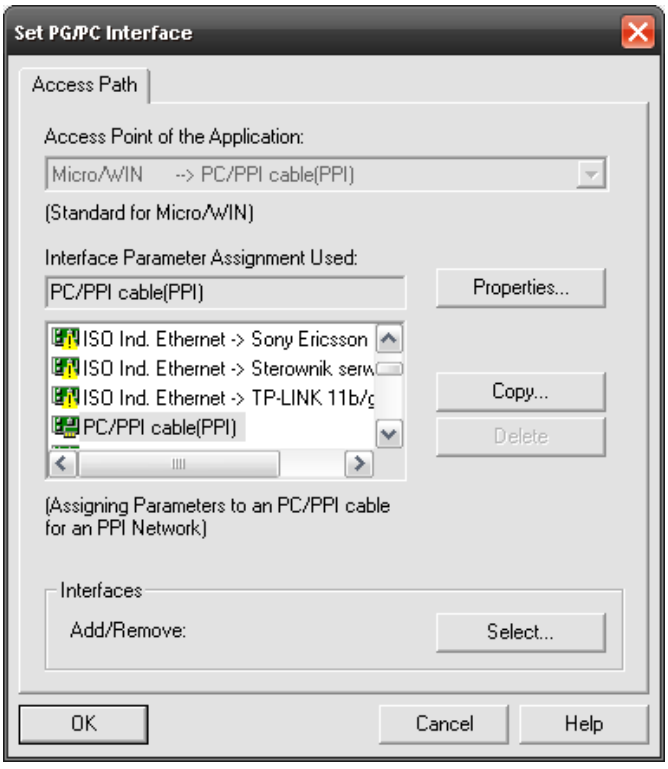
Rysunek 7.7 Proces instalacji STEP 7-Micro/WIN

Program zainstaluje także podstawowe sterowniki dostępu do urządzeń SIMATIC.



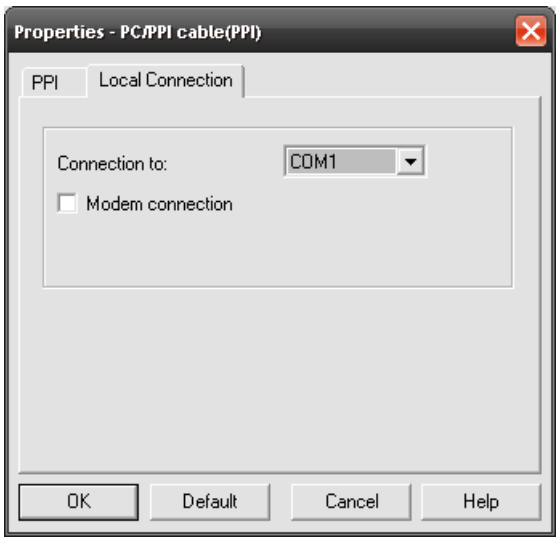
Rysunek 7.8 Instalacja sterowników

Po instalacji sterowników zostaniemy poproszeni o konfigurację interfejsu dostępowego do urządzeń. Z listy dostępnych wybieramy "*PC/PPI Cable(PPI)*" i naciskamy przycisk ***Properties...***.



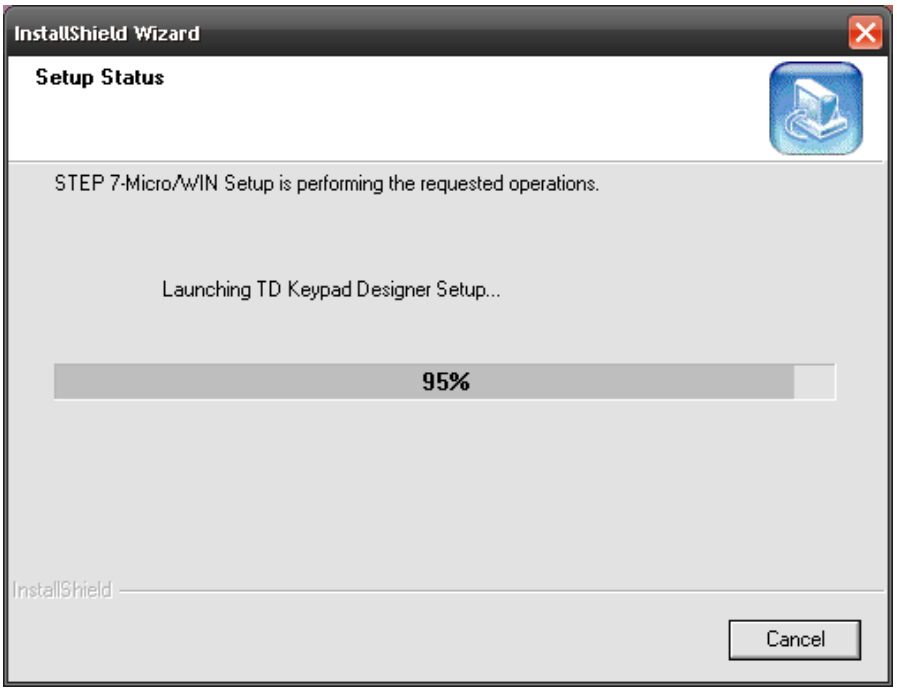
Rysunek 7.9 Konfiguracja interfejsu PG/PC

W zakładce ***Local Connection*** dokonujemy wyboru portu, do którego podłączony jest kabel interfejsu RS232/PPI (najczęściej będzie to port **COM1**).



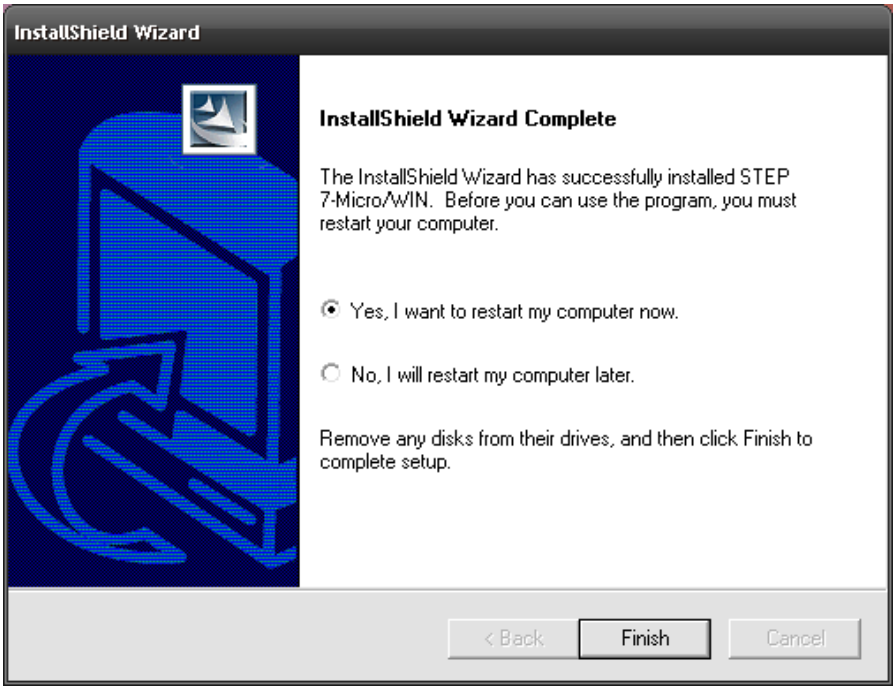
Rysunek 7.10 Ustawienia połączenia PC/PPI

po skonfigurowaniu interfejsu instalacja zostanie wznowiona.



Rysunek 7.11 Finalizacja instalacji

Po zakończeniu instalator zasugeruje ponowne uruchomienie komputera (zalecane).



Rysunek 7.12 Ponowne uruchomienie komputera

7.3 Programowanie sterownika

Znanych jest co najmniej 6 różnych języków programowania sterowników PLC (lista instrukcji, schemat stykowy, schemat funkcji, język przebiegu, tekst zhierarchizowany, graficzny edytor schematów funkcji), narzędzia programistyczne sterownika S7-200 umożliwiają programowanie w trzech z nich. Są to:

- LAD (ang. Ladder diagram) czyli schemat stykowy (zwany drabinką) to najpopularniejszy język programowania sterowników PLC. Program wyświetlany jest w postaci graficznej, podobnej do elektrycznego schematu połączeń. Jego logika jest prosta nawet dla początkujących. Graficzna interpretacja jest intuicyjna i popularna na całym świecie. Logika programu zorganizowana jest w blokach zwanych sieciami[1]
- FBD (ang. Function block diagram) to schemat funkcji podobny do języka LAD, połączenia także są zorientowane, jednak tutaj bloki, z których buduje się program przedstawiają bramki logiczne a nie elementy elektryczne. Język ten jest wygodny do tworzenia diagramów przepływu.
- STL (ang. Statement list) zwany listą instrukcji jest zalecany tylko dla zaawansowanych programistów. Umożliwia pisanie programu jako zestaw niskopoziomowych instrukcji do wykonywanych przez sterownik (podobnie do assemblera). Jest on bardziej elastyczny od poprzednich dwóch i umożliwia rozwiązanie problemów na które tamte nie pozwalają.

Program STEP 7-Micro/WIN pozwala na konwersję pomiędzy tymi językami, ale nie zawsze jest ona możliwa.

Szczegółowe informacje na temat programowania sterowników można znaleźć w ich dokumentacji. Producenci często również udostępniają darmowe podręczniki umożliwiające zapoznanie się z najprostszymi funkcjami sterowników i przykładowymi programami. Angielska wersja instrukcji do sterowników S7-200 zamieszczona jest na dołączonej płycie CD w pliku *\dokumenty\Siemens\S7-200_ en.pdf* Polską wersję można nieodpłatnie otrzymać od lokalnego dystrybutora produktów SIMATIC.

Omawiane w tej części programy znajdują się na załączonej płycie CD w katalogu *\projekty\S7-200*

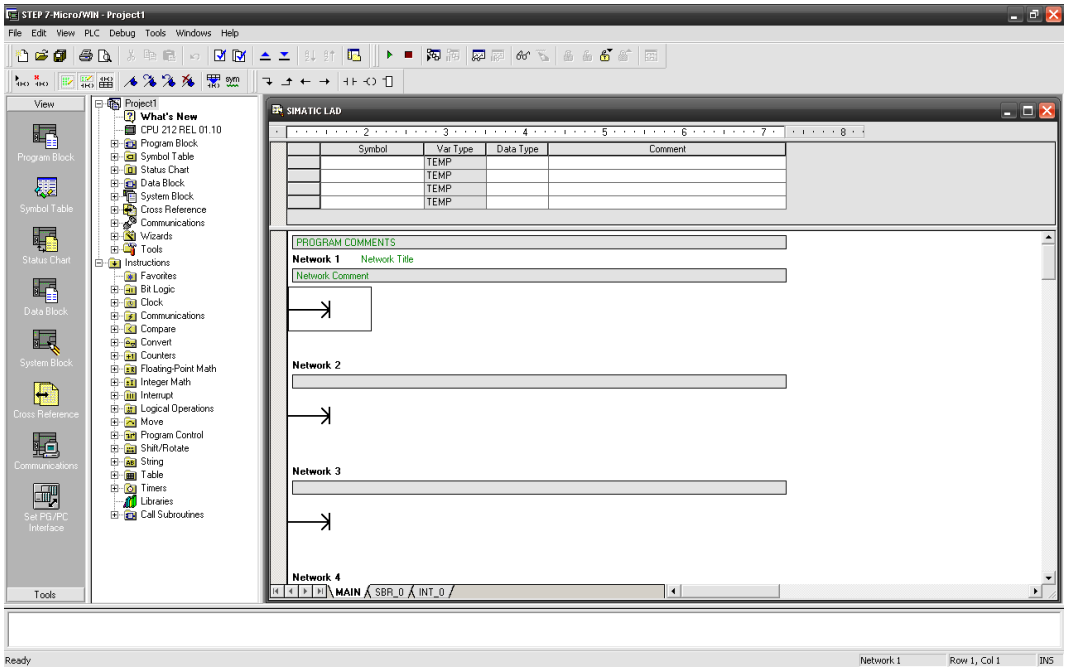
7.3.1 Tworzenie nowego projektu

Instalator domyślnie powinien utworzyć skróty do programów narzędziowych na pulpicie (patrz rysunek 7.13) oraz w menu start. Aby przystąpić do programowania uruchamiamy środowisko STEP 7-Micro/WIN.

Rysunek 7.13 Ikona programu STEP 7-Micro/WIN

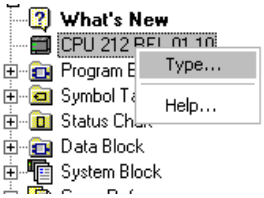
[1] ang. Networks

Po uruchomieniu programu, domyślnie powinien zostać otwarty pusty projekt. Domyślnym językiem jest LAD.



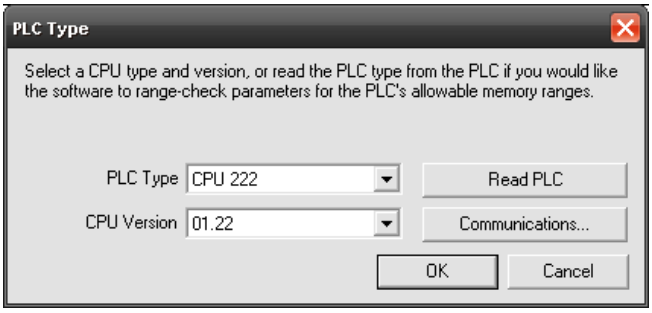
Rysunek 7.14 Pusty projekt

Pierwszą czynnością jaką należy zrobić jest konfiguracja typu sterownika, w tym celu klikamy prawym klawiszem myszy na jednostkę centralną widoczną w drzewie projektu i z menu kontekstowego wybieramy opcję ***Type...***.



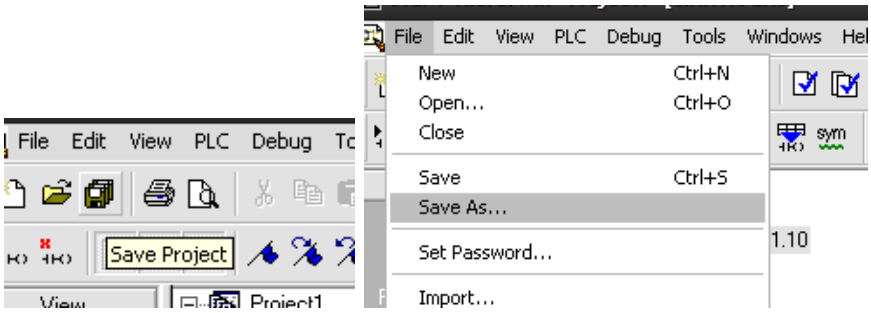
Rysunek 7.15 Wybór typu sterownika

W oknie konfiguracji wybieramy odpowiedni typ jednostki (patrz rysunek 7.16).



Rysunek 7.16 Typ i wersja sterownika

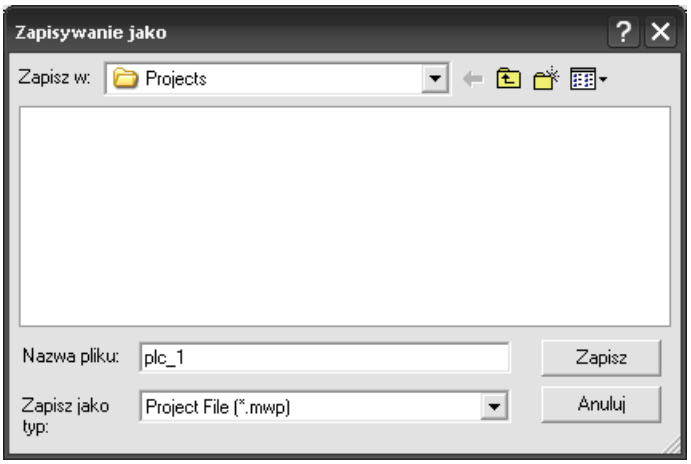
Po zakończeniu konfiguracji należy zapisać projekt.



(a) Przycisk paska narzędzi (b) Menu ***File***

Rysunek 7.17 Zapis projektu

Domyślnie projekty zapisywane są w katalogu *Projects* znajdującym się w folderze aplikacji (domyślnie: *C:\Program Files\Siemens\STEP 7-MicroWIN V4.0*).

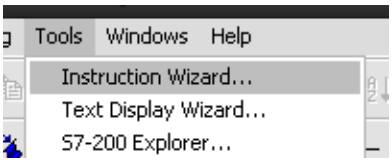


Rysunek 7.18 Wybór ścieżki zapisu

7.3.2 Konfiguracja szybkiego licznika

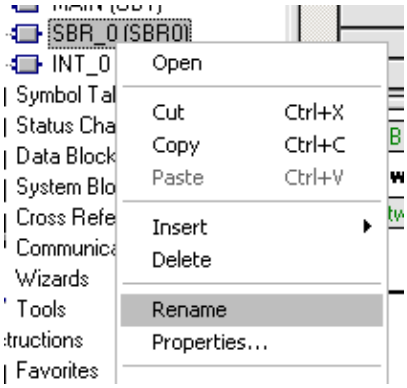
Gdy mamy już wstępnie skonfigurowany projekt możemy przystąpić do programowej konfiguracji urządzenia. Najpierw zajmiemy się konfiguracją szybkich liczników HSC[2]. Możemy do tego wykorzystać kreator instrukcji[3] dostępny z menu ***Tools*** (patrz rysunek 7.19).

Korzystając z tej opcji nie mamy możliwości skonfigurowania wszystkich parametrów. Dlatego w tym rozdziale opisana zostanie ręczna konfiguracja szybkich liczników.



Rysunek 7.19 Kreator konfiguracji instrukcji

Pierwszą czynnością jaką należy zrobić jest dodanie procedury (w domyślnie utworzonym projekcie powinna istnieć pusta procedura o nazwie **SBR_0 (SBR0)** widoczna w drzewie projektu w gałęzi ***Program Block***). Zmieniamy jej nazwę (patrz rysunek 7.20) na ***HSC_INIT*** - będzie to procedura inicjująca szybki licznik, wywoływana raz przy starcie sterownika[4].



Rysunek 7.20 Zmiana nazwy procedury

Po tej czynności klikamy nazwę procedury dwukrotnie, zostanie ona otworzona w edytorze programu.

[2]ang. High Speed Counter
[3]ang. Instruction Wizard...
[4]ang. First Scan

Pierwszą czynnością jaką musimy wykonać jest konfiguracja przerwania ***INT_ 0*** które będzie wyzwalane licznikiem ***T32***. W tym celu wstawiamy do programu styk otwarty i przypisujemy do niego bit systemowy ***SM0.1*** (symbolizuje on pierwszy przebieg programu, tzw. *First Scan* - dokładny wykaz bitów systemowych można znaleźć w dokumentacji sterownika, bądź w pomocy programu). Następnie z gałęzi ***Interrupt*** drzewa projektu przeciągamy blok **ATCH**, konfiguruje on przerwanie - pierwsza wartość wejściowa definiuje, która procedura zostanie wywołana w wyniku przerwania (w tym przypadku ***INT_ 0***). Druga wartość oznacza zdarzenie wyzwalające przerwanie (patrz tabela 7.1). Na koniec przeciągamy blok **ENI**, którego wywołanie spowoduje włączenie obsługi przerwań w mikroprocesorze sterownika.

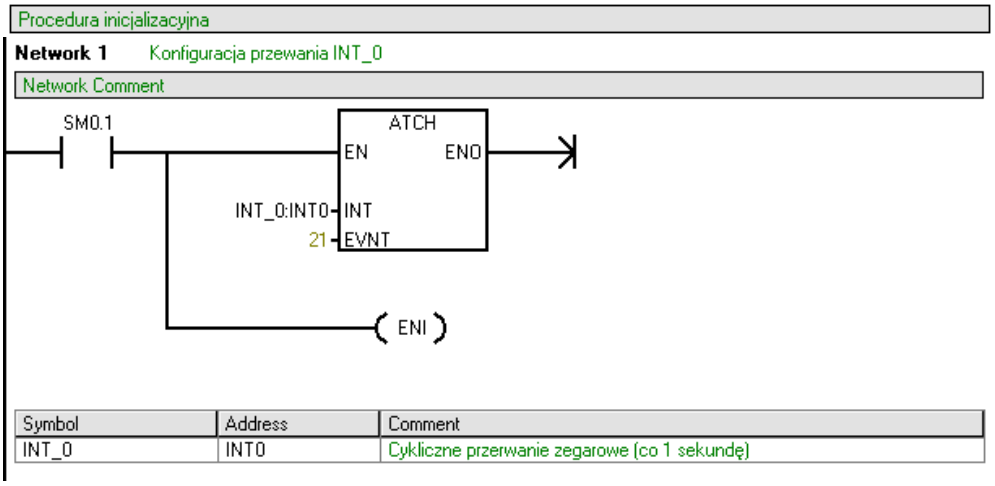


Tabela. 7.1 Lista przerwań sterownika S7-200 (CPU 222)

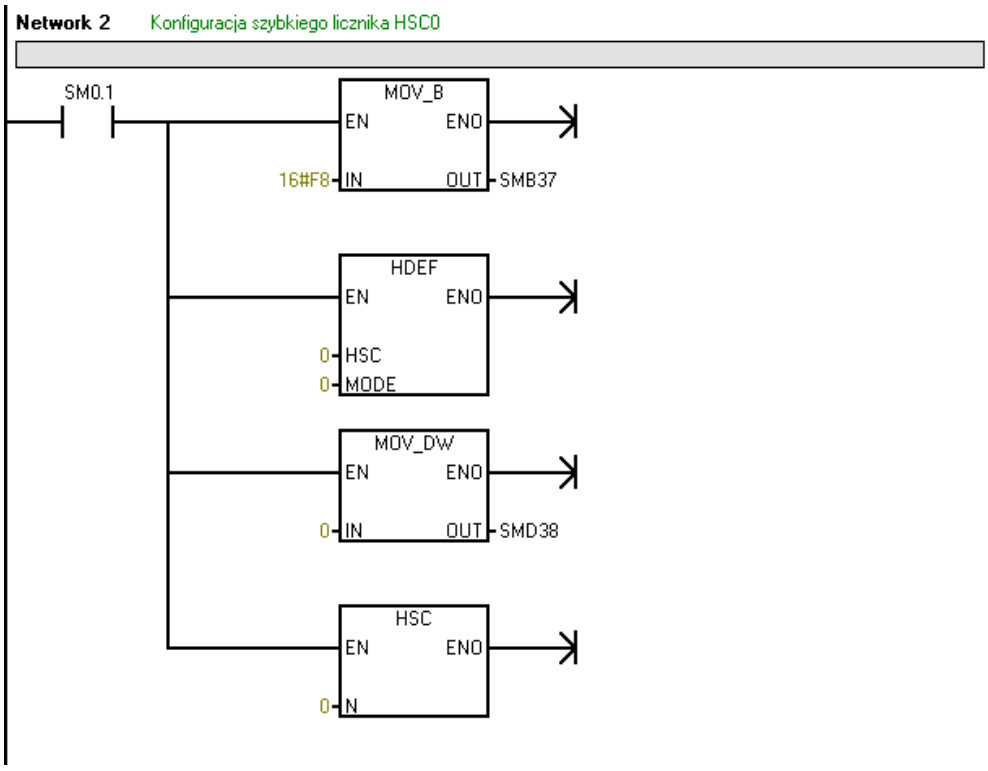
Przerwanie		Opis
0	I0.0	Zbocze narastające
1	I0.0	Zbocze opadające
2	I0.1	Zbocze narastające
3	I0.1	Zbocze opadające
...		
8	Port 0	Odbiór znaku
9	Port 0	Zakończenie transmisji
...		
12	HSC0	CV=PV (wartość bieżąca = wartość ustalona)
...		
19	PLS0	PTO przerwanie zakończenia zliczania impulsów
20	PLS1	PTO przerwanie zakończenia zliczania impulsów
...		
21	Timer T32	CT=CP wyzwala przerwanie
22	Timer T96	CT=CP wyzwala przerwanie
23	Port 0	Zakończenie odbioru wiadomości
...		
27	HSC0	Zmiana kierunku
28	HSC0	Zewnętrzny reset

W drugiej sieci (***Network 2***) tej samej procedury skonfigurujemy szczegółowe ustawienia licznika HSC0. Każdy szybki licznik posiada dwa rejestry:

- **Sterujący** - za jego pomocą ustawia się parametry licznika (***SM37*** dla licznika HSC0).
- **Statusowy** - tylko do odczytu, zawiera informacje o stanie licznika jak np. kierunek zliczania czy informację o stanie wartości bieżącej względem wartości zadanej(***SM36*** dla licznika HSC0).

Konfigurację zaczynamy od wstawienia styku otwartego przypisanego do bitu systemowego ***SM0.1***. Którym uaktywniamy następujące instrukcje (bloki):

1. **MOV_B** (instrukcja przesunięcia bajtu) za jej pomocą zaprogramujemy ustawienia licznika (wpiszemy odpowiednią wartość do rejestru ***SM37***). Aby odblokować licznik, włączyć zapisywanie nowej wartości bieżącej i zadanej, ustawić kierunek zliczania w górę oraz ustawić wejścia reset i start jako aktywne w stanie wysokim należy do rejestru sterującego wpisać wartość ***16#F8*** (patrz rysunek 7.21).
2. **HDEF** konfigurujemy tryb licznika - ustawiamy licznik w trybie 0 (szczegółowe informacje na temat trybów licznika znajdują się w pomocy programu oraz dokumentacji sterownika).
3. **MOV_DW** (instrukcja przesunięcia słowa) - ja jej pomocą wpisujemy do rejestru ***SM38*** (nowa wartość bieżąca) wartość 0 aby wyzerować licznik.
4. **HSC** programuje licznik zgodnie z ustawionymi wartościami.



Rysunek 7.21 Konfiguracja szybkiego licznika

Kolejną czynnością jest napisanie procedury przerwania. W tym celu otwieramy procedurę ***INT_0*** klikając dwukrotnie jej nazwę w drzewie projektu. Operacje jakie musimy wykonać w tym podprogramie to:

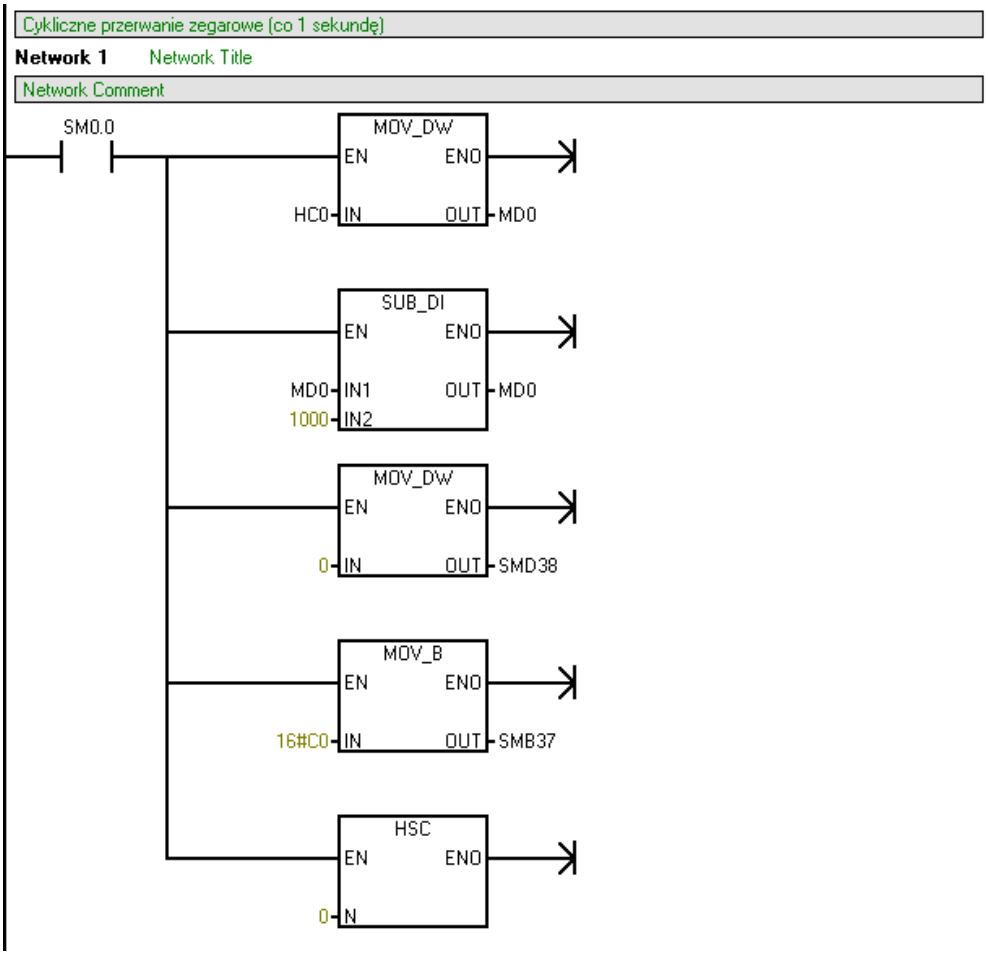
1. **MOV_DW** - skopiowanie bieżącej wartości licznika (z obszaru ***HC0***) do pamięci sterownika ***MD0***

2. **SUB_DI** - odejmujemy od skopiowanej wartości (pomiar częstotliwości sygnału w standardzie $1-5KHz$) 1000 aby przeskalować ją do zakresu $0 \div 4000$.

3. **MOV_DW** - wpisujemy do licznika (obszaru ***SM38***) nową wartość bieżącą (zerujemy go).

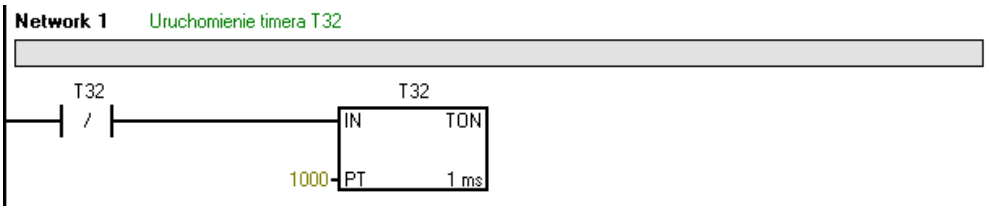
4. **MOV_B** - wpisanie $16\#C0$ do rejestru sterującego (***SM37***) wymusza odświeżenie wartości bieżącej oraz odblokowanie licznika.

5. **HSC** - przeprogramujemy licznik w celu ustawienia nowej wartości bieżącej.

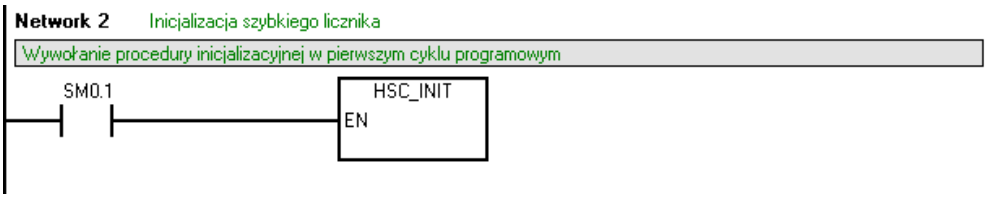


Po skonfigurowaniu procedury przerwania możemy przystąpić do programowania głównej pętli programowej.

W pierwszej sieci (***Network 1***) ustawiamy licznik **T32** tak, aby wywoływał przerwanie co 1 sekundę. Ponieważ licznik ten ma rozdzielczość $1ms$ jego nastawa powinna wynosić 1000 ($1ms \cdot 1000 = 1s$). Szczegóły konfiguracji przedstawia rysunek 7.3.2. Więcej informacji o trybach pracy i ustawieniach liczników można znaleźć w dokumentacji sterownika i pomocy programu.



W drugiej sieci (***Network 2***) wstawiamy procedurę inicjującą tak, aby wykonywana była tylko w pierwszym przebiegu programu - wyzwalamy ją stykiem otwartym bitu systemowego ***SM0.1***.



Utworzony program zapisujemy - posłuży on nam jako projekt wyjściowy do programowania sterowania dwu- i trójstanowego.

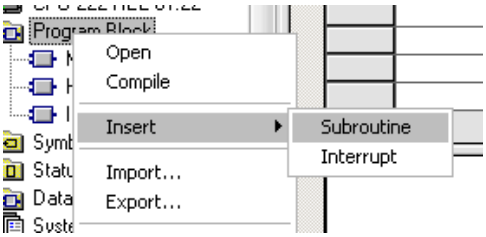
7.3.3 Implementacja algorytmu sterowania dwustawnego

Dla ułatwienia zrozumienia programu w poniższej tabeli podano listę obszarów pamięci procesora wraz z przypisanymi do nich wartościami procesowymi i zmiennymi pośrednimi. Sterowanie następuje poprzez załączenie wyjścia binarnego $Q0.0$.

Tabela. 7.2 Wykaz zmiennych przechowywanych w pamięci procesora

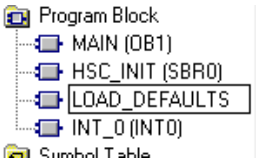
Obszar pamięci	Opis zmiennej
MD0	Zmienna procesowa (PV)
MD1	Wartość zadana (SP)
MD2	Histereza (H)
MD3	$\frac{H}{2}$ *(zmienna pośrednia)*
MD4	$-\frac{H}{2}$ *(zmienna pośrednia)*
MD5	Uchyb regulacji $(E = SP - PV)$ *(zmienna pośrednia)*

Do stworzonego uprzednio projektu dodajemy procedurę, która będzie zawierać domyślne nastawy regulatora wczytywane przy uruchomieniu sterownika.



Rysunek 7.22 Wstawienie podprogramu

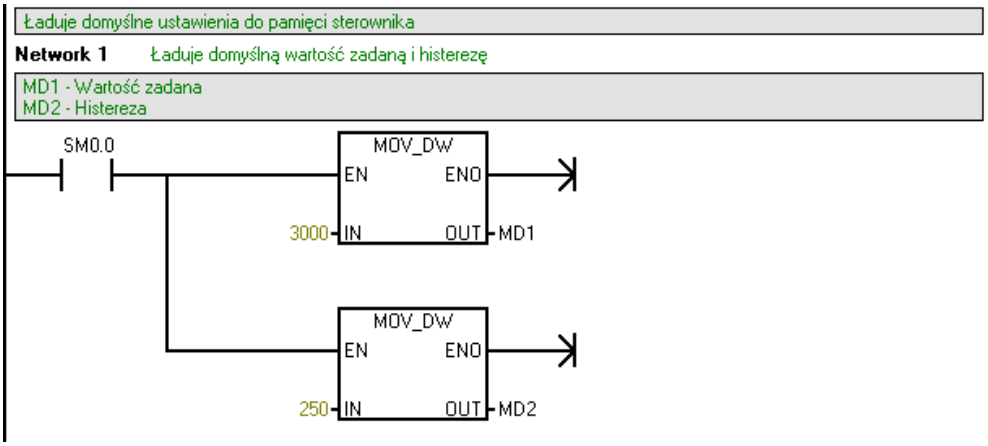
Zmieniamy jej nazwę na **LOAD_DEFAULTS**.



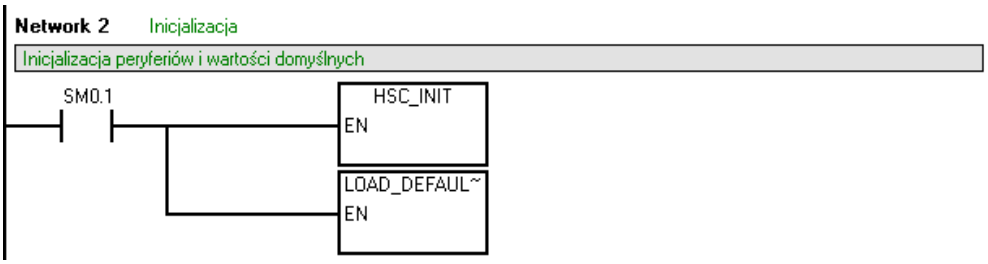
Rysunek 7.23 Zmiana nazwy procedury

Wartości domyślne ustawiamy za pomocą bloków **MOV_DW**.

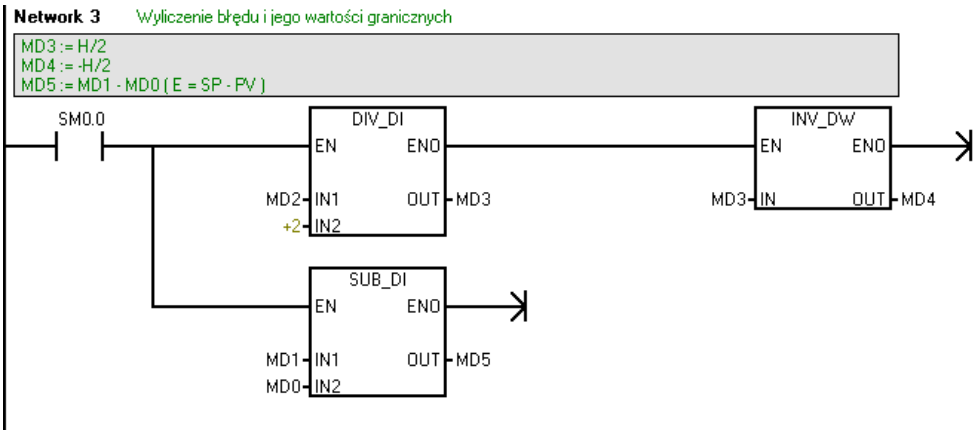
1. do ***MD1*** wpisujemy 2000 (domyślna wartość zadana - 50% zakresu)
2. do ***MD2*** wpisujemy 250 (domyślna histereza - 6.25% zakresu)



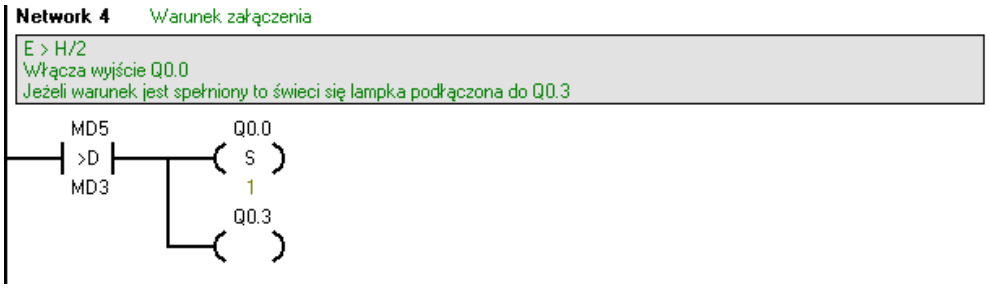
Gotową procedurę umieszczamy ją w drugiej sieci (***Network 2***) programu głównego (***MAIN***) równolegle do procedury **HSC_INIT**



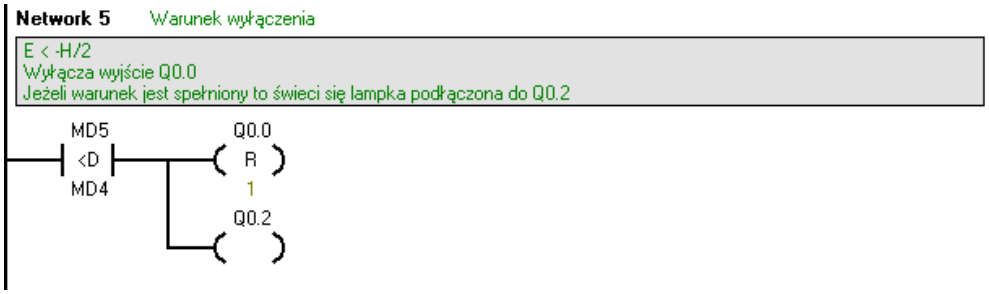
Korzystając z instrukcji **DIV_DI** (operacja dzielenia), **INV_DV** (inwersja bitów - w kodowaniu U2 odpowiada zmianie znaku) oraz **SUB_DI** (operacja odejmowania) obliczamy zmienne pośrednie. (patrz tabela 7.2)



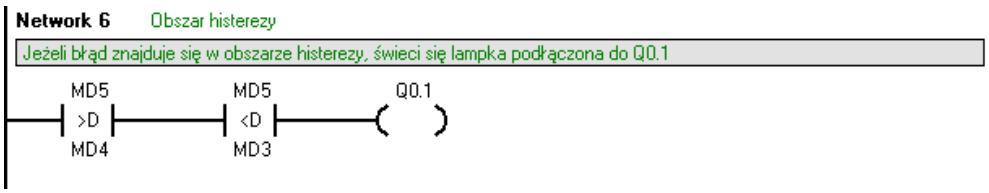
Programujemy warunek załączenia: jeżeli $E > \frac{H}{2}$ to włącz wyjście $Q0.0$. Dodatkowo, gdy warunek jest spełniony świeci się lampka podłączona do wyjścia $Q0.3$



Programujemy warunek załączenia: jeżeli $E < -\frac{H}{2}$ to wyłącz wyjście $Q0.0$. Dodatkowo, gdy warunek jest spełniony świeci się lampka podłączona do wyjścia $Q0.2$



Jeżeli uchyb znajduje się w obszarze histerezy ($-\frac{H}{2} < E < \frac{H}{2}$) świeci się lampka podłączona do wyjścia $Q0.1$



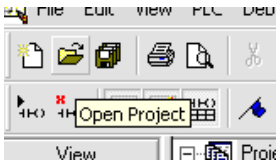
7.3.4 Implementacja algorytmu sterowania trójstawnego

Dla ułatwienia zrozumienia programu w poniższej tabeli podano listę obszarów pamięci procesora wraz z przypisanymi do nich wartościami procesowymi i zmiennymi pośrednimi. Sterowanie następuje poprzez załączenie wyjść $Q0.0$ (stan $+1$) oraz $Q0.0$ (stan -1)

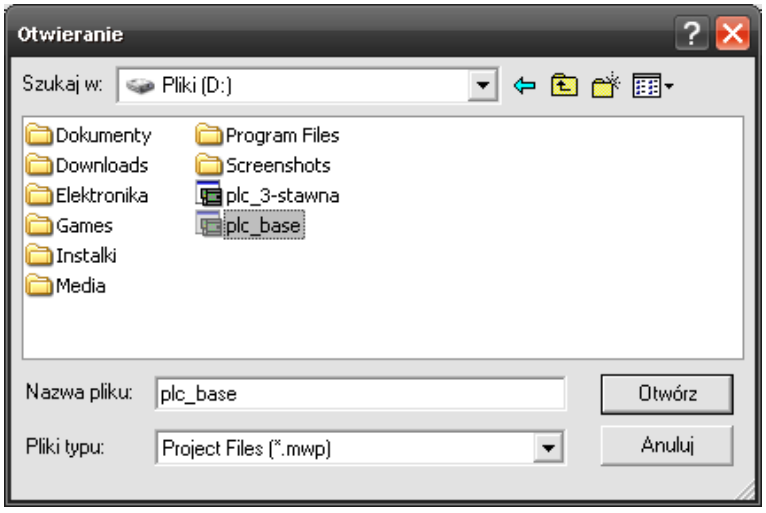
Tabela. 7.3 Wykaz zmiennych przechowywanych w pamięci procesora

Obszar pamięci	Opis zmiennej
MD0	Zmienna procesowa (PV)
MD1	Wartość zadana (SP)
MD2	Obszar nieczułości (N)
MD3	Histereza (H)
MD4	Uchyb regulacji ($E = SP - PV$)
MD10	$\frac{N}{2}$
MD11	$-\frac{N}{2}$
MD12	$\frac{H}{2}$
MD16	$\frac{N}{2} + \frac{H}{2}$
MD17	$\frac{N}{2} - \frac{H}{2}$
MD18	$-\frac{N}{2} + \frac{H}{2}$
MD19	$-\frac{N}{2} - \frac{H}{2}$

Zanim przystąpimy do zaprogramowania sterowania trójstawnego musimy otworzyć projekt ze wstępnie skonfigurowanym licznikiem ***HSC0***

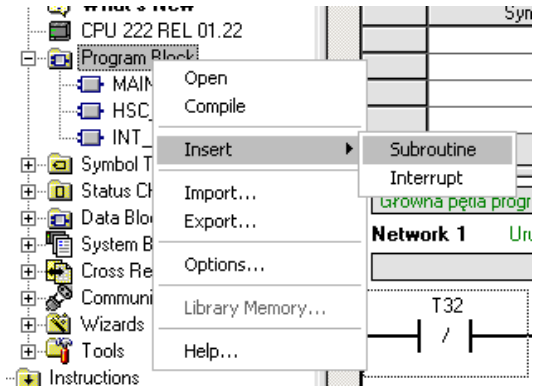


Rysunek 7.24 Otwarcie projektu



Rysunek 7.25 wybór projektu

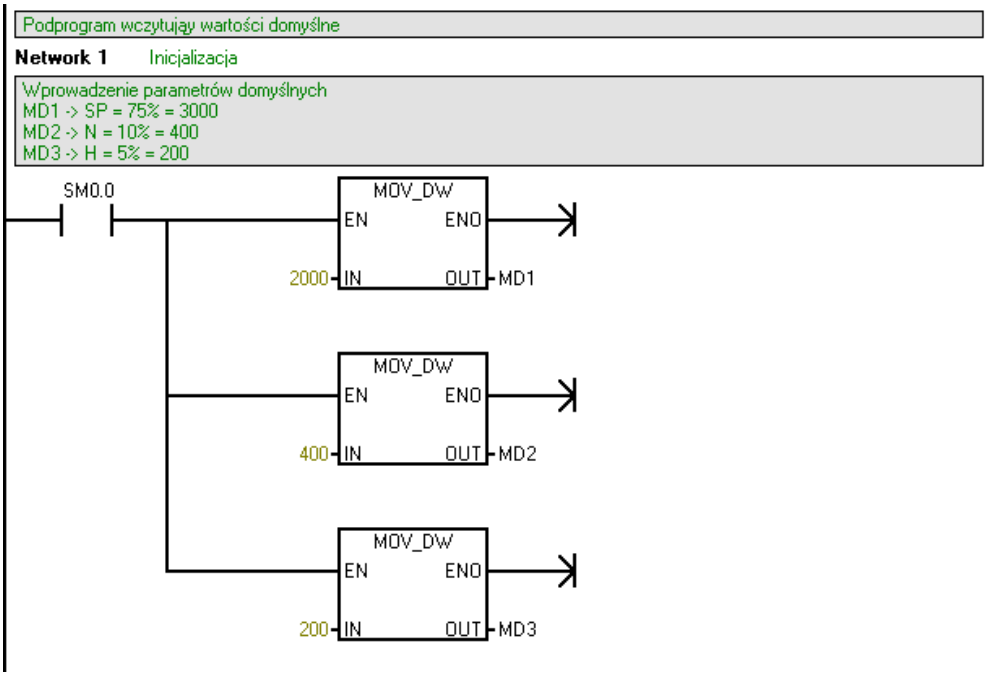
Podobnie jak w poprzednim przypadku wstawiamy procedurę z wartościami domyślnymi. Nazywamy ją ***LOAD_DEFAULTS***.



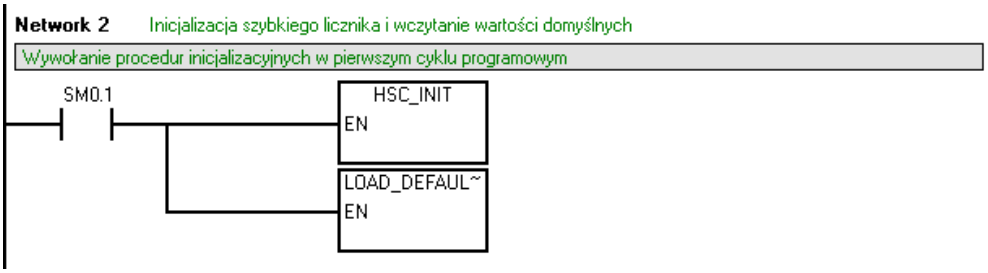
Rysunek 7.26 Wstawienie procedury

Wartości domyślne ustawiamy za pomocą bloków **MOV_DW**.

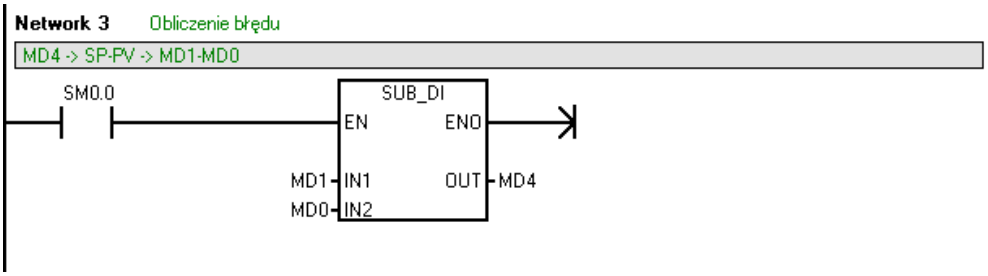
1. do ***MD1*** wpisujemy 2000 (domyślna nastawa - 50% zakresu)
2. do ***MD2*** wpisujemy 400 (domyślna strefa nieczułości - 10% zakresu)
3. do ***MD3*** wpisujemy 200 (domyślna histereza - 5% zakresu)



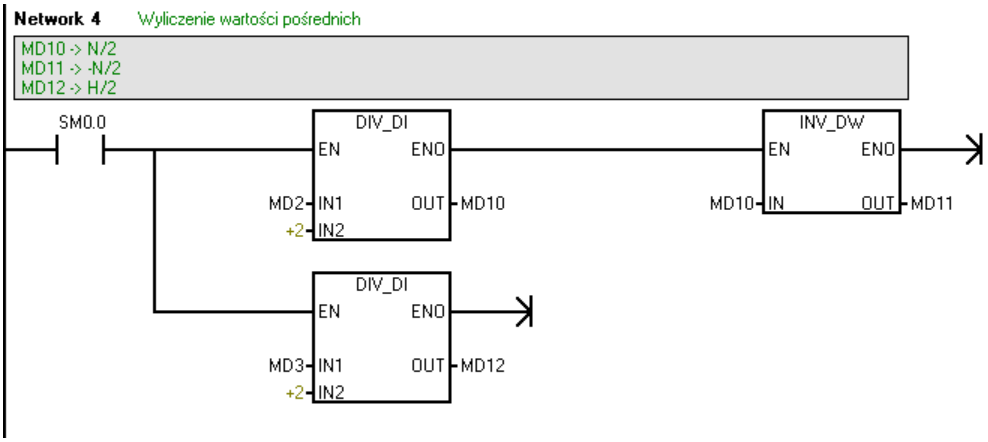
Gotową procedurę umieszczamy w drugiej sieci (***Network 2***) programu głównego (***MAIN***) równolegle do procedury **HSC_INIT**



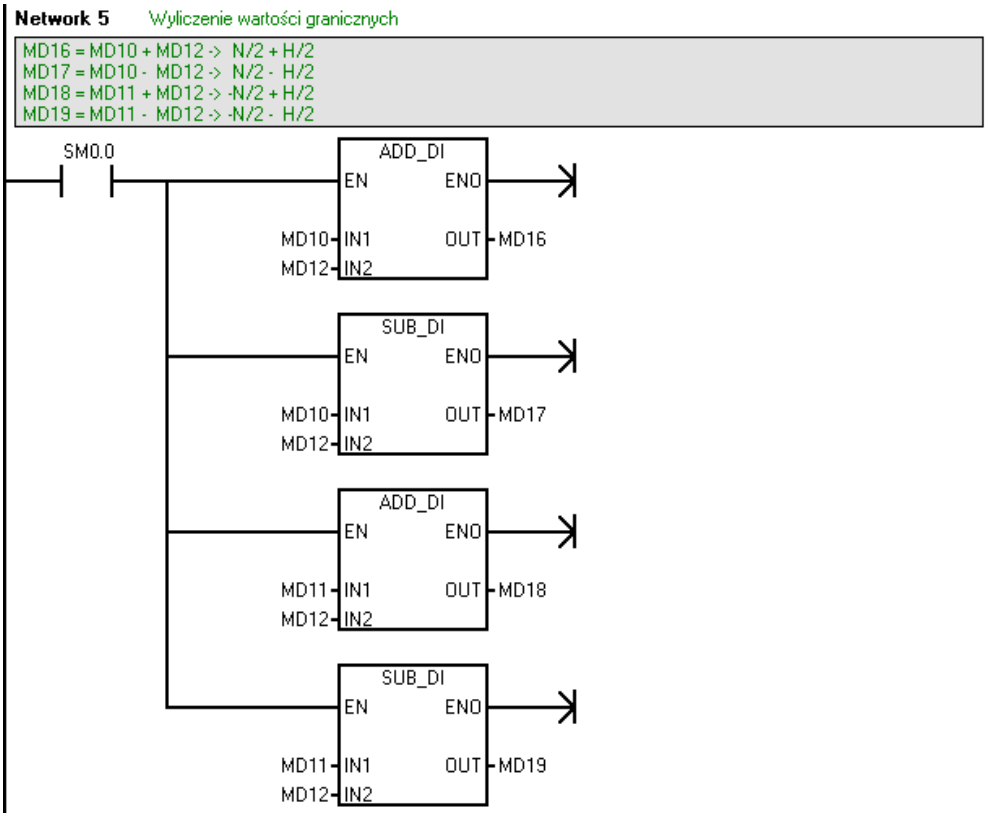
Dalej program wygląda analogicznie jak w przypadku regulacji dwustawnej. Najpierw wyznaczamy uchyb wartości regulowanej.



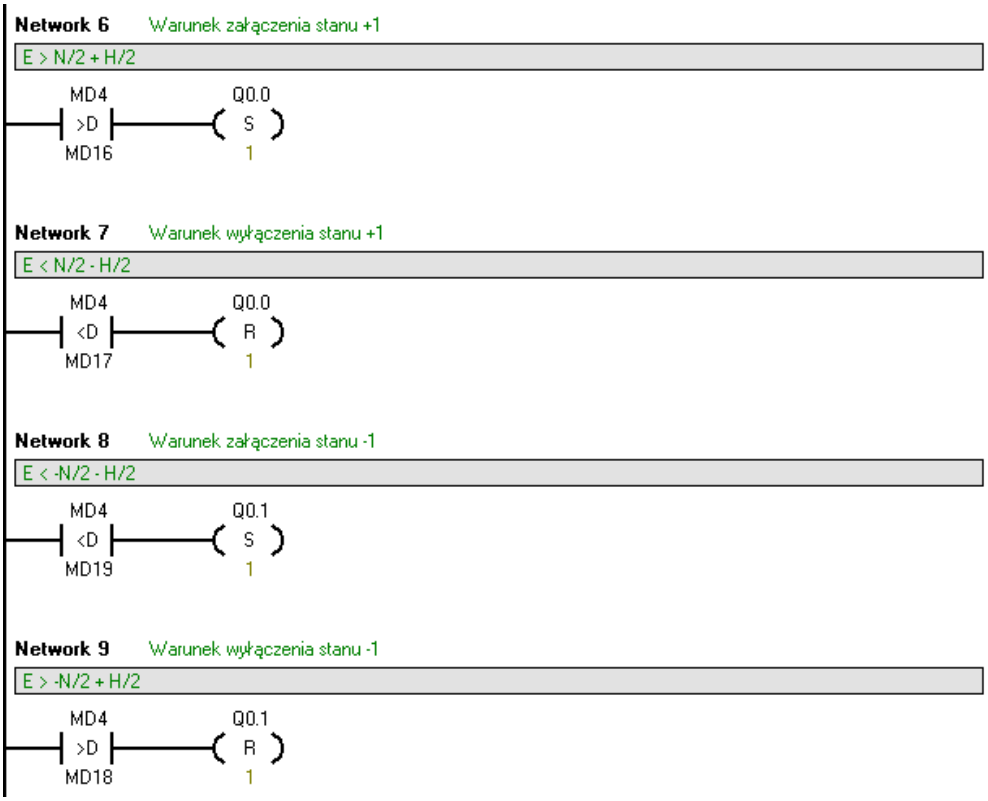
Wyznaczamy wartości pośrednie.



Wyznaczamy wartości graniczne.



Programujemy warunki załączenia/wyłączenia.



Rozdział 8

Wizualizacja procesu sterowania

Rysunek 8.1 Wonderware InTouch 9.5

Wizualizacja sterowania poziomem cieczy wykonana zostanie za pomocą oprogramowania InTouch firmy Wonderware.

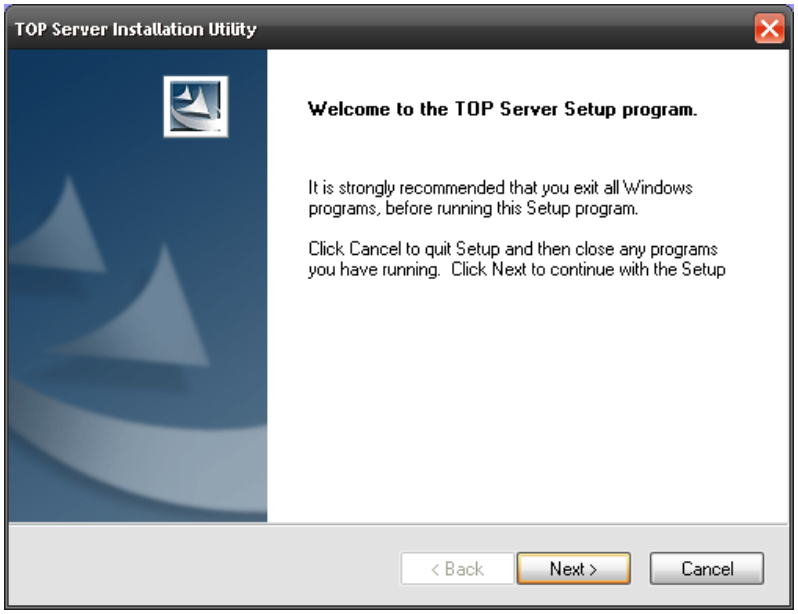
InTouch to jeden z najpopularniejszych programów typu HMI[1]/SCADA[2], służący do wizualizacji, kontroli oraz sterowania procesami technologicznymi lub produkcyjnymi. Charakteryzuje się ono łatwym w użyciu środowiskiem tworzenia aplikacji oraz rozległą funkcjonalnością, umożliwiającą szybkie budowanie, testowanie i wdrażanie wartościowych systemów udostępniających operatorom dane wprost z produkcji. Aplikacje napisane w środowisku InTouch są elastyczne, łatwo można je dostosować do aktualnych potrzeb użytkownika. Dzięki zastosowaniu najpopularniejszych protokołów komunikacyjnych system firmy Wonderware jest się wstanie skomunikować z większością urządzeń stosowanych w przemyśle (dostępnych jest ponad 1000 gotowych programów komunikacyjnych).

Aby aplikacja wizualizacji miała dostęp do zmiennych wewnętrznych sterownika PLC konieczne jest zewnętrzne źródło danych. W tym celu zostanie zainstalowany serwer danych OPC, który stanowi interfejs pomiędzy wizualizacją a sterownikiem.

[1]Human-Machine Interface (Interfejs Człowiek-Maszyna)
[2]Supervisory Control And Data Acquisition (system nadzorujący przebieg procesu przemysłowego)

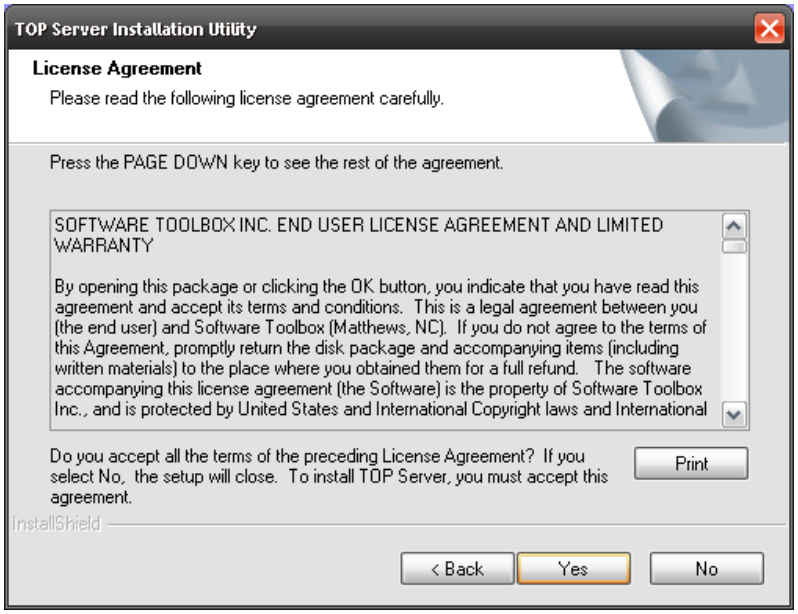
8.1 Instalacja serwera wymiany danych OPC

Przed rozpoczęciem, instalator poprosi nas o zamknięcie wszystkich programów działających w "tle" oraz wyłączenie programów antywirusowych i firewalli, które mogły by przeszkodzić w procesie instalacji.



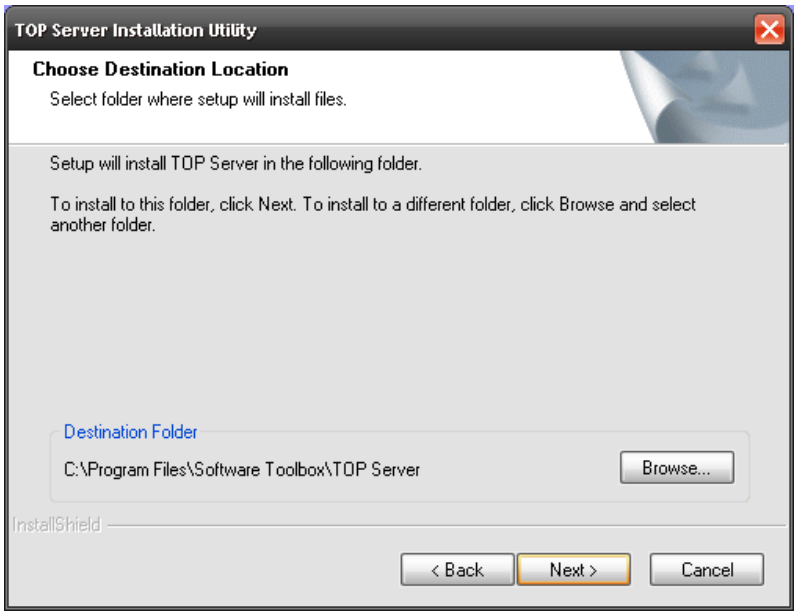
Rysunek 8.2 Okno instalatora

Aby kontynuować należy przeczytać i zaakceptować licencję oprogramowania.



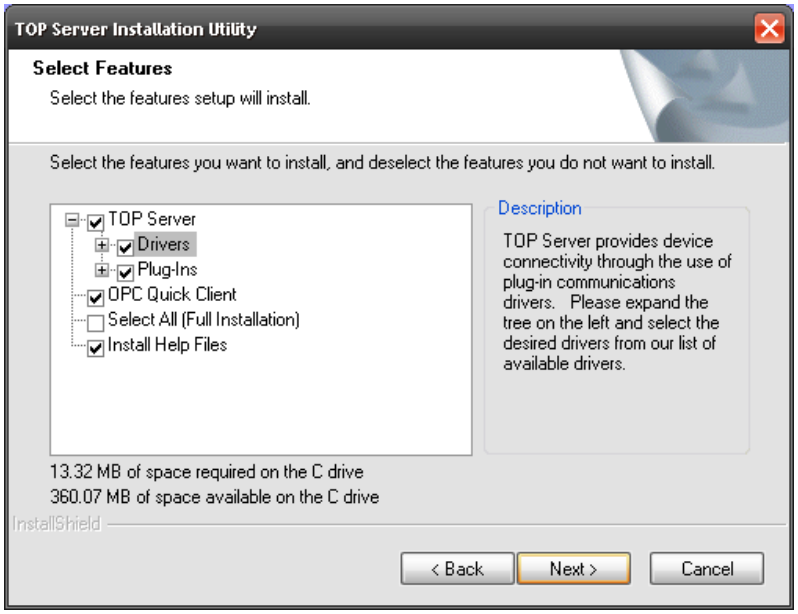
Rysunek 8.3 Akceptacja licencji

Następnie zostaniemy poproszeni o wybór folderu docelowego.



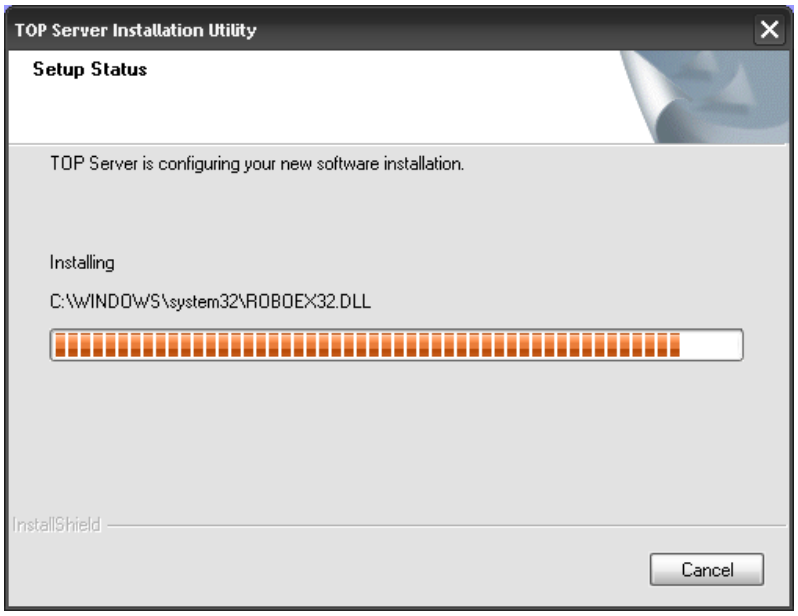
Rysunek 8.4 Wybór katalogu instalacyjnego

Spośród listy komponentów należy wybrać odpowiednie sterowniki (S7-200 i Simulator).



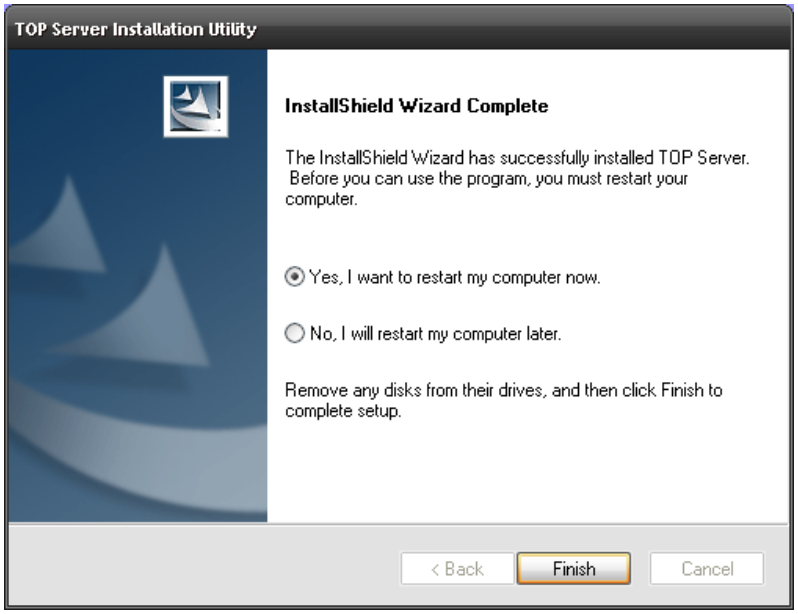
Rysunek 8.5 Wybór komponentów

Rozpocznie się instalacja oprogramowania.



Rysunek 8.6 Proces instalacji

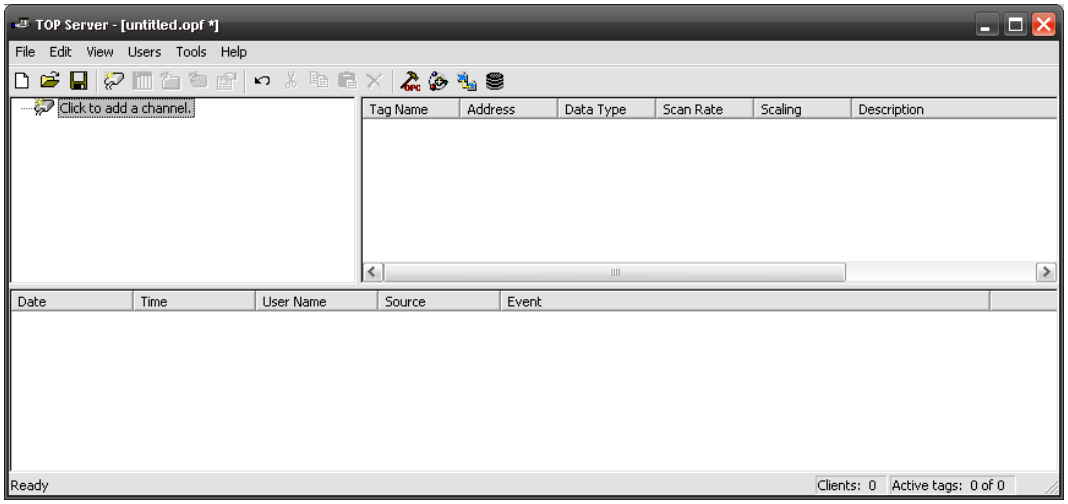
Po zakończeniu instalator zasugeruje ponowne uruchomienie komputera (zalecane).



Rysunek 8.7 Ponowne uruchomienie

8.2 Konfiguracja serwera wymiany danych OPC

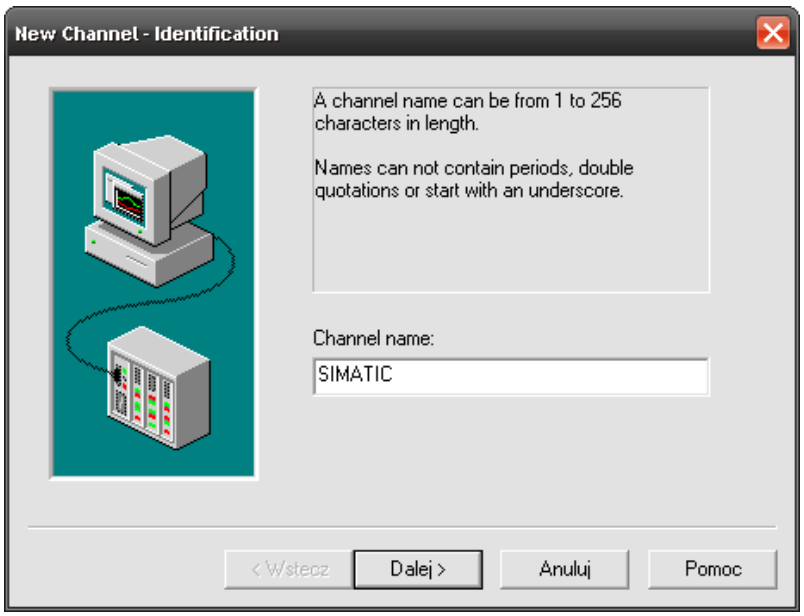
Po uruchomieniu programu TOP Server należy przystąpić do jego konfiguracji.



Rysunek 8.8 Główne okno programu

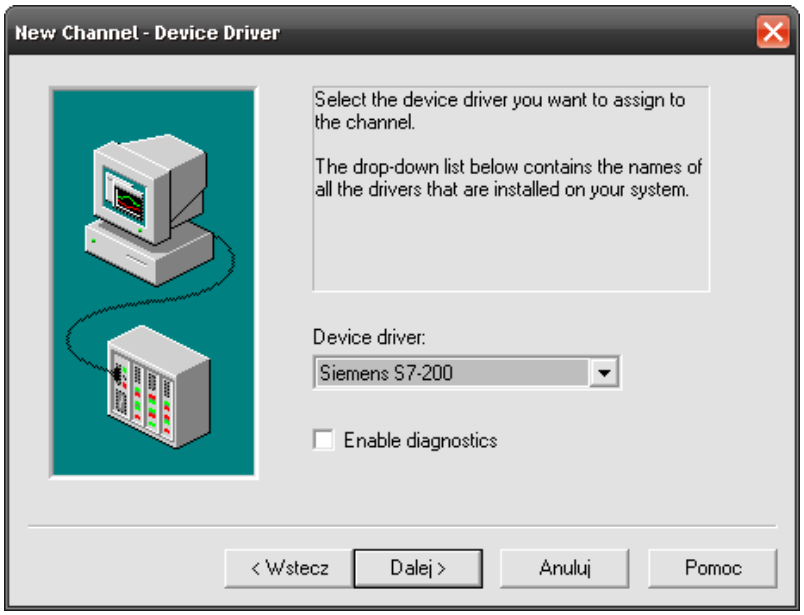
8.2.1 Tworzenie nowego kanału

Aby udostępnić zmienne sterownika należy najpierw stworzyć kanał komunikacyjny. W tym celu klikamy pole *"Click to add channel"* widoczne w głównym oknie programu (patrz rysunek 8.8). Zostanie otworzony kreator, który przeprowadzi nas przez proces jego tworzenia. W pierwszym kroku należy wybrać jego nazwę np.: *SIMATIC*.



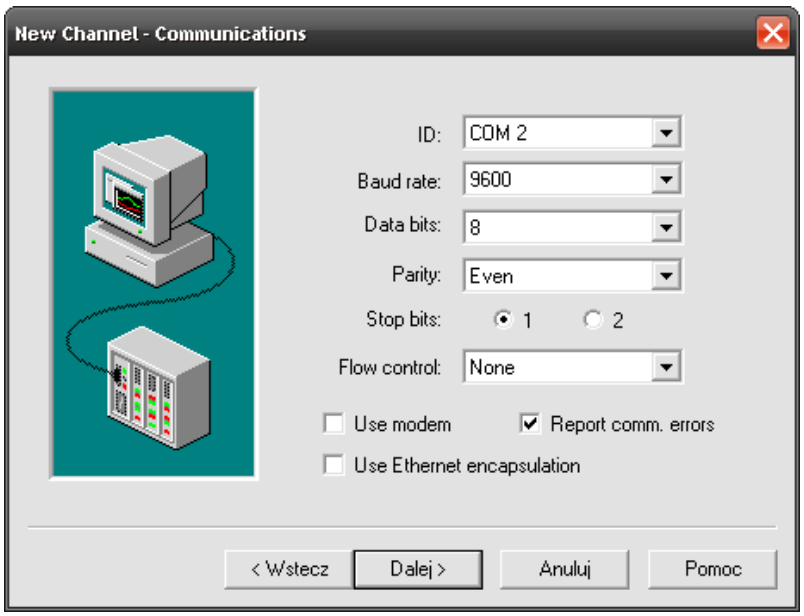
Rysunek 8.9 Tworzenie nowego kanału

W kolejnym kroku zostaniemy poproszeni o wybór sterownika urządzenia. Należy wówczas wybrać *"Siemens S7-200"*



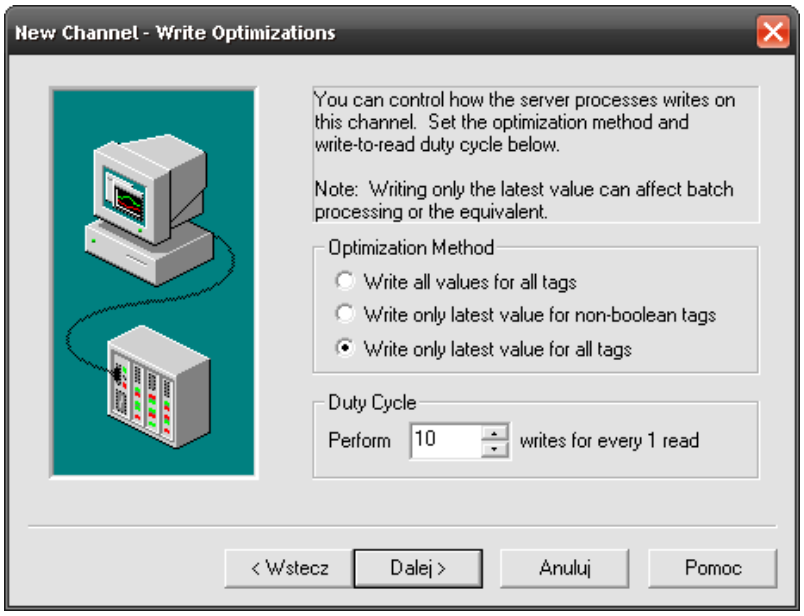
Rysunek 8.10 Wybór sterownika

Następnie należy skonfigurować ustawienia połączenia. W większości przypadków należy zostawić domyślne, zmieniając jedynie port komunikacyjny komputera do którego podłączony jest sterownik.



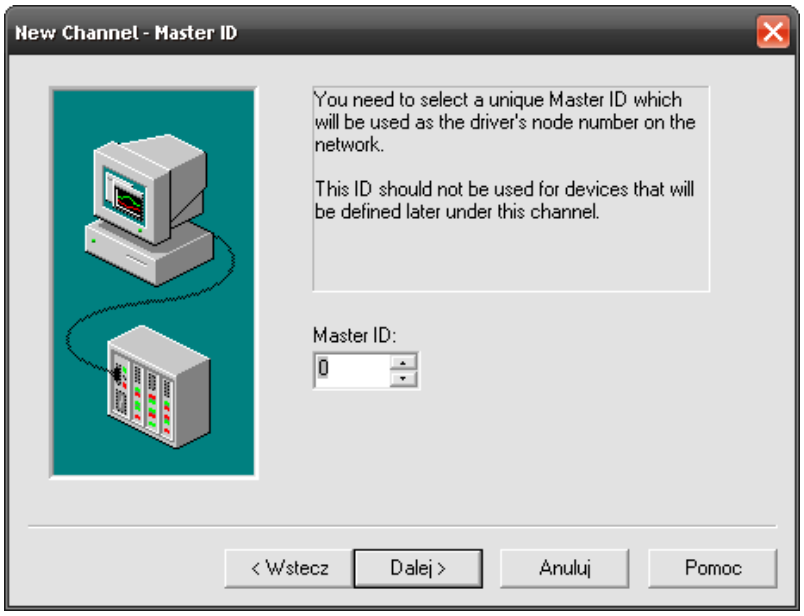
Rysunek 8.11 Konfiguracja połączenia

Zostaniemy poproszeni o wybór optymalizacji zapisu (dla bezpieczeństwa należy zostawić wartości domyślne).



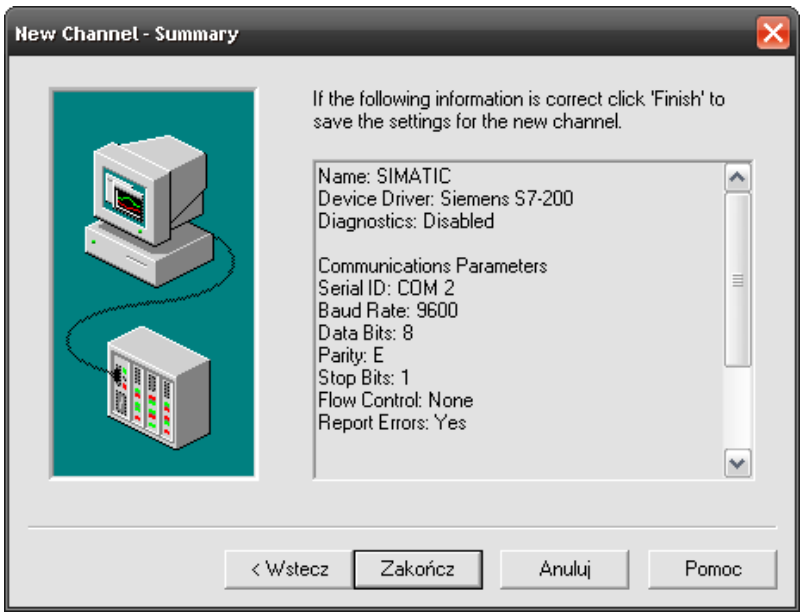
Rysunek 8.12 Optymalizacje zapisu

Ostatnim krokiem jest wybór tzw. *"Master ID"*, które w większości przypadków jest równe 0.



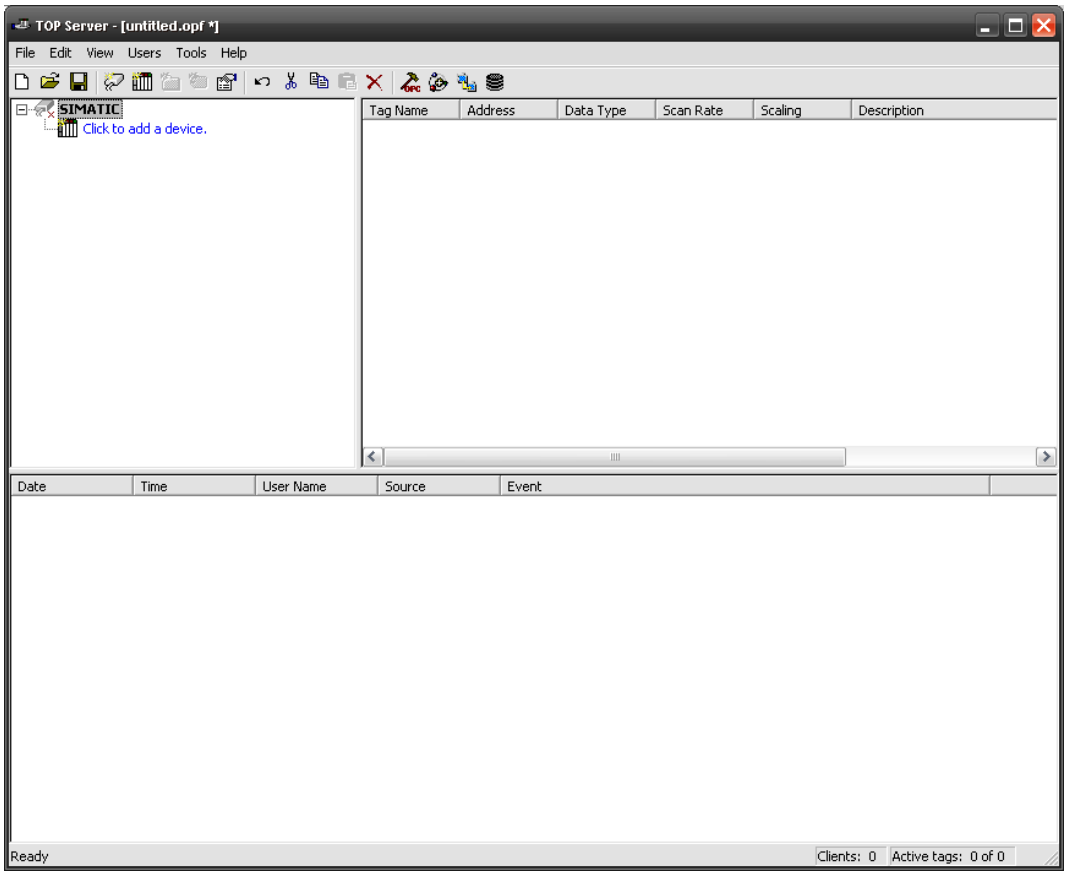
Rysunek 8.13 Wybór identyfikatora

Przed zakończeniem kreator pokaże nam wybrane opcje.



Rysunek 8.14 Podsumowanie

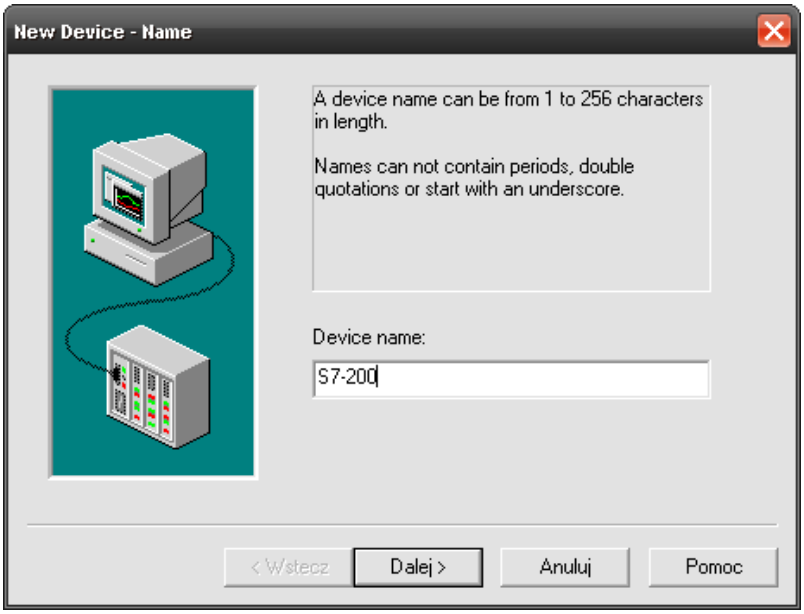
W głównym oknie powinien pojawić się nowy kanał komunikacyjny - *SIMATIC*.



Rysunek 8.15 Główne okno programu

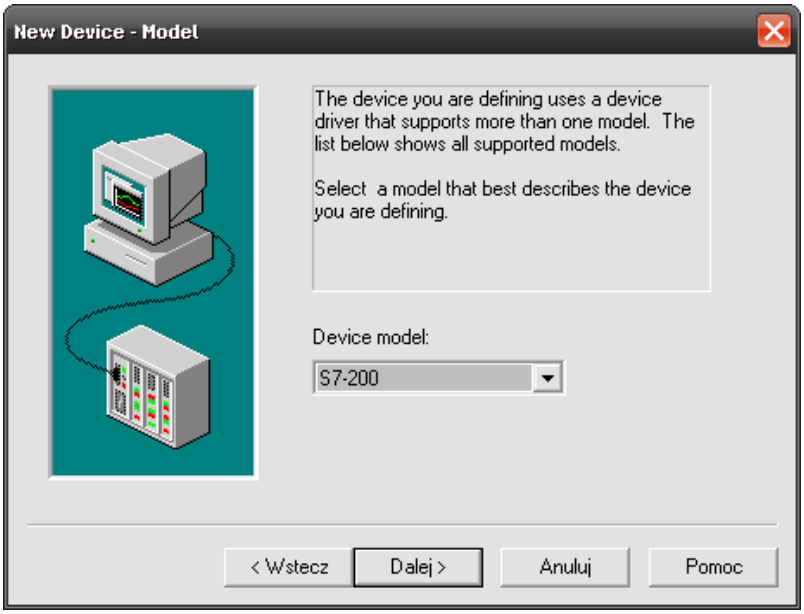
8.2.2 Dodawanie urządzenia

Do stworzonego kanału komunikacyjnego należy jeszcze dodać urządzenie. W tym celu klikamy pole *"Click to add device"* poniżej stworzonego kanału. Otworzony w ten sposób kreator poprosi o podanie nazwy urządzenia.



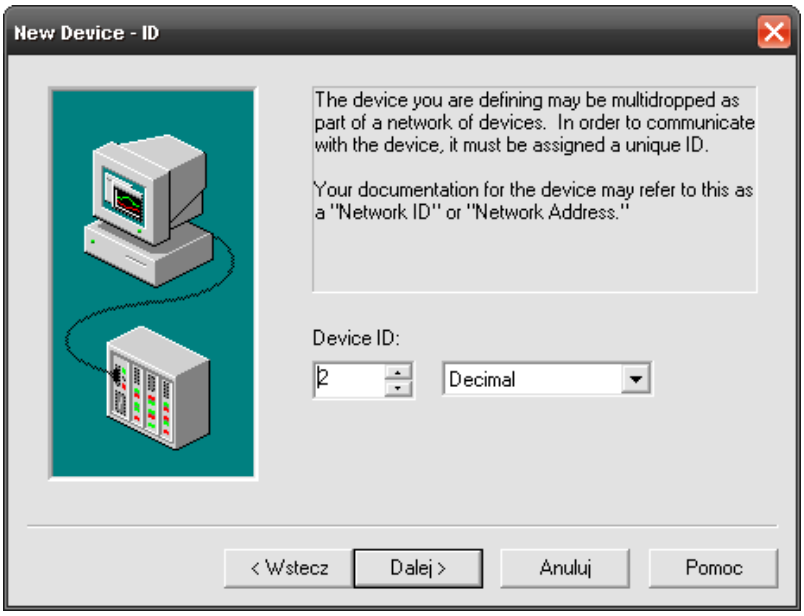
Rysunek 8.16 Tworzenie nowego urządzenia

W kolejnym kroku należy podać model urządzenia.



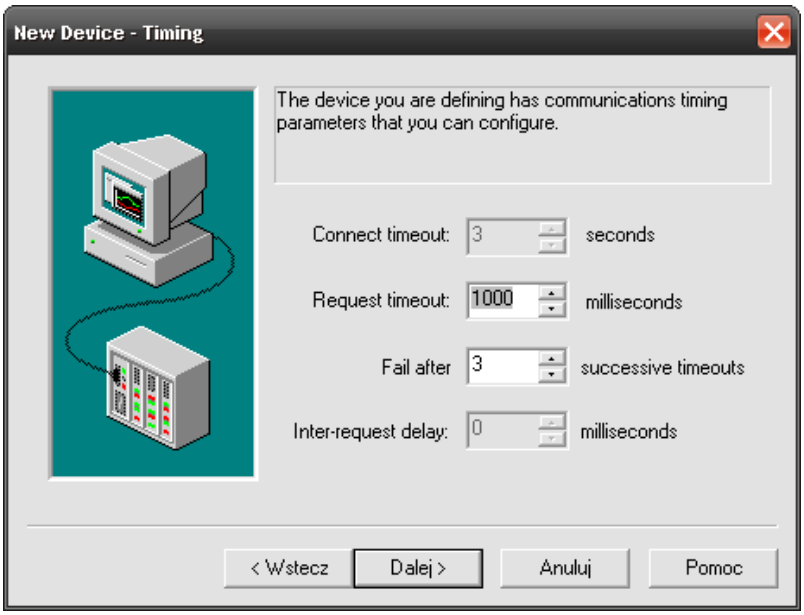
Rysunek 8.17 Wybór modelu urządzenia

Następnie zostaniemy poproszeni o wybór identyfikatora urządzenia. Domyślne wartości pokazane są na rysunku 8.18.



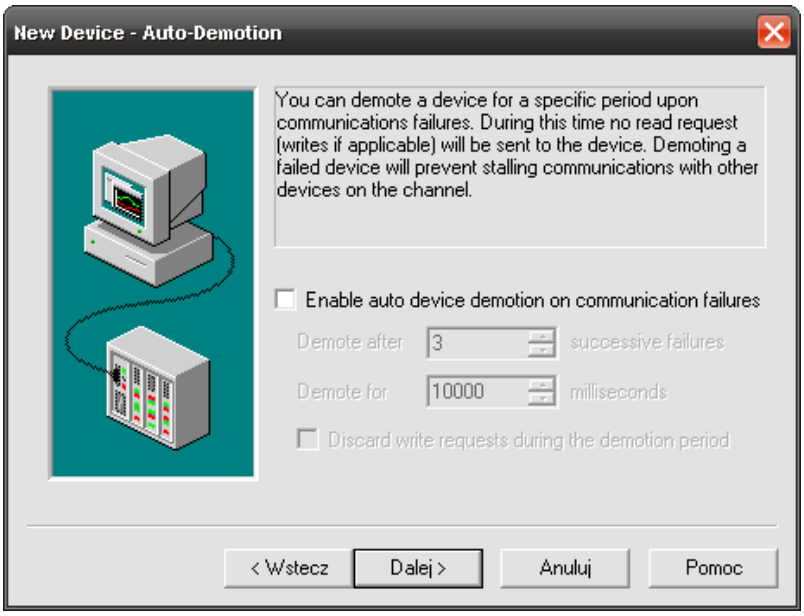
Rysunek 8.18 Wybór identyfikatora urządzenia

Ustawienia połączenia w większości przypadków nie muszą być zmieniane.



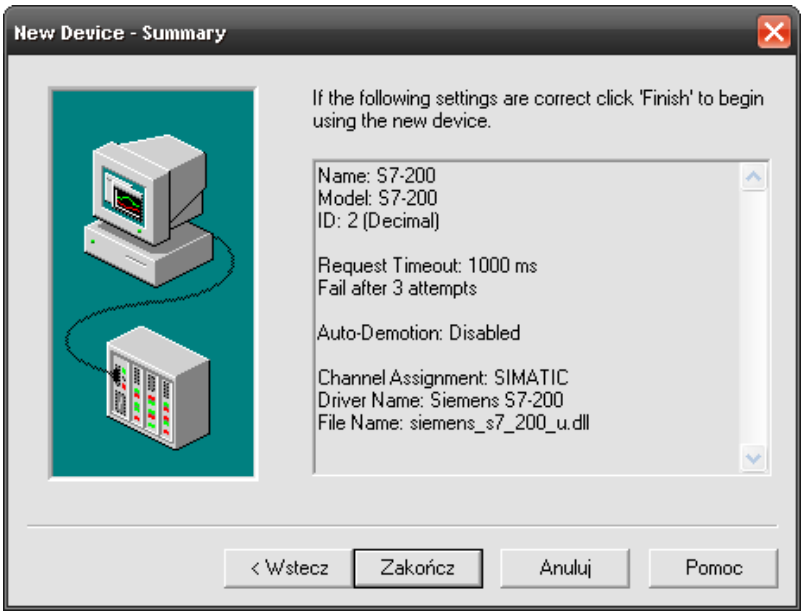
Rysunek 8.19 Ustawienia połączenia

Możemy ustawić specjalne akcje na wypadek awarii urządzenia.



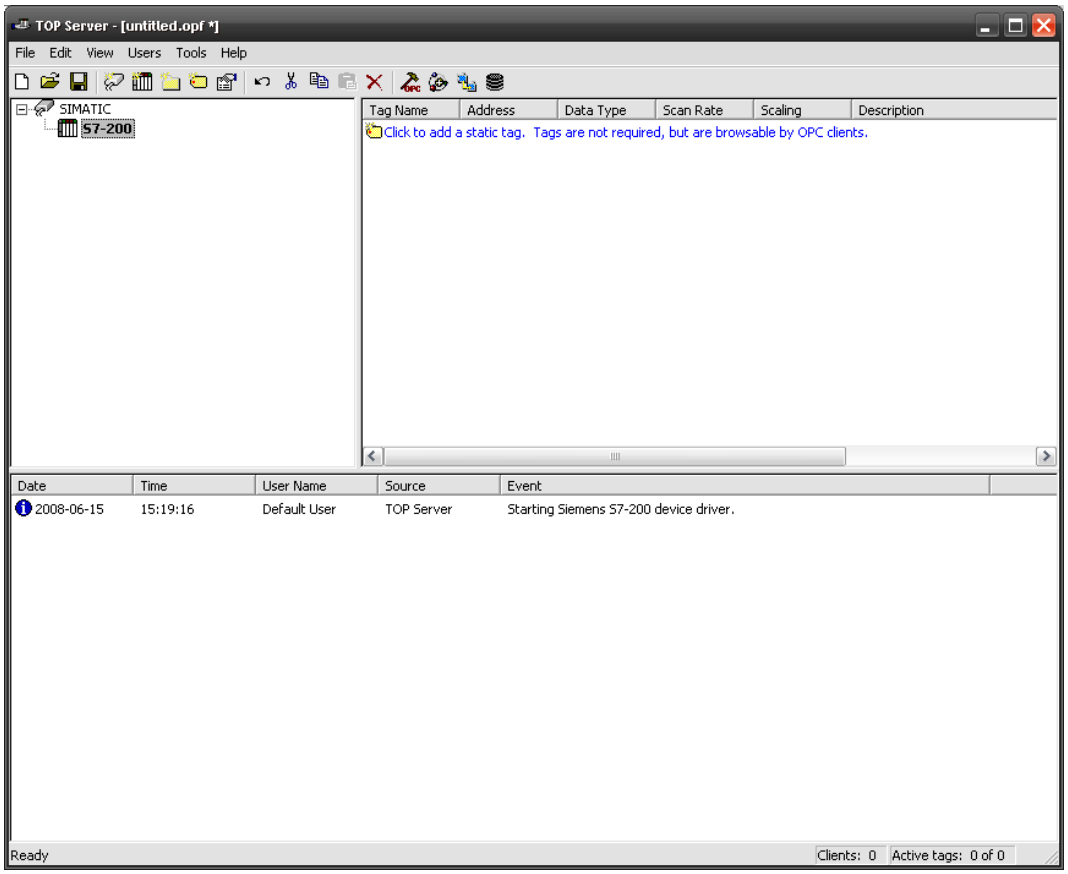
Rysunek 8.20 Ustawienia połączenia c.d.

Podobnie jak w poprzednim przypadku, przed zakończeniem zostaniemy poproszeni o akceptację wybranych ustawień.



Rysunek 8.21 Podsumowanie konfiguracji

W głównym oknie programu powinno pojawić się dodane urządzenie.

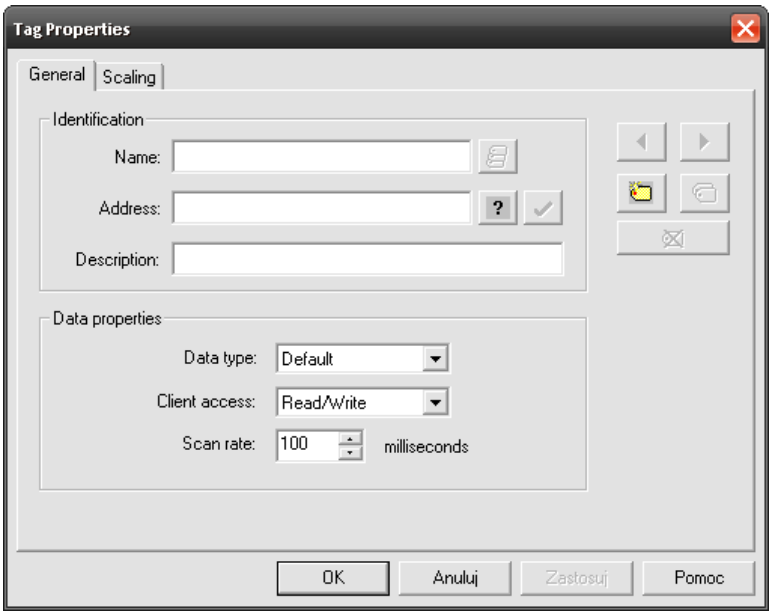


Rysunek 8.22 Główne okno programu

8.2.3 Dodawanie tagów

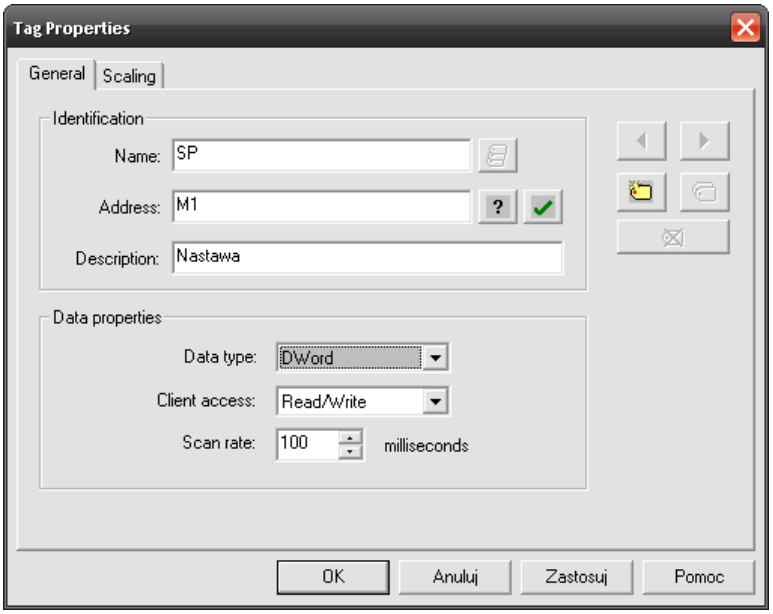
Po skonfigurowaniu połączenia ze sterownikiem, należy zdefiniować tzw. tagi, które reprezentować będą wewnętrzne zmienne sterownika. Aby dodać nowy tag klikamy *"Click to add a static tag"* w głównym oknie programu.

W oknie ustawień należy podać nazwę tagu oraz adres przypisanej do niego zmiennej sterownika. Opcjonalnie możemy podać opis zmiennej oraz ustawić skalowanie wartości.



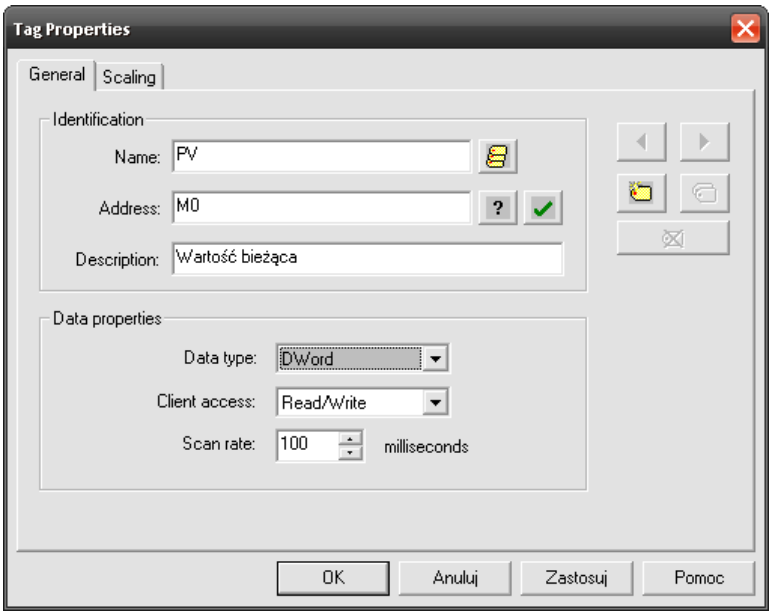
Rysunek 8.23 Okno dodawania tagów

Na rysunku 8.24 przedstawiona jest konfiguracja tagu ***SP*** dla zmiennej ***M1***.



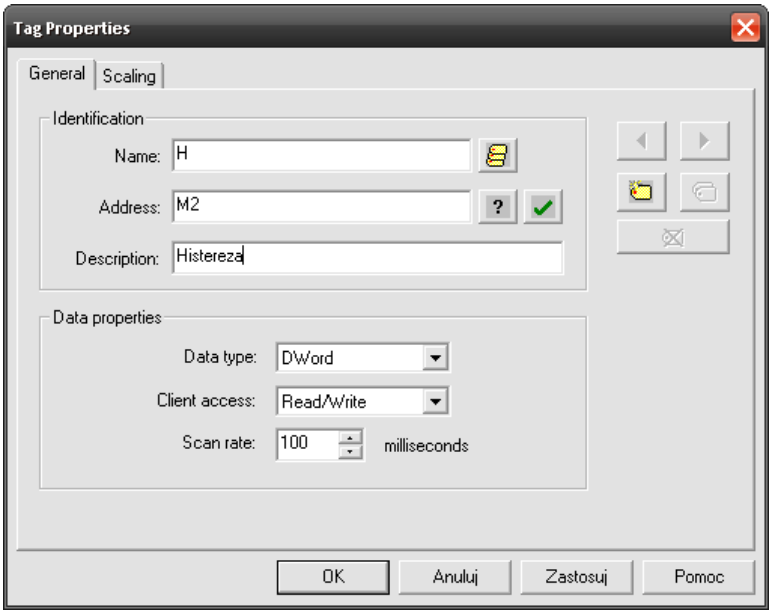
Rysunek 8.24 Dodawanie tagu SP

Na rysunku 8.24 przedstawiona jest konfiguracja tagu ***PV*** dla zmiennej ***M0***.



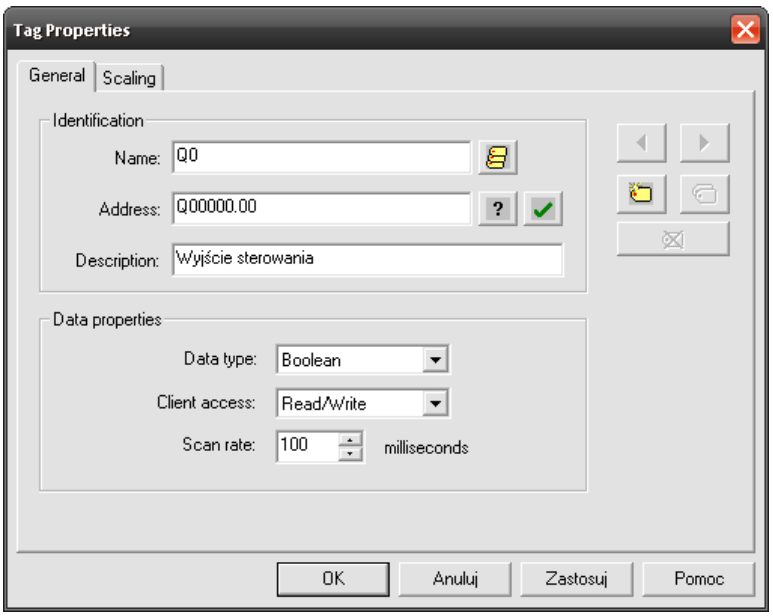
Rysunek 8.25 Dodawanie tagu PV

Na rysunku 8.24 przedstawiona jest konfiguracja tagu ***H*** dla zmiennej ***M2***.



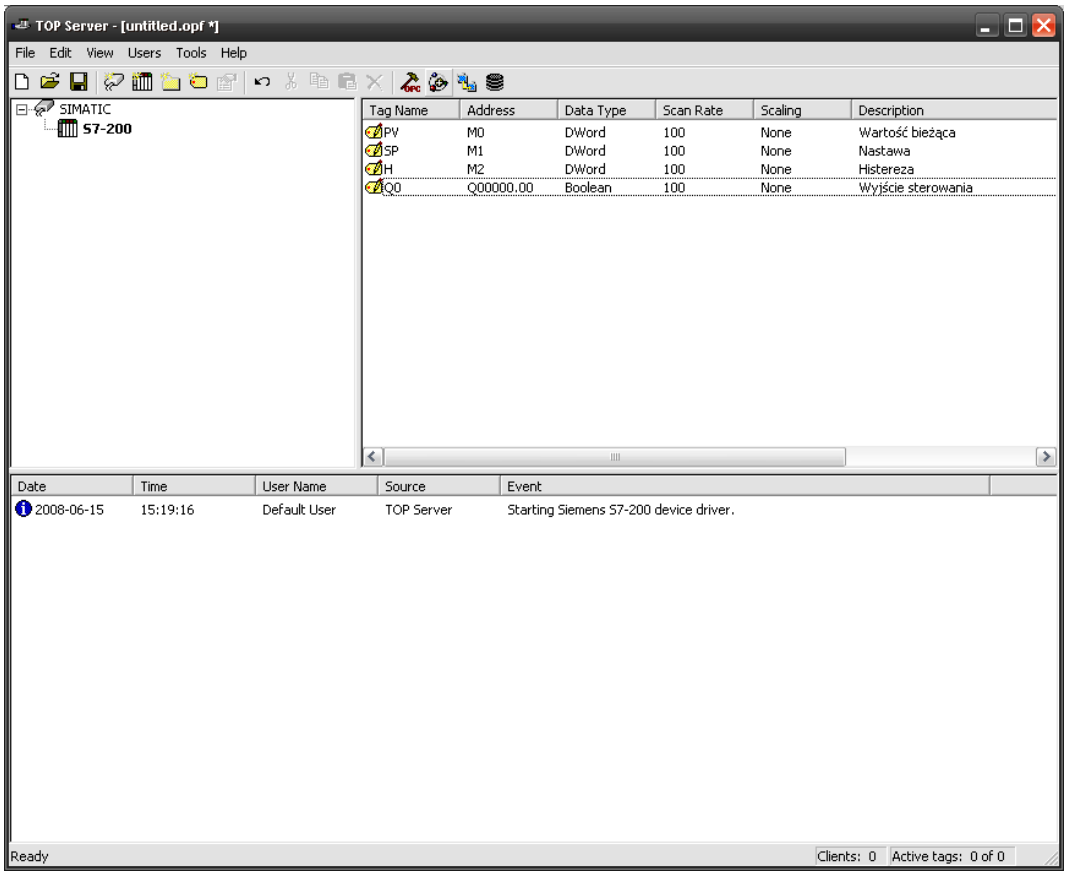
Rysunek 8.26 Dodawanie tagu H

Na rysunku 8.24 przedstawiona jest konfiguracja tagu ***Q0*** dla wyjścia ***Q0***.



Rysunek 8.27 Dodawanie tagu Q0

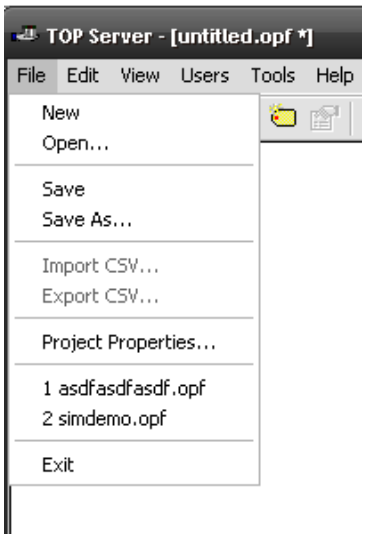
Poniżej przestawione jest okno programu z dodanymi tagami.



Rysunek 8.28 Okno programu z dodanymi tagami

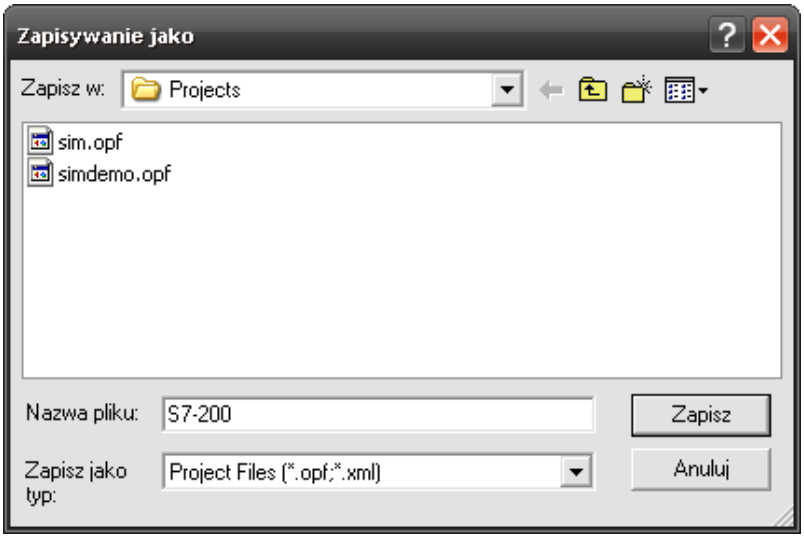
8.2.4 Zapis projektu

Aby nie stracić zmian należy zapisać projekt. W tym celu należy wybrać *"Save as..."* z menu *"File"*.



Rysunek 8.29 Zapis projektu

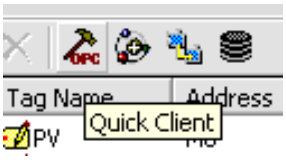
W menu dialogowym należy podać ścieżkę zapisu i nazwę pliku projektu.



Rysunek 8.30 Wybór pliku do zapisu

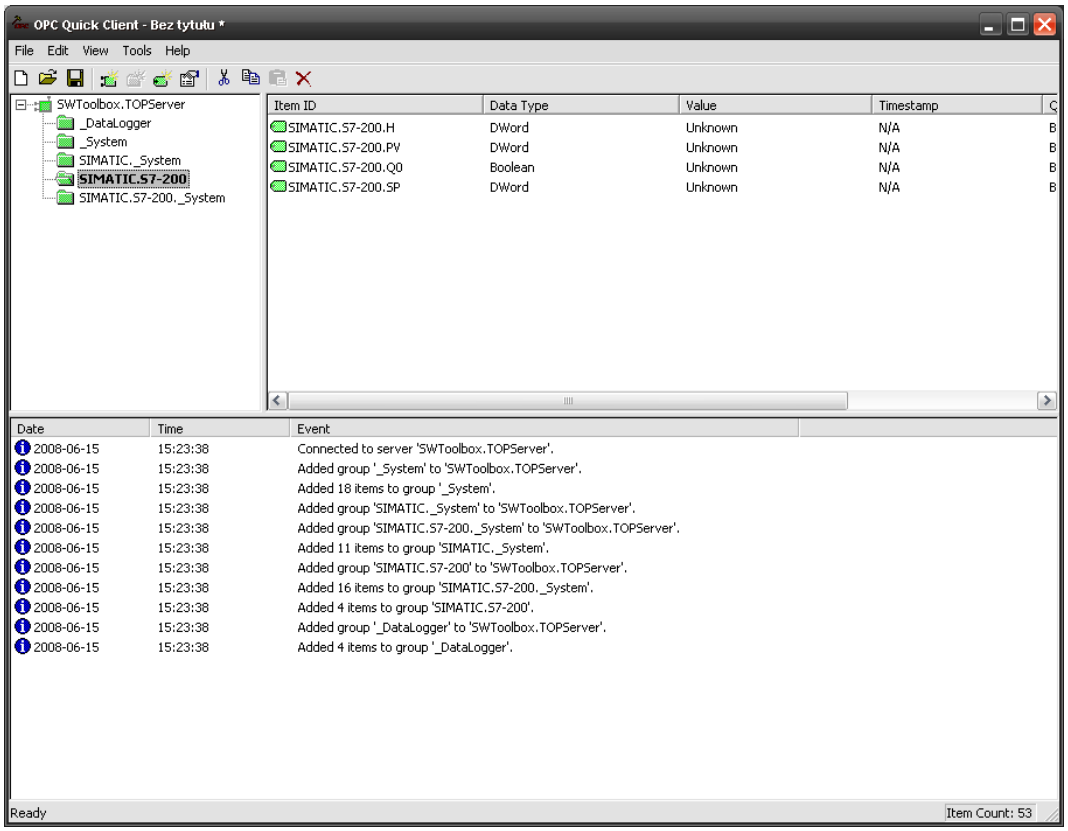
8.2.5 Klient testowy

Aby sprawdzić poprawność działania połączenia serwer dysponuje testowym klientem, który możemy uruchomić z paska narzędzi aplikacji klikając przycisk *"Quick Client"* (patrz rysunek 8.31).



Rysunek 8.31 Testowy klient OPC

W oknie klienta powinny być widoczne dostępne kanały oraz udostępniane zmienne (tagi) wraz z ich bieżącymi wartościami i stanem.



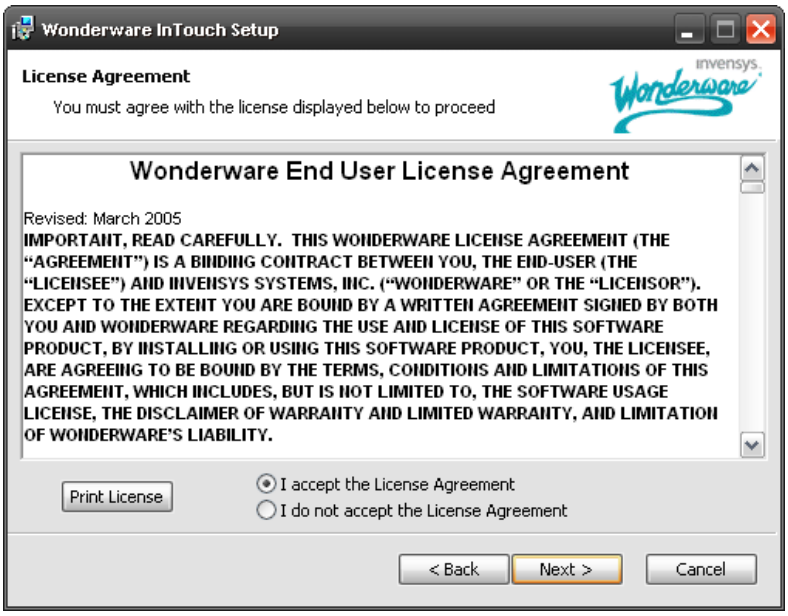
Rysunek 8.32 Klient OPC

8.3 Instalacja programu Wonderware InTouch 9.5

Instalator programu uruchomi się automatycznie po włożeniu płyty z oprogramowaniem do napędu.

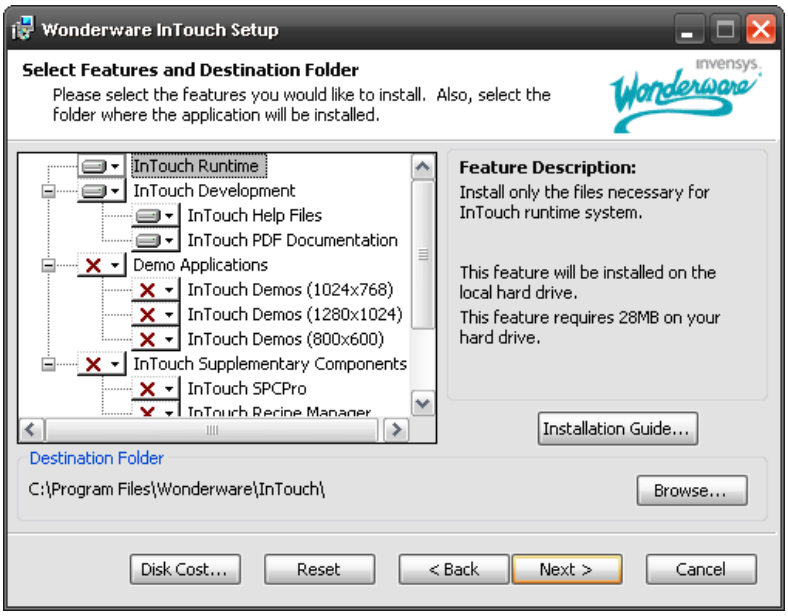
Rysunek 8.33 Instalator programu

Aby kontynuować należy przeczytać i zaakceptować licencję oprogramowania.



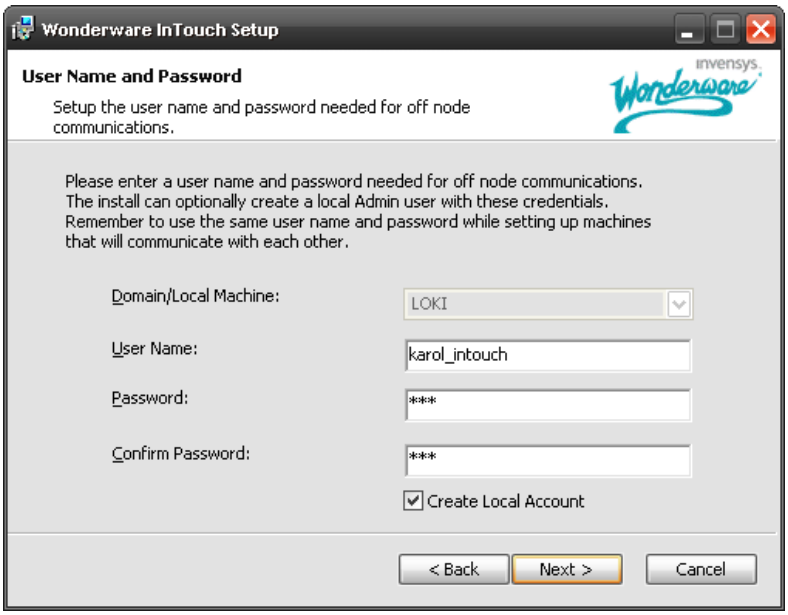
Rysunek 8.34 Akceptacja licencji

Następnie zostaniemy poproszeni o wybór komponentów oraz folderu docelowego.



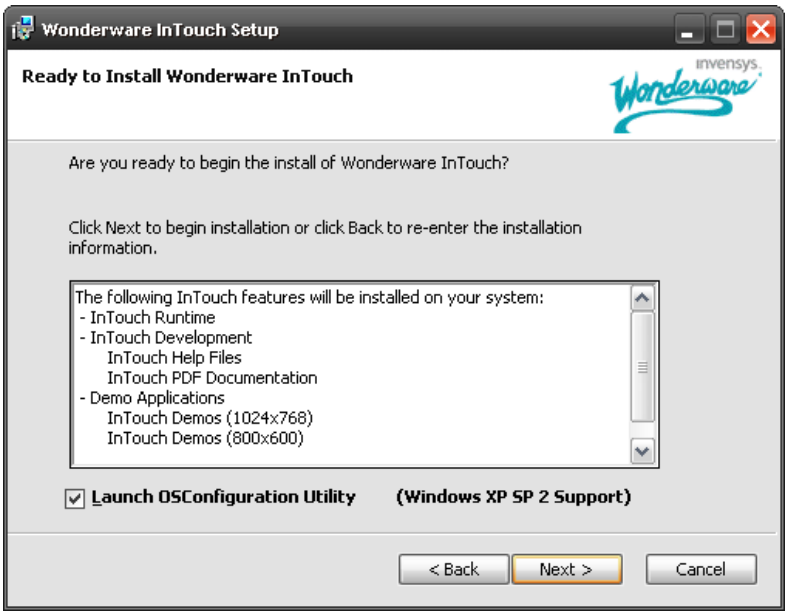
Rysunek 8.35 Wybór komponentów programu

Następnie, ze względów bezpieczeństwa zostaniemy poproszeni o utworzenia specjalnego konta użytkownika, którego używać będzie aplikacja.



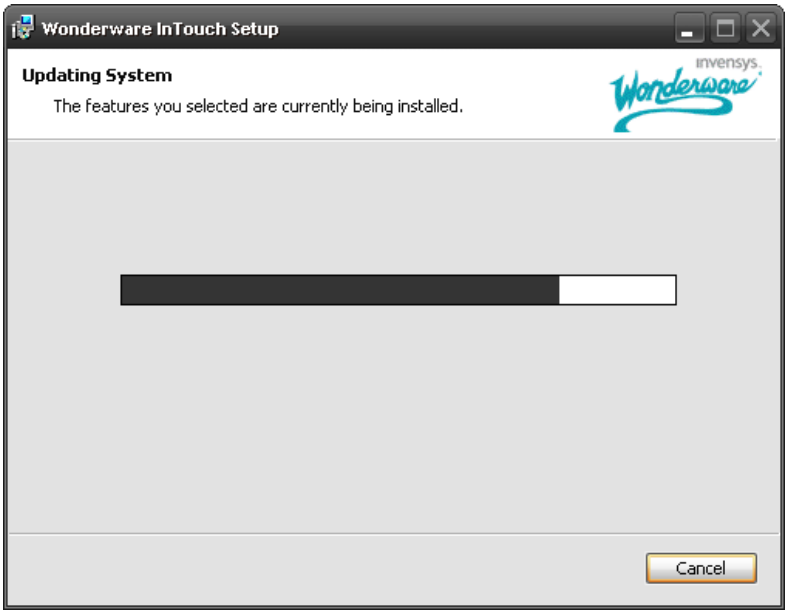
Rysunek 8.36 Zakładanie konta użytkownika

Przed rozpoczęciem procesu instalacji możemy sprawdzić wybrane ustawienia.



Rysunek 8.37 Podsumowanie konfiguracji

Po potwierdzeniu rozpocznie się kopiowanie plików.

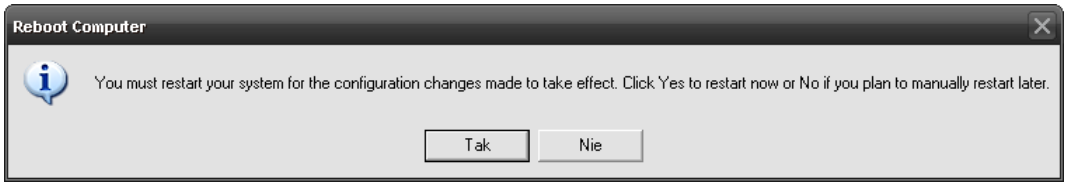


Rysunek 8.38 Instalacja programu

Po zakończeniu instalacji mamy możliwość obejrzenia pliku readme.

Rysunek 8.39 Koniec instalacji

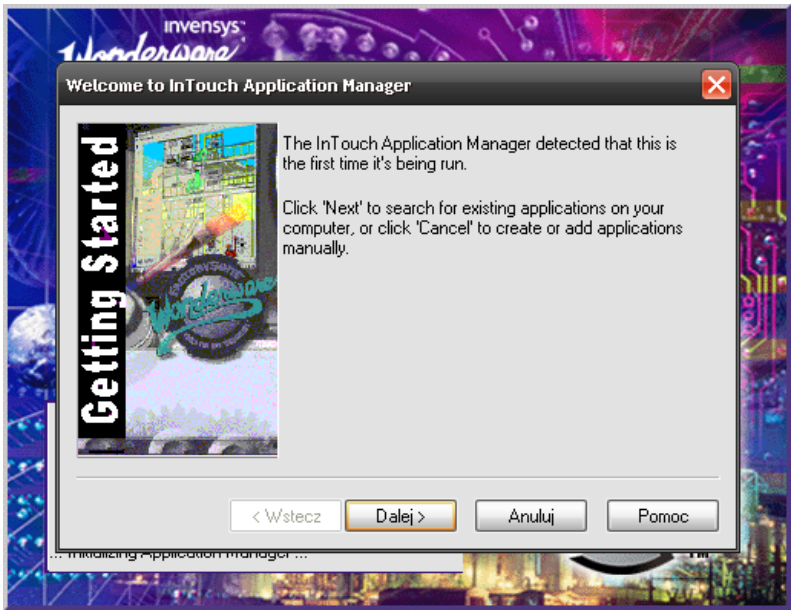
Przed uruchomieniem aplikacji zalecane jest ponowne uruchomienie komputera.



Rysunek 8.40 Ponowne uruchomienie komputera

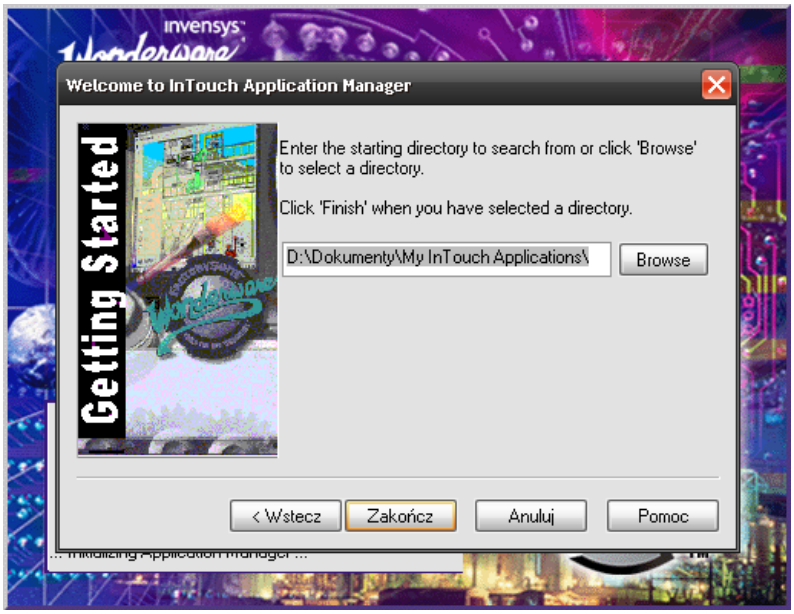
8.4 Praca z programem Wonderware InTouch 9.5

Przy pierwszym uruchomieniu programu przywita nas kreator konfiguracji.



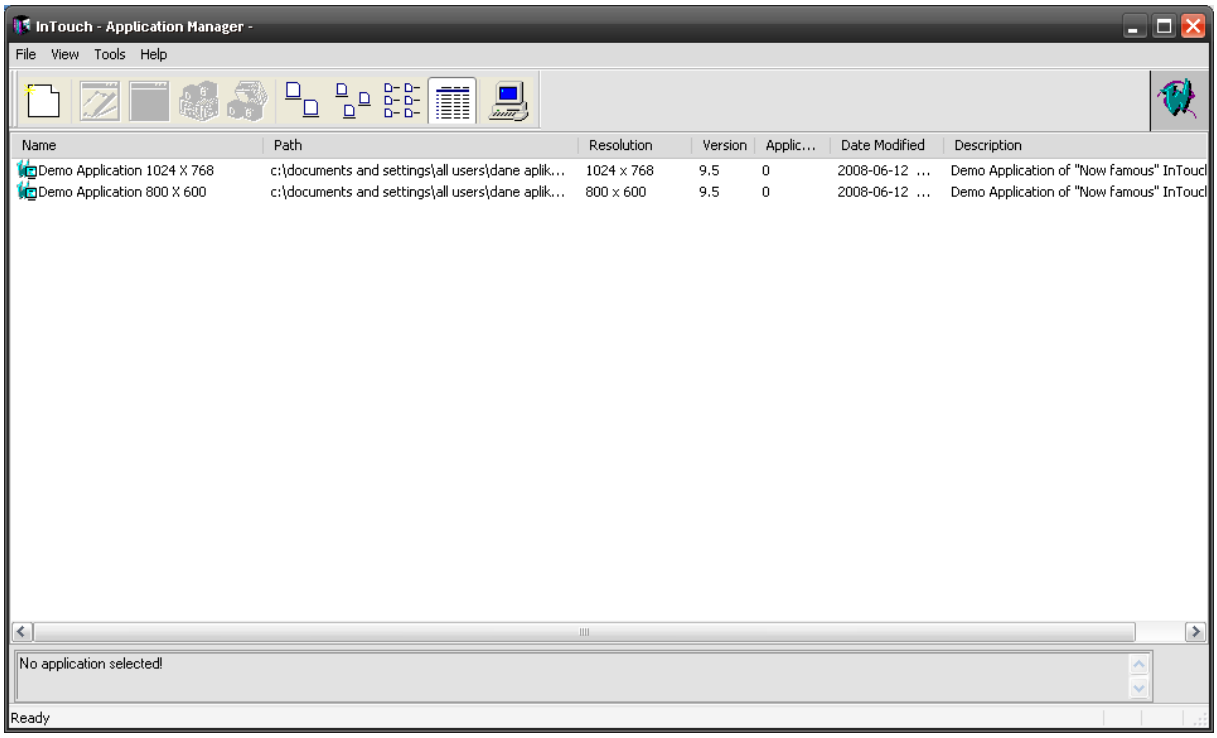
Rysunek 8.41 Menedżer aplikacji

Zostaniemy poproszeni o wybór katalogu projektów.



Rysunek 8.42 Wybór katalogu projektów

Po uruchomieniu programu pokaże nam się okno z listą dostępnych aplikacji (jeżeli zainstalowaliśmy aplikacje demonstracyjne będą one widoczne w tym oknie).



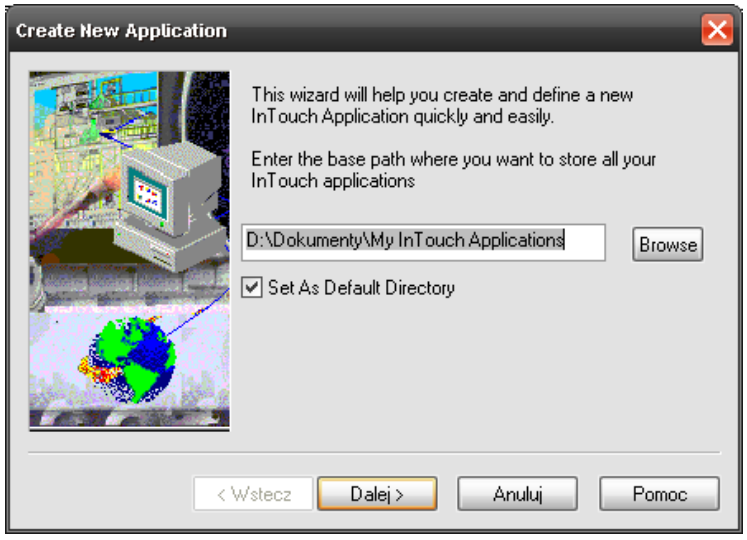
Rysunek 8.43 Okno aplikacji

8.4.1 Tworzenie projektu

Aby utworzyć projekt wybieramy opcję “New...” z menu “File”, bądź naciskamy przycisk “New” na pasku narzędzi aplikacji (patrz rysunek 8.44).

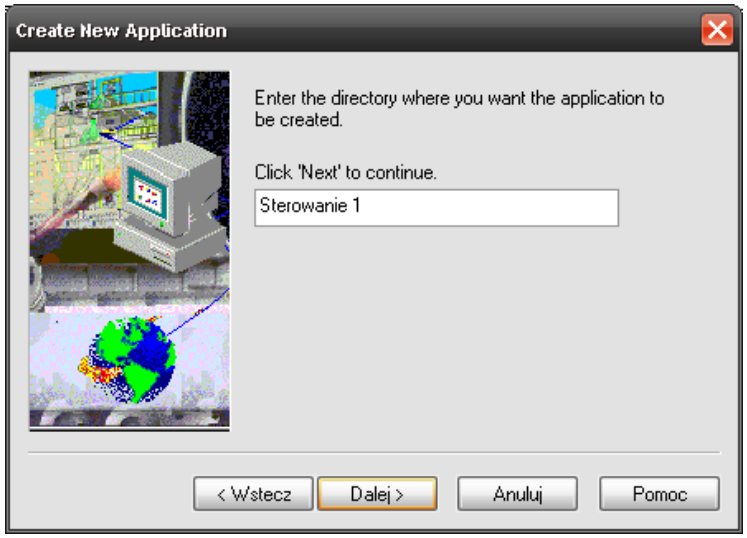
Rysunek 8.44 Pasek narzędzi

Program poprosi o wybór folderu projektów.



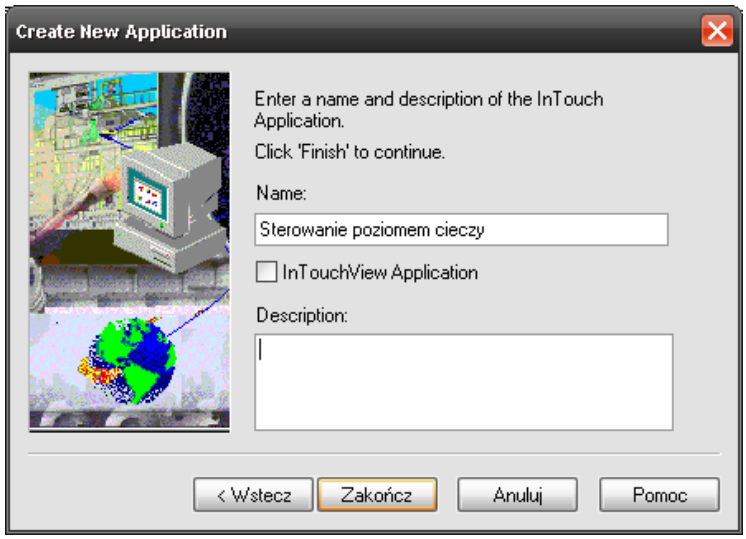
Rysunek 8.45 Wybór folderu aplikacji

Program poprosi o podanie nazwy folderu aplikacji InTouch.



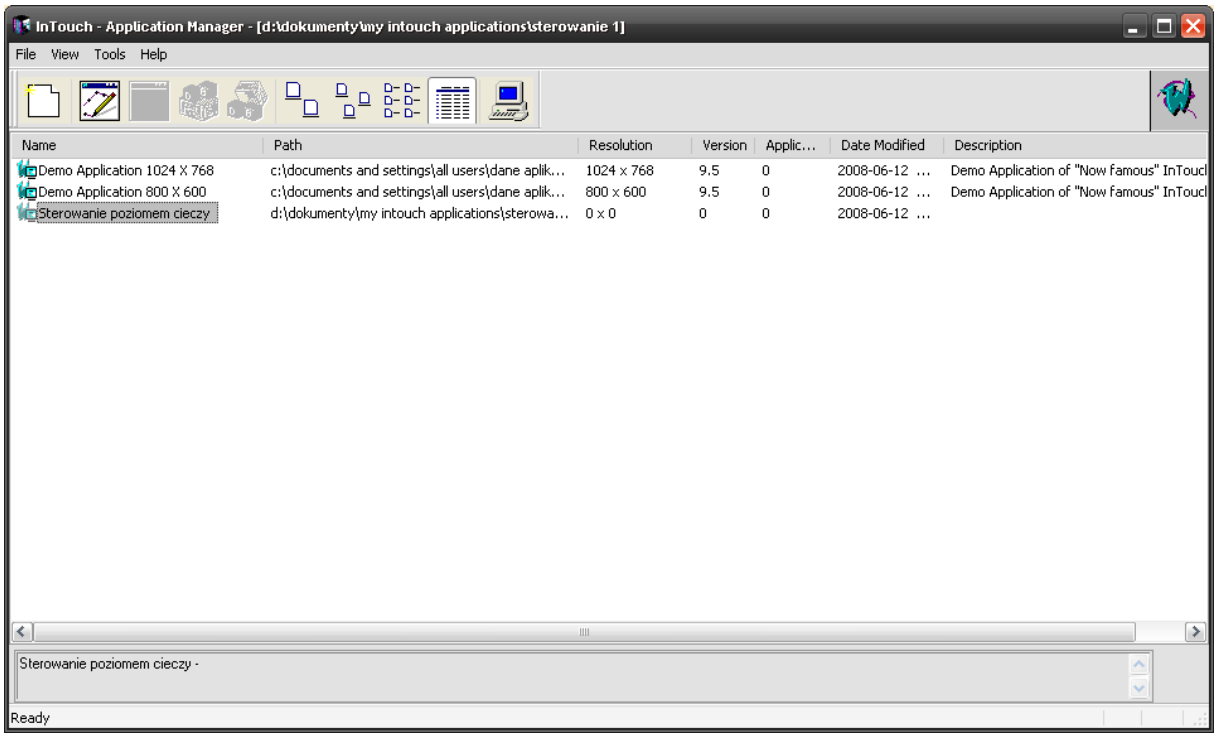
Rysunek 8.46 Wybór folderu wizualizacji

Musimy jeszcze podać nazwę i opis wizualizacji.



Rysunek 8.47 Nazwa i opis wizualizacji

Projekt został utworzony.



Rysunek 8.48 Utworzony projekt

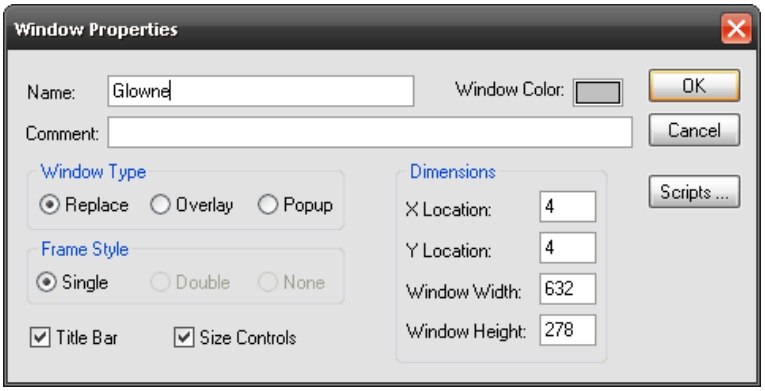
Możemy otworzyć projekt klikając dwukrotnie na jego nazwę.

8.4.2 Tworzenie okna wizualizacji

Po otworzeniu projektu należy utworzyć okno wizualizacji (jedna wizualizacja może mieć wiele okien). Aby to zrobić wybieramy opcję "New Window..." z menu "File"

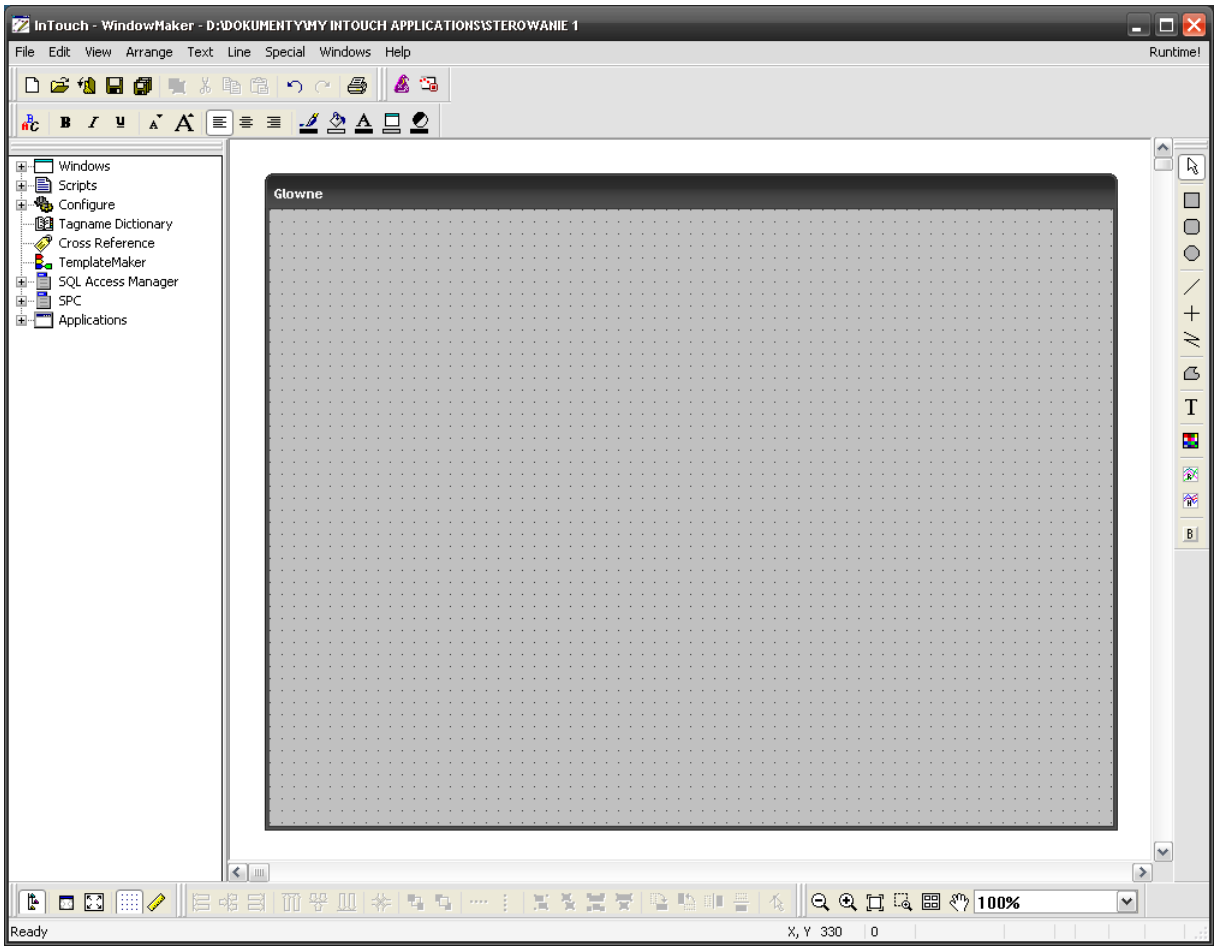
Rysunek 8.49 Tworzenie nowego okna

Otworzone zostanie okno, w którym musimy określić nazwę okna oraz wybrać jego parametry.



Rysunek 8.50 Właściwości okna

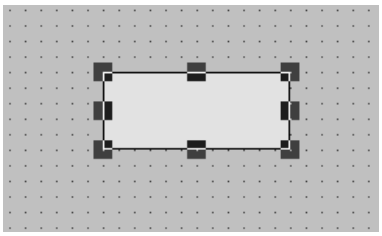
Po zakończeniu okno aplikacji powinno wyglądać następująco:



Rysunek 8.51 Nowe okno

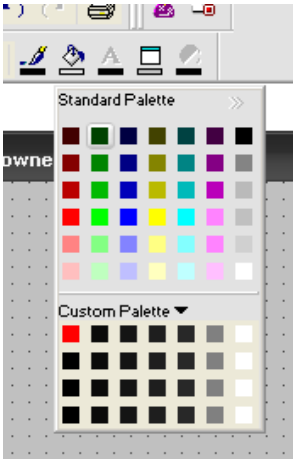
8.4.3 Tworzenie wskaźników

Używając narzędzia Rectangle ( ) tworzymy w oknie prostokąt (patrz rysunek 8.52).



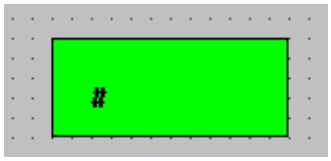
Rysunek 8.52 Rysowanie prostokąta

Wypełniamy go kolorem np. zielonym (patrz rysunek 8.53).



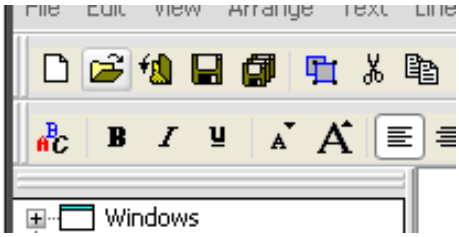
Rysunek 8.53 Wybór koloru wypełnienia

Następnie, używając narzędzia do wstawiania tekstu ( ) wstawiamy blok testowy i wypełniamy go znakiem # "hash" (rysunek 8.54).

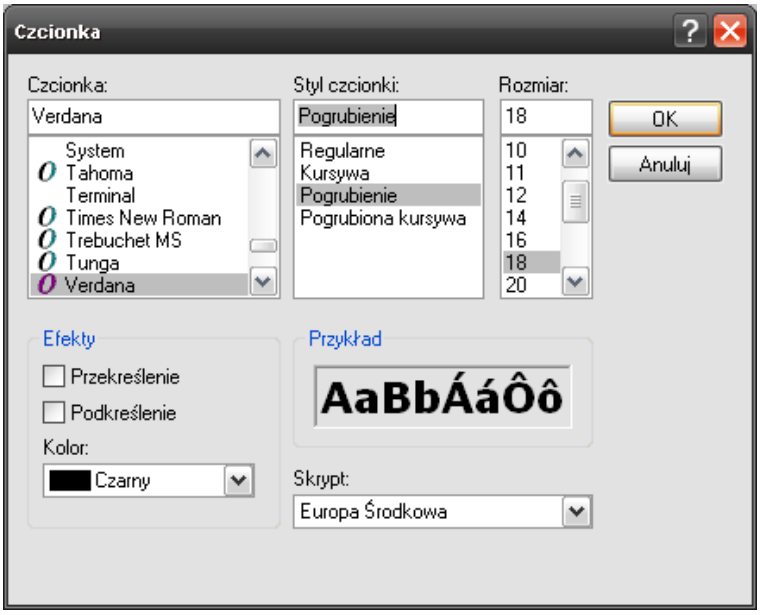


Rysunek 8.54 Prostokąt z polem tekstowym

Dostosowujemy tekst, korzystając z paska narzędzi (rysunek 8.55) lub opcji formatowania czcionki (rysunek 8.56).

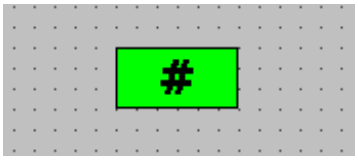


Rysunek 8.55 Pasek narzędzi formatowania tekstu



Rysunek 8.56 Okno formatowania czcionki

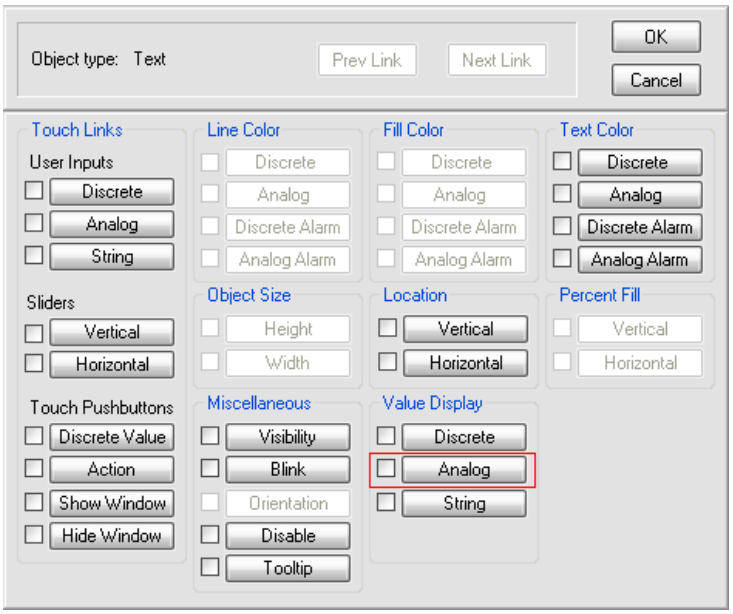
Po zakończeniu operacji wskaźnik może wyglądać tak jak na rysunku 8.57.



Rysunek 8.57 Wskaźnik

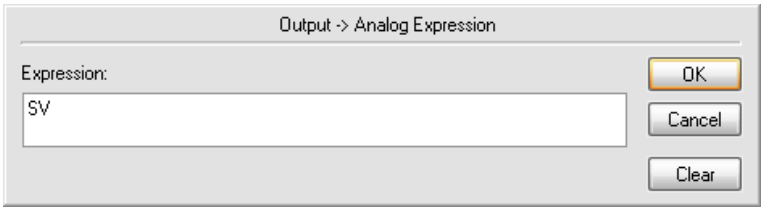
8.4.4 Podłączanie zmiennych z serwera danych

Kiedy mamy gotowy wskaźnik należy do niego przypisać zmienną z serwera OPC. W tym celu klikamy dwukrotnie na pole tekstowe. W wyniku tej operacji otworzy się okno ustawień obiektu (rysunek 8.58). Z okna wybieramy przycisk ***Analog*** znajdujący się w obszarze ***Value Display***



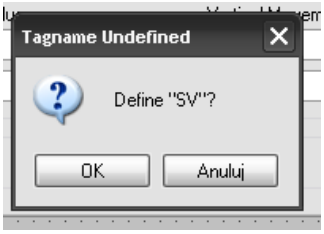
Rysunek 8.58 Konfiguracja obiektu tekstowego

Zostanie otworzone okno przypisujące wyrażenie do pola tekstowego (rysunek 8.59). Wpisujemy w to pole "SV" - Set Value.



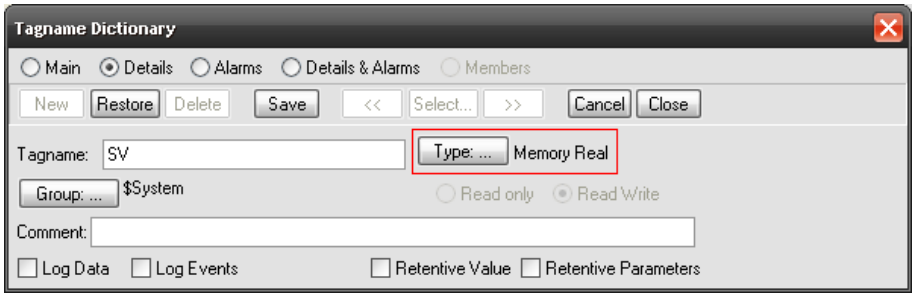
Rysunek 8.59 Przypisywanie wyrażenia

Ponieważ zmienna “SV” nie jest zadeklarowana, program zapyta czy ją utworzyć (rysunek 8.60) - wybieramy “OK”.

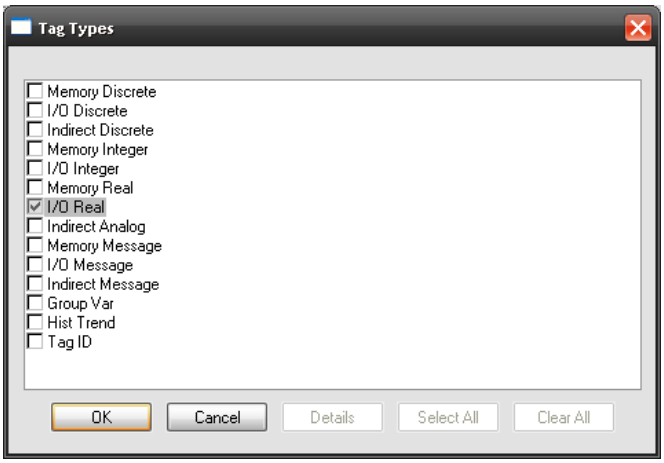


Rysunek 8.60 Deklaracja zmiennej (tagu)

Zostanie otworzone okno konfiguracji zmiennej “SV” (rysunek 8.61). Naciskamy przycisk ***Type: ...*** aby zmienić typ z ***Memory Real*** na ***I/O Real*** (patrz rysunek 8.62).

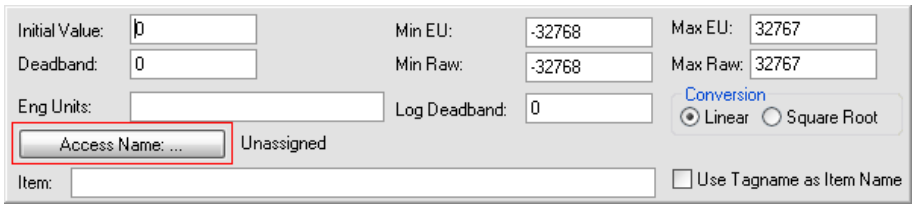


Rysunek 8.61 Okno konfiguracji zmiennej (tagu)



Rysunek 8.62 Okno zmiany typu zmiennej

W oknie konfiguracji zmiennej pojawi się nowy obszar (rysunek 8.63). Wybieramy w nim ***Access Name: ...*** aby wybrać źródło danych.

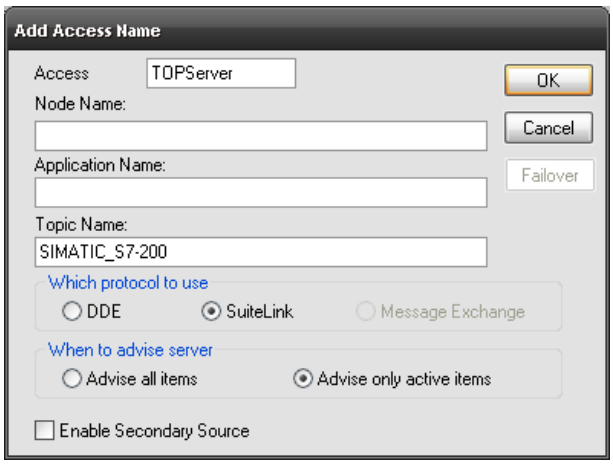


Rysunek 8.63 Okno konfiguracji zmiennej zewnętrznej

Zostanie otworzone okno konfiguracji źródeł danych (rysunek 8.64), z którego wybieramy opcję ***Add...*** (o ile zewnętrzne źródło danych nie zostało wcześniej dodane).

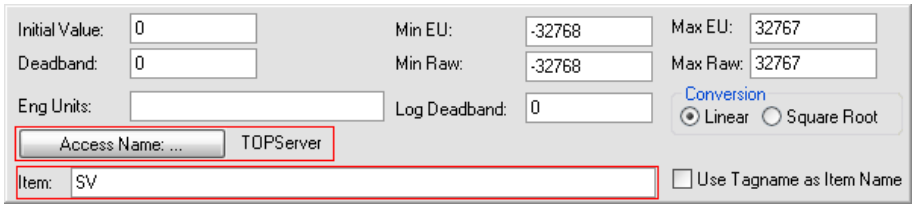
Rysunek 8.64 Okno wyboru źródła danych

Wypełniamy pola ***Access*** oraz ***Topic Name***. Pierwsze z nich może zawierać dowolny tekstowy identyfikator źródła. Drugie natomiast wypełniamy zgodnie z konfiguracją serwera danych OPC. Wartość którą należy wpisać w to pole możemy znaleźć przy pomocy klienta OPC - patrz rozdział 8.2.5. Należy jednak pamiętać, aby zamienić "." (kropkę) w nazwie Tematu na "_" (znak podkreślenia).



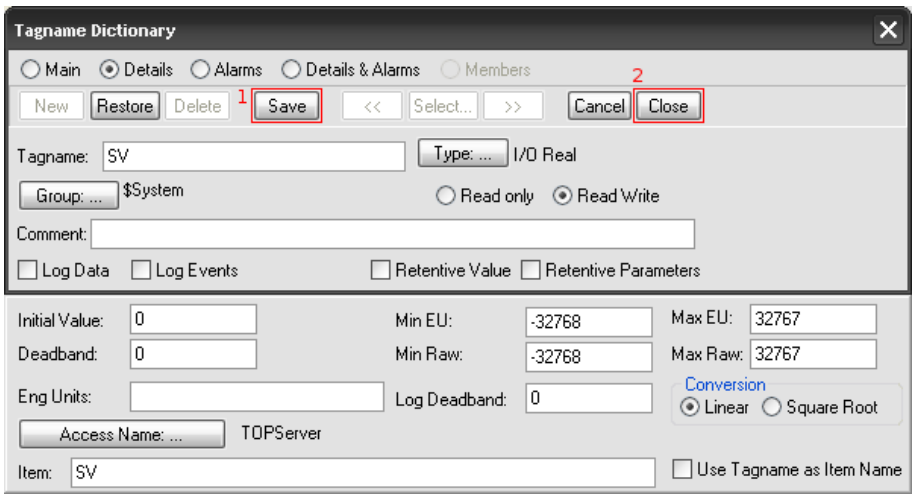
Rysunek 8.65 Dodawanie źródła danych

Po dodaniu i wybraniu źródła danych w oknie konfiguracji zmiennej, obok przycisku ***Access Name: ...*** powinna pojawić się nazwa wybranego źródła (rysunek 8.66). Aby dokończyć ustawienia należy jeszcze, w polu ***Item:*** wpisać nazwę zmiennej (tagu) serwera OPC, którą chcemy przypisać do zmiennej wizualizacji.



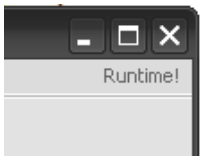
Rysunek 8.66 Przypisywanie zmiennej

Konfiguracja zmiennej została zakończona. Należy zapisać zmiany przyciskiem ***Save*** i zamknąć okno konfiguracji zmiennej przyciskiem ***Close*** (patrz rysunek 8.67).



Rysunek 8.67 Zapisywanie ustawień

Aby sprawdzić czy wszystko zostało skonfigurowane poprawnie, należy uruchomić wizualizację - do tego celu służy menu ***Runtime!*** dostępne w prawej, górnej części okna (patrz rysunek 8.68).

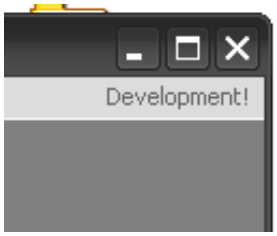


Rysunek 8.68 Przycisk ***Runtime!***

Jeżeli nie wystąpiły błędy, aplikacja powinna zostać uruchomiona a w polu tekstowym, zamiast znaku "#" powinna pojawić się liczba odczytana z serwera OPC. Uruchomioną wizualizację ilustruje rysunek 8.69.

Rysunek 8.69 Uruchomiona wizualizacja

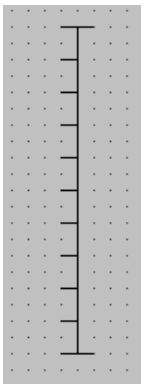
Aby powrócić do trybu projektowania, należy nacisnąć przycisk ***Development!*** znajdujący się prawej, górnej części ekranu (patrz rysunek 8.70).



Rysunek 8.70 Powrót do trybu projektowania

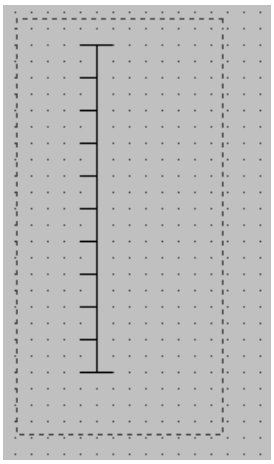
8.4.5 Tworzenie suwaków

Aby utworzyć prosty suwak z podstawowych elementów należy, korzystając z narzędzia rysowania linii () utworzyć podziałkę, przykładowa pokazana jest na rysunku 8.71.



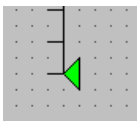
Rysunek 8.71 Podziałka

W celu stworzenia obiektu, narysowane linie należy zgrupować. Aby to zrobić należy zaznaczyć obszar w którym się one znajdują - tak jak pokazuje to rysunek 8.72 i użyć narzędzia grupowania ().



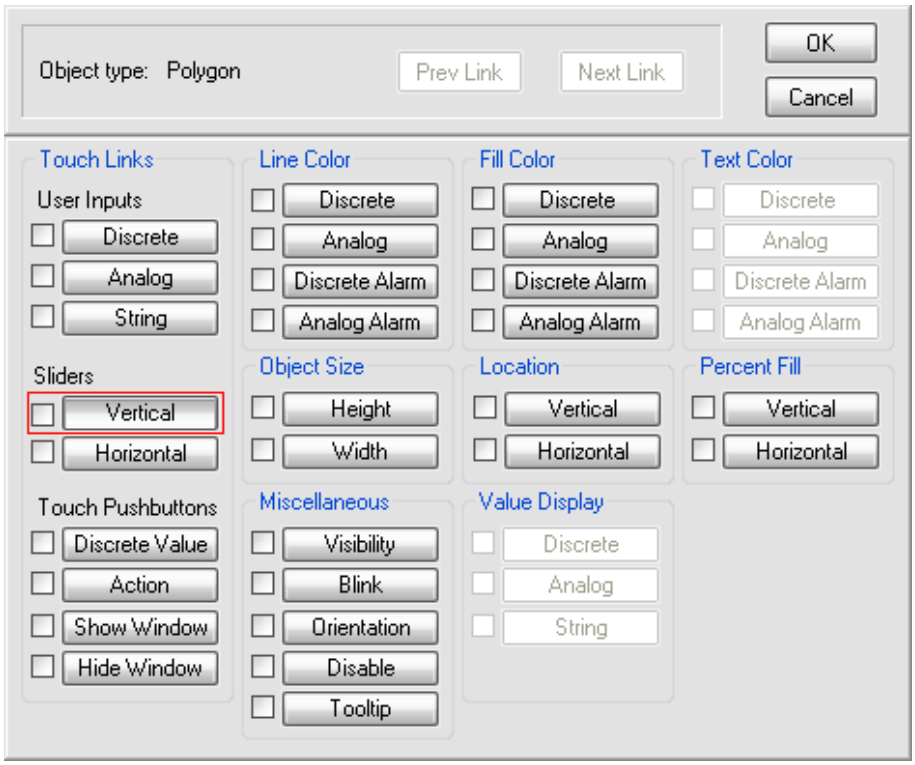
Rysunek 8.72 Zaznaczenie wszystkich elementów podziałki

Kolejnym krokiem jest utworzenie wskaźnika, który będzie przesuwany w celu ustawienia wartości. Posłużymy się narzędziem rysowania łamanej (). Przykładowy obiekt ilustruje rysunek 8.73.



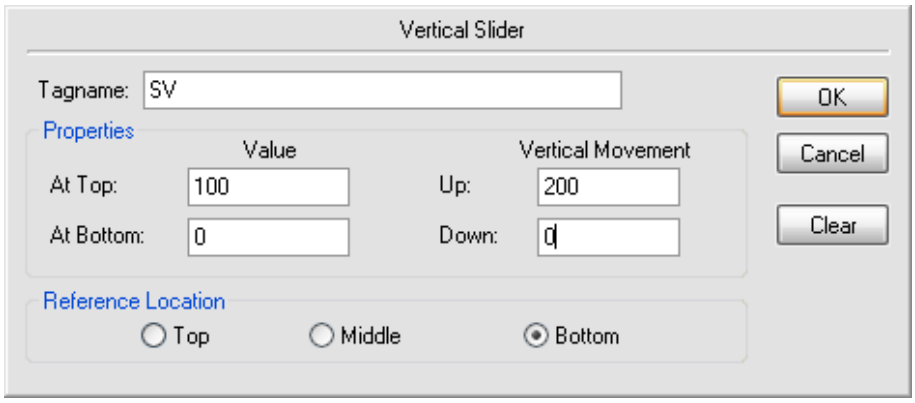
Rysunek 8.73 Suwak

Aby skonfigurować suwak, klikamy go dwukrotnie prawym klawiszem myszy, to spowoduje otwarcie okna właściwości obiektu. Z okna wybieramy przycisk ***Vertical*** znajdujący się w obszarze ***Sliders*** (patrz rysunek 8.74).



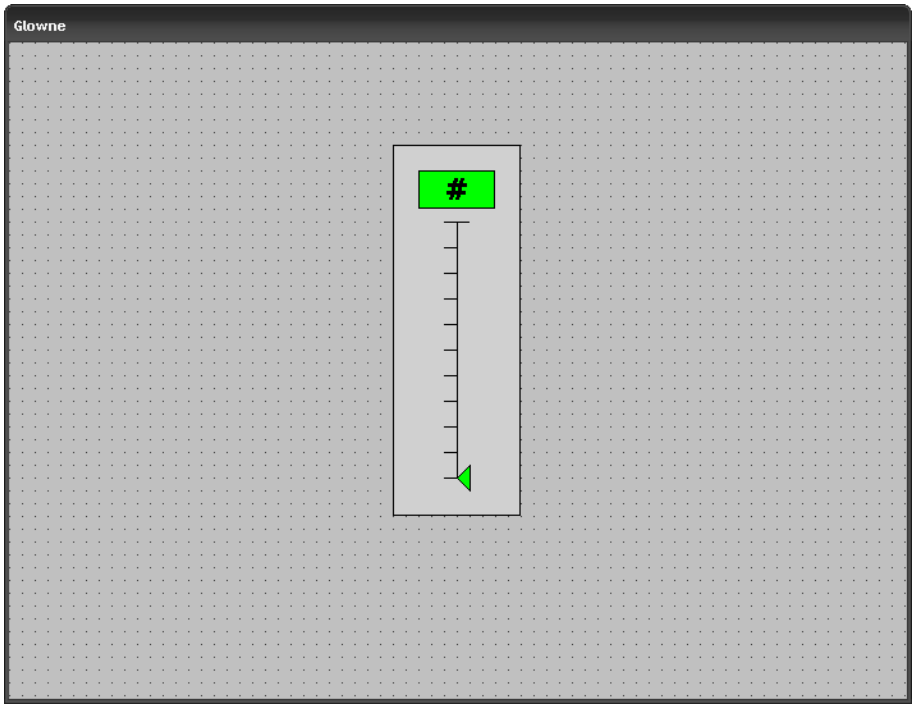
Rysunek 8.74 Konfiguracja obiektu tekstowego

Otworzone zostanie okno ustawień suwaka. W polu ***Tagname*** należy podać nazwę zmiennej lokalnej, którą chcemy ustawić (może być to zmienna zewnętrzna - np serwera OPC). Pola ***Vertical Movement*** określają zakres w jakim można przeciągać dany obiekt na pulpicie wizualizacji (względem bieżącego położenia). Pola ***Value*** określają jakie wartości przyjmuje zmienna na górze i na dole zakresu przesuwania ustalonego w polach ***Vertical Movement***.Przykładowa konfiguracja jest zilustrowana na rysunku 8.75.



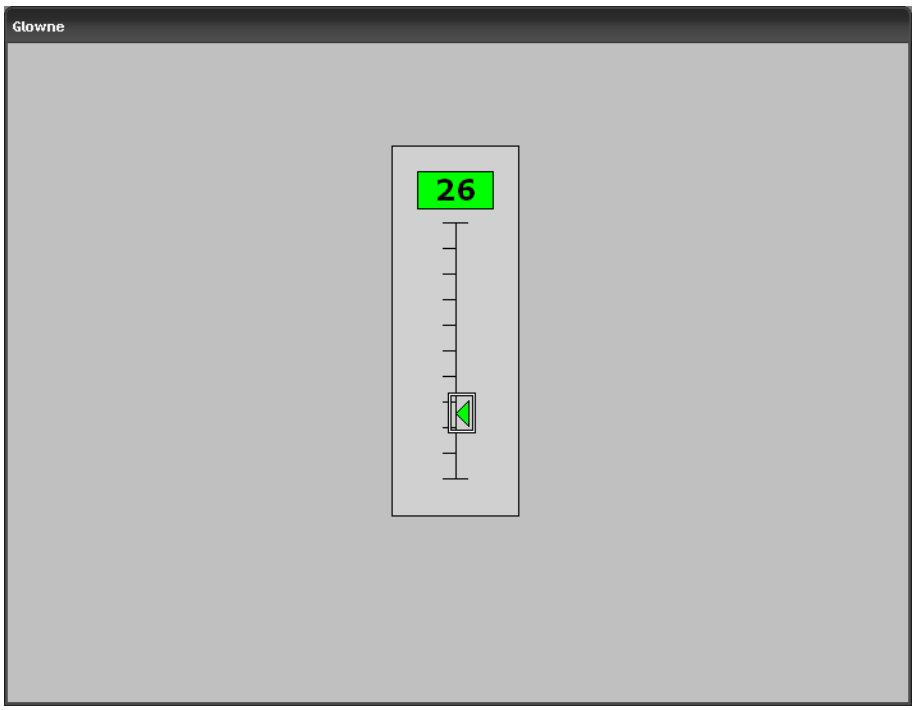
Rysunek 8.75 Konfiguracja suwaka

Utworzone elementy należy jeszcze skonfigurować względem siebie, tak aby utworzyły zadajnik. Przykładowe ustawienie przedstawia rysunek (8.76).



Rysunek 8.76 Gotowy zadajnik

Po uruchomieniu wizualizacji zadajnik powinien prezentować się tak jak na rysunku 8.77.

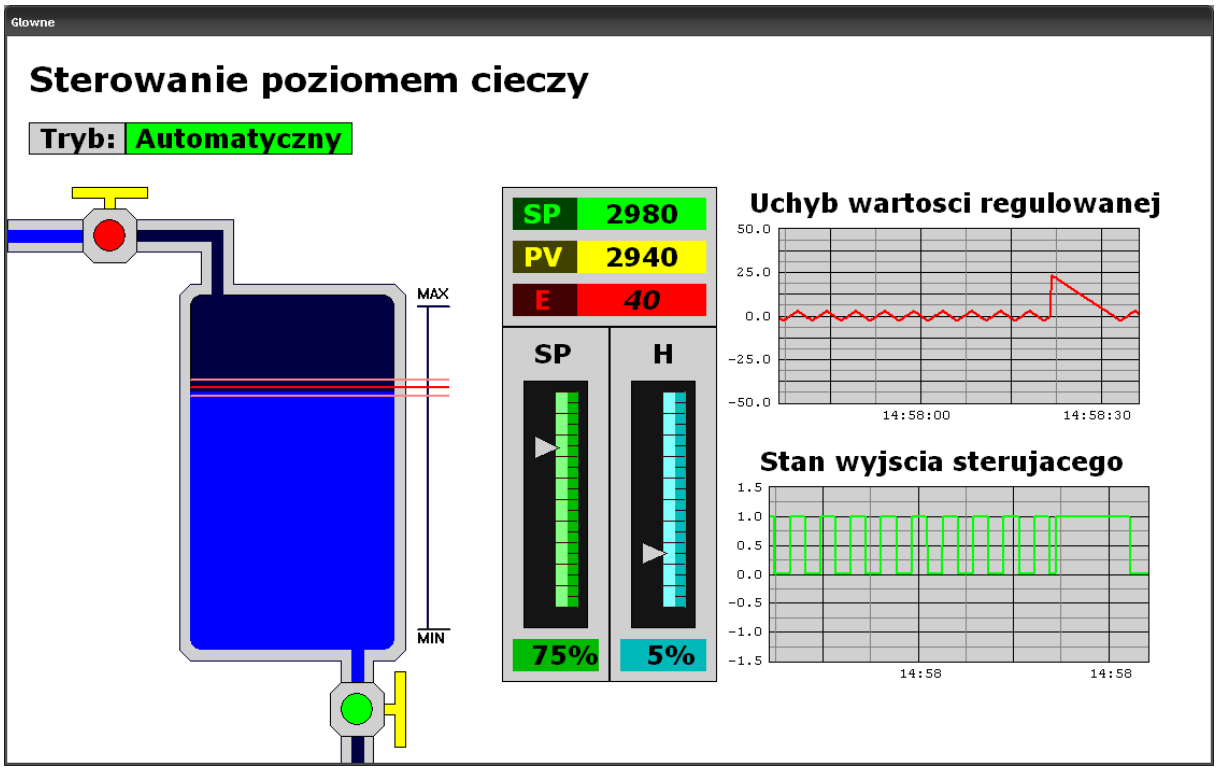


Rysunek 8.77 Uruchomiona wizualizacja

8.4.6 Gotowa aplikacja

Dla programu sterowania dwustawnego opisanego w rozdziale 7.3.3 przygotowana została bardziej skomplikowana wizualizacja. Pulpit sterowniczy umożliwia dobór wartości zadanej[3] oraz histerezy[4]. Uchyb regulacji oraz stan wyjścia sterującego wskazują dwa wykresy. Stan zaworów prezentowany jest również kolorowymi wskaźnikami.

Wygląd tej wizualizacji przestawia rysunek 8.78.



Rysunek 8.78 Gotowa wizualizacja

[3] SV - ang. Set Value
[4] H - ang. Hysteresis

Rozdział 9

Wnioski

W pracy udało się zrealizować wszystkie zadania. Przedstawiono opis podstawowych metod pomiaru poziomu cieczy, takich jak ultradźwiękowa, elektroniczna, pojemnościowa, przewodnościowa, różnicy ciśnień, hydrostatyczna czy radiometryczna. Uwzględniono również przegląd najbardziej popularnych algorytmów sterowania poziomem jak np. algorytmy sterowania dyskretnego (dwustawne i trójstawne) oraz algorytmy regulacji ciągłej - PID. Dla tych ostatnich podano metody wyznaczania parametrów obiektu (identyfikacja) oraz zasady doboru nastaw regulatorów ciągłych. Przedstawiono opis budowy mikrokontrolerów ATmega i uwzględniono takie elementy architektury jak porty wejścia/wyjścia, przerwania, układy licznikowe oraz sposoby jego programowania (flashowania). Szczegółowo opisano instalację niezbędnego oprogramowania, tworzenie i konfigurację projektu w środowisku AVRStudio, kodowanie, kompilację oraz symulację napisanego programu. Najtrudniejszą częścią pracy stanowiło wykonanie przetwornika pomiarowego, którego budowę a w szczególności architekturę, konstrukcję obwodów zasilania, wejściowcyh i wyjściowych przedstawiono w postaci schematów. Zamieszczono również schematy elektyczne i wzory płytek obwodów drukowanych stanowiące opis realizacji układu. Dla zaprojaktowanego układu laobratoryjnie wyznaczono charakterystyki przetwarzania analogowo cyfrowego oraz modulacji częstotliwości. Następnie układ został podłaczony do sterownika PLC gdzie zaimplementowane zostały algorytmy sterowania dwu- i trójstawnego. W opisie implementacji algorytmów kontroli uwzględniono także instalację oprogramowania i podstawową konfigurację sprzętową. Do sterowika wykonana został projekt stacji operatorskiej zrealizowany w środowisku Wonderware InTouch 9.5 uwględniający zarówno instalację oprogramowania wizualizacji, jak i serwera wymiany danych ze sterownikiem (TopServer).

W niniejszej pracy eksperymentalne dane uzyskane laboratoryjnie potwierdziły założenia projektowe, co potwierdzają przedstawione charakterystyki. Opisane w części teoretycznej pracy, algorytmy sterowania dwu- i trójstawnego z powodzeniem udało się zaimplementować na sterowniku S7-200, z którym współpracuje przygotowana w środowisku InTouch wizualizacja. Ze wględu na złożoność procesu wytwarzania obwodu drukowanego, finalna wersja układu działa wyłącznie na makiecie deweloperskiej dla mikrokontolerów AVR.

Bibliografia

[1] Endress+Hauser - aparatura pomiarowa. http://www.pl.endress.com/.

[2] Strona domowa programu AVRStudio. http://www.atmel.com/avrstudio/.

[3] Wikipedia, wolna encyklopedia. http://pl.wikipedia.org/.

[4] Atmel. *AVR ATmega16 Datasheet (Dokumentacja techniczna procesora Atmel ATmega16)*. Atmel Corporation, wydanie 2466p–avr–08/07, 2007.

[5] R. Baranowski. *Mikrokontrolery AVR ATmega w praktyce*. Wydawnictwo BTC, Warszawa, wydanie 1, 2005.

[6] J. Brzózka. *Regulatory i układy automatyki*. Mikom, Warszawa, wydanie 1, 2004.

[7] J. Halawa. *Wyznaczanie parametrów regulatorów na podstawie transmitancji układu zamkniętego*. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, wydanie 1, 2004.

[8] J. Halawa. *Symulacja i komputerowe projektowanie dynamiki układów sterowania*. Oficyna Wydawnicza Politechniki Wrocławskiej, wydanie 1, 2007.

[9] J. Klimesz, W. Solnik. *Urządzenia automatyki*. Wydawnictwo Politechniki Wrocławskiej, Wrocław, wydanie 1, 1991.

[10] A. Królikowski, D. Horla. *Identyfikacja obiektów sterowania : metody dyskretne*. Wydawnictwo Politechniki Poznańskiej, Poznań, wydanie 1, 2005.

[11] B. Łysakowska, G. Mzyk. *Komputerowa symulacja układów automatycznej regulacji w środowisku MATLAB/Simulink*. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, wydanie 1, 2005.

[12] K. Mańczak, Z. Nahorski. *Komputerowa identyfikacja obiektów dynamicznych*. Państwowe Wydawnictwo Naukowe, Warszawa, 1983.

[13] P. Prof. Zoran Vikic. *Lectures on PID controllers*. University of Zagrzeb, 2002. http://arri.uta.edu/acs/jyotirmay/EE4343/Labs_Projects/pidcontrollers.p%df.

[14] Siemens. *SIMATIC S7-200 Programmable Controller System Manual*. Siemens AG, 2005.

[15] W. yong Han, J. wook Han, C. goo Lee. *Development of a Self-tuning PID Cntroller based on Neural Network for Nonlinear Systems*. Dept.of Electrical Engineering, Jeonju Technical College, 1999. http://med.ee.nd.edu/MED7/med99/papers/MED182.pdf.

[16] Z. Zajda, L. Żebrowski. *Urządzenia i układy automatyki.* Wydawnictwo Politechniki Wrocławskiej, Wrocław, 1993.

Dodatek A

Kod programu przetwornika

```
1 #include <inttypes.h>
2 #include <avr/io.h>
3 #include <avr/interrupt.h>
4 #include <util/delay.h>
5 #include "dm_lcd.h"
6
7 #define PWM_out(value)          OCR1A=value
8
9
10 volatile uint16_t freq;              // zmienna okreslajaca czestotliwosc
11 volatile uint16_t adcvalue;            // przerworzona wartosc z ADC
12 uint16_t params[10];                // tablica parametrow programu
13
14 SIGNAL(SIG_ADC) {                  // przerwanie z przetwornika ADC
15   adcvalue = ADCW;                 // czytaj wartosc z przetwornika ADC
16 }
17
18 void lcd_puti(uint8_t value) {           // wyswietla liczbe 8 bitowa na LCD
19   char string[3];
20   if ( value < 10 ) { lcd_puts(" "); }
21   if ( value < 100 ) { lcd_puts(" "); }
22   itoa(value, string, 10);
23   lcd_puts(string);
24 }
25
26 void lcd_putd(uint16_t value) {          // wyswietla liczbe 16 bitowa na
     LCD
27   char string[5];
28   if ( value < 10 ) { lcd_puts(" "); }
29   if ( value < 100 ) { lcd_puts(" "); }
30   if ( value < 1000 ) { lcd_puts(" "); }
31   //if ( value < 10000 ) { lcd_puts(" "); } // i tak wykorzystujemy liczby
        max 10b
32   itoa(value, string, 10);
33   lcd_puts(string);
34 }
35
36 int kbb_read() {                   // obsluga klawiatury
37
38   uint8_t read;                  // inicjalizacja zmiennej tymczasowej
39   read = ( PINC & DDRC );             // odczyt portu C, maskowanie rejesterm
        kirunkowym
40
```

```
41    if (read ==  4) { return 1; }        // zwracanie odczytu
42    if (read ==  8) { return 2; }
43    if (read == 16) { return 3; }
44    if (read == 32) { return 4; }
45
46    return 0;
47  }
48
49  void pwm_init(void) {                  // inicjalizacja PWM
50    // Ustaw licznik 1 w trybie 10 bitowego wyjscia PWM
51    //TCCR1A = _BV(COM1A1)|_BV(COM1B1)|_BV(WGM10)|_BV(WGM11);    // 10bit pwm
52    TCCR1A = _BV(COM1A1)|_BV(COM1B1)|_BV(WGM11);      // 10bit fast pwm
53
54    // Ustaw czestotliwosc licznika zgodna z czestotliwoscia taktownia zegara
55    //TCCR1B = _BV(CS10);
56    TCCR1B = _BV(CS10)|_BV(WGM12)|_BV(WGM13);
57
58    ICR1 = 1;                        // prescaler
59  }
60
61  void adc_init(void) {                  // inicjalizacja ADC
62    // wlacz przetwornik adc i uruchom generowanie przerwan
63    // czestotilwosc taktowania F_ADC=F_CPU/64(125 kHz)
64    ADCSRA = _BV(ADEN)|_BV(ADIE)|_BV(ADPS2)|_BV(ADPS1);
65
66    //  Bit | Nazwa | Opis
67    //   0  | ADPS0 | preskaler
68    //   1  | ADPS1 | preskaler
69    //   2  | ADPS2 | preskaler
70    //   3  | ADIE  | generowanie przerwania przez przetwornik
71    //   4  | ADIF  | znacznik przerwania z przetwornika
72    //   5  | ADFR  | przetwarzanie samodzielne (Free Run)
73    //   6  | ADSC  | start konwersji - jesli jest ustawiony ten bit oraz
          ADEN
74    //   7  | ADEN  | wlaczenie przetwornika (AD Enable)
75    //
76    // Preskaler:
77    // ADPS2  ADPS1  ADPS0  podzial
78    // ---------------------------
79    //   0      0      0       1
80    //   0      0      1       2
81    //   0      1      0       4
82    //   0      1      1       8
83    //   1      0      0      16
84    //   1      0      1      32
85    //   1      1      0      64
86    //   1      1      1     128
87
88    // kanal przetwornika
89    ADMUX = 0;
90  }
91
92  void menu(int m) {                   //obsluga menu programu
93    lcd_clrscr();
94    switch (m) {
95      case 0 :
96        lcd_gotoxy(0,0);
97        lcd_puts("wyp PWM: ");
98        lcd_gotoxy(12,0);
```

```
99        lcd_putd(params[m]);
100       return;
101     case 1 :
102       lcd_gotoxy(0,0);
103       lcd_puts("adcvalue: ");
104       lcd_gotoxy(12,0);
105       lcd_putd(adcvalue);
106       return;
107     case 2 :
108       lcd_gotoxy(0,0);
109       lcd_puts("wejscie: ");
110       lcd_gotoxy(12,0);
111       lcd_puti(TCCR1A);
112       return;
113   }
114 }
115
116 int main(void) {
117   // inicjalizacja peryferiow
118   lcd_init(LCD_DISP_ON);              // wyswietlacza
119   adc_init();                       // przetwornika A/C
120   pwm_init();                       // wyjscia PWM
121
122   // konfiguracja portow
123   DDRC = _BV(2)|_BV(3)|_BV(4)|_BV(5);
124   PORTC = 0x00;
125   DDRD = 0xff;
126   PORTD = 0x00;
127
128   // deklaracja i inicjalizacja zmiennych
129   char key, key_;
130   params[0]=512;
131   uint8_t refresh = 0, menuitem = 0;
132   uint16_t value = 0, keycount = 0, softdelay = 0, adcvalue_ = 0;
133
134   sei();                          // wlacz obsluge przerwan
135   menu(menuitem);                   // wyswietl menu
136
137   while(1) {
138
139     key_ = key;
140     key = kbb_read();
141
142     if (key == key_ && key != 0) {
143       keycount++;
144
145       if (keycount >= 1500) {
146         refresh = 1;
147         keycount = 0;
148         value = 16;
149       }
150     }
151     else {
152       keycount = 0;
153       value = 1;
154     }
155
156
157     if ( (key && key != key_) || (refresh) ) {
```

```
158
159         if (key == 1) { menuitem--;}
160         if (key == 2) { menuitem++;}
161
162         if (key == 3) { params[menuitem]-=value;}
163         if (key == 4) { params[menuitem]+=value;}
164
165         menuitem = menuitem % 3;
166         menu(menuitem);
167         refresh = 0;
168       }
169
170
171       if (99 == softdelay%100 ) {
172         ADCSRA |= _BV(ADSC);
173       }
174
175       if (softdelay == 2000) {
176         //refresh = 1;
177         softdelay = 0;
178       }
179
180       PWM_out(params[0]);
181       OCR1B=adcvalue;
182
183       softdelay++;
184     }
185 }
```

Dodatek B

Zawartość załączonej płyty CD

Wykaz katalogów wraz zawartością:

- **dokumenty** - dokumenty i specyfikacje wykożystane przy realizacji pracy
  - **AVR**
  - **avrlibc**
  - **InTouch**
  - **Karty Katalogowe**
  - **Karty Katalogowe (Czujniki)**
  - **LaTeX**
  - **Siemens**
  - **sterowanie**
- **instalki** - pliki instalacyjne omawianych programów
- **praca** - źródła pracy w fotmacie LaTeX
- **projekty** - projekty wykonane w trakcie realizacji pracy
  - **AVR Studio**
  - **Eagle**
  - **InTouch**
  - **kontrollerlab**
  - **MATLAB**
  - **OrCAD**
  - **S7-200**
  - **TOPServer**
- **zrodla** - źródła programów