przemysł chemiczny

MIESIĘCZNIK NAUKOWO-TECHNICZNY ZAŁOŻONY W 1917 ROKU ■ ORGAN STOWARZYSZENIA INŻYNIERÓW I TECHNIKÓW PRZEMYSŁU CHEMICZNEGO I POLSKIEJ IZBY PRZEMYSŁU CHEMICZNEGO

NUMER 6/2012 TOM 91 czerwiec 2012

Modbus

Wykorzystanie programu Auma Modbus Tester firmy Proloc do komunikacji i diagnozowania komunikacji Modbus RTU

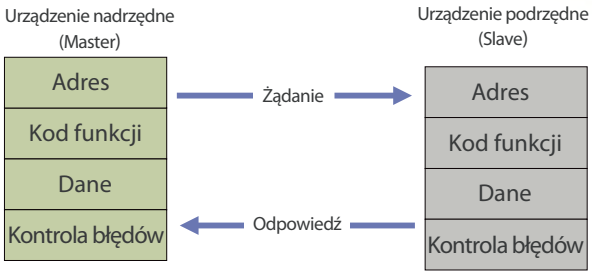
Protokół Modbus został opracowany w 1979 r. przez firmę Modicon. Komunikacja za jego pośrednictwem polega na wymianie komunikatów w układzie nadrzędny-podrzędny (*master-slave*). Komunikat może zawierać żądanie wykonania polecenia (przesyłane przez jednostkę *master*) albo odpowiedź na żądanie (przesyłane przez jednostkę *slave*), para żądanie-odpowiedź nazywana jest transakcją. Modbus jest w pełni otwartym standardem i jednym z najczęściej stosowanych protokołów sieciowych używanych w przemyśle. Współpracuje z interfejsem RS485.

Urządzenia wyposażone w możliwość komunikacji za pośrednictwem protokołu Modbus używają techniki *master-slave*, w której tylko jedno urządzenie (*master*) inicjuje przesyłanie komunikatów (żądań). Inne urządzenia (*slaves*) odpowiadają, wysyłając żądane dane do urządzenia nadrzędnego albo podejmują działania określone w odebranym komunikacie (rys.). Urządzeniem typu *slave* może być każde urządzenie zewnętrzne, np. zawór, przetwornik pomiarowy lub sterownik PLC, które przetwarza żądania i wysyła na nie odpowiedzi.

Jednostka *master* może wysłać komunikaty indywidualnie do każdej zaadresowanej jednostki *slave* bądź też wysyłać komunikaty w trybie rozgłoszenia (*broadcast*) do wszystkich jednostek *slave*. Urządzenie podrzędne przesyła odpowiedź na żądanie tylko, jeżeli zostało indywidualnie zaadresowane. W żądaniu wysłanym przez jednostkę *master* są umieszczone następujące pola: adres jednostki *slave*, kod funkcji, którą ma wykonać jednostka *slave* (np. zapis, odczyt wartości rejestru) i dane niezbędne do wykonania żądanej funkcji (np. wartość, którą należy wprowadzić do odpowiedniego rejestru, numer rejestru początkowego, z którego należy odczytać wartość), a także informacje o kontroli błędów, pozwalające jednostce *slave* sprawdzić poprawność odebranych danych. Komunikat odpowiedzi jednostki *slave* zawiera następujące pola: adres jednostki *slave*, kod funkcji, która miała być wykonana, dane zawierające wynik wykonanej funkcji oraz informacje o kontroli błędów pozwalające sprawdzić jednostce *master* poprawność odebranych danych.

Jeżeli urządzenie podrzędne odpowiada poprawnie, wtedy dwa pierwsze pola są odbiciem pól wysłanych przez urządzenie nadrzędne. Jeżeli powstanie błąd, wtedy urządzenie *slave* zmienia kod funkcji w celu powiadomienia o wystąpieniu błędu i w polu danych zwraca opisujące go informacje.

Urządzenie z zaimplementowanym protokołem Modbus zawiera zazwyczaj mapę rejestrów (*register map*), na której są wykonywane różne operacje w celu monitoringu, konfiguracji albo sterowania. Dzięki takiej konstrukcji każde urządzenie jest widziane jako zbiór rejestrów.



Rys. Wymiana komunikatów pomiędzy jednostkami *master-slave*, cykl żądanie-odpowiedź

Transmisja szeregowa w sieciach Modbus

Transmisja szeregowa może odbywać się w trybie ASCII lub RTU. Kiedy urządzenie jest ustawione do komunikacji w trybie ASCII (*american standard code for information interchange*), każdy 8-bitowy bajt wiadomości jest przesyłany jako dwa znaki ASCII. Główną zaletą tego trybu jest to, że pozwala on na maksymalnie 1-sekundowe odstępy czasowe pomiędzy przesyłanymi znakami, bez generowania błędu. Format bajtu w trybie ASCII jest następujący: kodowanie szesnastkowe, jeden znak szesnastkowy zawarty w każdym znaku komunikatu ASCII; w bajcie 1 bit startu, 7 bitów danych, 1 bit parzystości oraz 1 bit stopu; sprawdzanie błędów LRC.

Kiedy urządzenie korzysta z trybu RTU (*remote terminal unit*), każdy 8-bitowy bajt w komunikacie zawiera dwa 4-bitowe znaki szesnastkowe. Główną zaletą trybu RTU jest lepsza przepustowość w stosunku do trybu ASCII przy tej samej prędkości łącza. Każdy komunikat musi być przesłany ciągłym strumieniem. Format bajtu w trybie RTU jest następujący: kodowanie to dwa znaki szesnastkowe zawarte w każdym 8-bitowym polu komunikatu; w bajcie 1 bit startu, 8 bitów danych, 1 bit parzystości oraz 1 bit stopu; sprawdzanie błędów CRC.

W celu zaznaczenia początku oraz końca wiadomości każdy komunikat przesyłany za pośrednictwem protokołu Modbus jest zawarty w tzw. ramce wiadomości (*message frame*). Dzięki ramce urządzenie jest w stanie określić, czy komunikat jest kierowany do niego oraz czy przesłana wiadomość jest kompletna. Każda ramka komunikatu jest umieszczona przez urządzenie nadające w ramce danych (*data frame*), do której jest dodawany bit startu, stopu oraz kontroli parzystości. W zależności od trybu transmisji urządzenie nadające wysyła różne ramki.

Ramki komunikatów

Komunikat w trybie ASCII rozpoczyna się znakiem dwukropka (:), a kończy znakiem Enter (CRLF). Dozwolone znaki w innych polach ramki są transmitowane w kodzie szesnastkowym. Urządzenia w sieci monitorują wiadomości pod kątem znaku początku i kiedy taki odbiorą, dekodują pole „adres". Jeżeli pole „adres" zgadza się z adresem urządzenia, wtedy następuje dekodowanie reszty wiadomości. Typowa ramka wiadomości w trybie ASCII jest przedstawiona w tabeli 1.

W trybie RTU komunikat rozpoczyna się „ciszą na łączu" trwającą co najmniej 3,5 · (czas trwania pojedynczego znaku). Po upływie tego

Tabela 1. Typowa ramka wiadomości w trybie ASCII

Start	Adres	Funkcja	Dane	Kontrola LRC	Koniec
Znak :	2 znaki	2 znaki	N znaków	2 znaki	2 znaki (CR i LF)

czasu rozpoczyna się transmisja pola zawierającego adres. Wszystkie dozwolone znaki transmitowane są w kodzie szesnastkowym. Urządzenia sieciowe nasłuchują przesyłane komunikaty (łącznie z odstępami czasowymi). Kiedy odbiorą one pole adresu, każde z nich rozkodowuje informacje w nim zawarte. Jeżeli adres przesłany w komunikacie zgadza się z adresem urządzenia, to rozpoczyna się dekodowanie pozostałych pól. Ostatnie pole (cisza na łączu) jest zarazem pierwszym polem następnego komunikatu. Cała wiadomość musi być przesłana ciągłym strumieniem. Jeżeli przed odebraniem kompletnej ramki komunikatu cisza na łączu będzie dłuższa od 1,5 · (czas trwania pojedynczego znaku), urządzenie odbierające czyści bufor danych i spodziewa się początku następnego komunikatu. Jeżeli cisza na łączu będzie krótsza niż 3,5 · (czas trwania pojedynczego znaku), to urządzenie odbiorcze zinterpretuje to jako kontynuację poprzedniej ramki, co będzie powodowało błąd CRC.

Standard Modbus w sieciach opartych na transmisji szeregowej zapewnia sprawdzanie parzystości (*parity checking*), opcjonalnie stosowane do każdego znaku, oraz sprawdzanie ramki (LRC, CRC), stosowane do każdej ramki wiadomości. Obie metody są inicjowane przez urządzenie nadrzędne i zastosowane przed rozpoczęciem nadawania transmisji. Urządzenie podrzędne przed przyjęciem całego komunikatu sprawdza, czy nie występuje błąd.

Sprawdzanie parzystości (kiedy jest włączone) ma dwie konfigurowalne opcje, jakimi są parzystość (*even*) i nieparzystość (*odd parity*). Jeżeli parzystość jest włączona (tabela 2), wtedy bit parzystości będzie obliczany dla każdego znaku komunikatu i ustawiony tak, aby wszystkie bity były ustawione zgodnie z wcześniej wybraną opcją. Przykład, rozważmy 8-bitowy znak zawarty w ramce znaku RTU 1100 0101. Ramka zawiera cztery jedynki. Jeżeli jest włączona „parzystość", wtedy bit parzystości będzie ustawiony na 0 (suma jedynek jest parzysta). Jeżeli jest włączona opcja „nieparzystość", wtedy bit parzystości będzie ustawiony na 1 (suma jedynek jest parzysta). Gdy sprawdzanie parzystości jest włączone (opcja

Tabela 2. Parzystość włączona

Start	1	2	3	4	5	6	7	8	PAR	Stop

parity), to bit parzystości jest dołączany do każdego znaku transmitowanego komunikatu. Urządzenie odbiorcze oblicza liczbę jedynek i wykrywa błąd w przypadku, gdy ich suma nie zgadza się z bitem parzystości ustawionym przez urządzenie nadawcze. Jeżeli sprawdzanie parzystości jest wyłączone (*no parity*), wtedy bit parzystości nie jest obliczany, a zamiast niego w ramce znaku znajduje się dodatkowy bit stopu (tabela 3).

Pole LRC (*longitudinal redundancy check*) ramki komunikatu składa się z jednego 8-bitowego bajtu. Wartość LRC jest obliczana

Tabela 3. Parzystość wyłączona

Start	1	2	3	4	5	6	7	8	Stop	Stop

przez urządzenie nadawcze. Urządzenie odbiorcze przelicza ponownie LRC i porównuje wartość otrzymaną w ramce komunikatu z wartością wyliczoną przez siebie. Jeżeli obie wartości różnią się, to oznacza, że wystąpił błąd. Obliczanie LRC polega na sumowaniu kolejnych 8-bitowych bajtów komunikatu, odrzuceniu przeniesień i na koniec wyznaczeniu uzupełnienia dwójkowego. Sumowanie obejmuje całą wiadomość z wyjątkiem znaczników początku i końca ramki.

Pole CRC (*cyclic redundancy check*) ramki komunikatu składa się z 16-bitowego rejestru podzielonego na dwie części. Wartość CRC jest obliczana przez transmitujące urządzenie i dodawana do komunikatu. Urządzenie odbiorcze oblicza ponownie wartość CRC i porównuje ją z wartością odebraną (obliczoną przez urządzenie nadawcze). Jeżeli obliczone wartości nie są równe, wtedy powstaje błąd.

W momencie rozpoczęcia obliczania sumy CRC rejestr wypełniany jest jedynką FFFF(hex). Następnie pobierany jest 8-bitowy znak danych komunikatu (bity startu, stopu i parzystości nie biorą udziału w generowaniu CRC), na którym, wraz z dotychczasową wartością rejestru, przeprowadzana jest operacja logiczna XOR. Wynik operacji XOR jest umieszczany w rejestrze CRC, następnie wartość rejestru jest przesuwana o jedną pozycję w stronę najmniej znaczącego bitu (LSB). Miejsce po dotychczasowym najbardziej znaczącym bicie jest wypełniane zerem. Następnym krokiem jest sprawdzenie wartości LSB. Jeżeli wynosi ona 1, wtedy jest przeprowadzana kolejna operacja XOR wartości przesuniętego rejestru z wcześniej zdefiniowaną stałą, w przeciwnym razie następuje kolejne przesunięcie. Powyższy proces jest powtarzany, dopóki nie zostanie przesunięty ostatni bit (8-bitowego) znaku danych. Kiedy obliczenie CRC dla pojedynczego znaku danych zostanie zakończone, następuje pobranie następnego 8-bitowego znaku (dwa znaki w kodzie szesnastkowym), na którym wraz z wcześniej przetworzoną wartością rejestru jest wykonywana operacja XOR, następnie jest wykonywane przesunięcie wartości rejestru itd. Ostateczną wartością CRC jest wynik obliczeń przeprowadzony dla wszystkich pól transmitowanej ramki komunikatu.

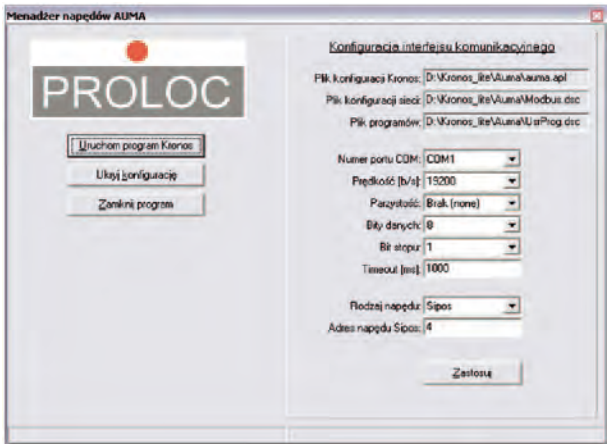
Kody funkcji w protokole Modbus

W zależności od rodzaju implementacji protokołu Modbus w urządzeniu dostępne są różne funkcje przez niego realizowane. W tabeli 4 przedstawiono najpopularniejsze funkcje implementowane w urządzeniach (kody podane dziesiętnie).

Tabela 4. Przykładowe kody funkcji w protokole Modbus

Kod funkcji	Nazwa
1	odczyt wyjścia dwustanowego
2	odczyt wejścia dwustanowego
3	odczyt rejestru wyjściowego
4	odczyt rejestru wejściowego
5	ustawienie pojedynczego wyjścia dwustanowego
6	zapis pojedynczego rejestru
7	odczyt statusu urządzenia *slave*
8	test diagnostyczny

W napędach grupy Auma został udostępniony tryb RTU jako najczęściej stosowana i używana konfiguracja protokołu Modbus w automatyce przemysłowej. Tryb ten współpracuje z interfejsem RS485, dla którego został opracowany program komunikacyjny „Auma Modbus Tester", dalej zwany w skrócie AMT (fot. 1).

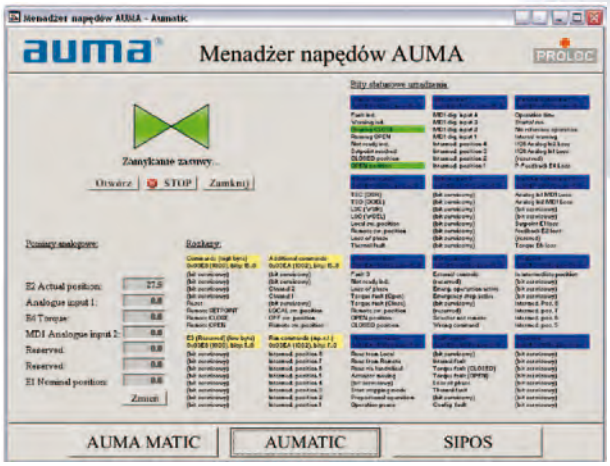


Fot. 1. Obraz ekranu głównego programu AMT (Auma Modbus Tester)

Typowy *master* to urządzenie z procesorem głównym (*host processor*), zawierające programowalny panel, np. komputer PC lub nadrzędny sterownik logiczny, a typowy *slave* to programowalny sterownik logiczny (sterownik Auma). Istnieje również odmiana protokołu Modbus (Modbus Multi-Master), w której jest więcej niż jeden *master* i w którym jednostki nadrzędne przekazują sobie wzajemnie prawo do nadawania.

Najważniejsze cechy protokołu Modbus to: zasada dostępu do łącza *query-response* (*master-slave*), która gwarantuje bezkonfliktowe współdzielenie magistrali przez wiele węzłów, węzeł nadrzędny (*master*), którym jest np. komputer klasy PC, sterujący pracą sieci, oraz węzły podrzędne (*slaves*), które nie podejmują samodzielnie transmisji, a jedynie odpowiadają na zdalne polecenia od węzła nadrzędnego. Każdy z węzłów podrzędnych ma przypisany unikatowy adres; są 2 różne tryby transmisji: ASCII (znakowy) lub RTU (binarny), a komunikaty zawierające polecenia i odpowiedzi mają identyczną strukturę; maksymalna długość komunikatów wynosi 256 bajtów i wreszcie znaki są przesyłane szeregowo od najmłodszego do najstarszego bitu.

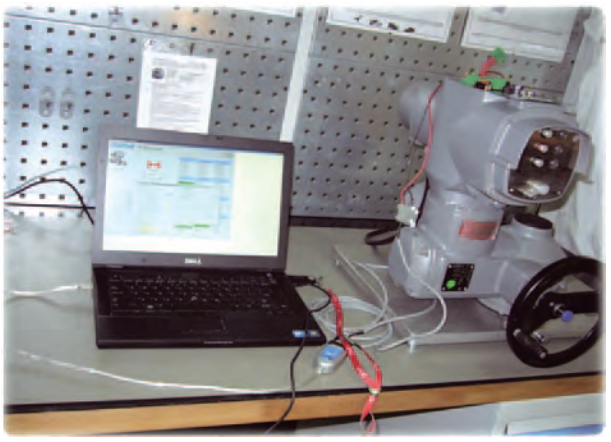
Oprogramowanie aplikacyjne systemu Kronos służące do prac serwisowych związanych z diagnostyką oraz sterowaniem napędami Auma z wykorzystaniem protokołu komunikacyjnego Modbus RTU jest przystosowane do obsługi napędów Auma Matic, Aumatic i Sipos (fot. 2).



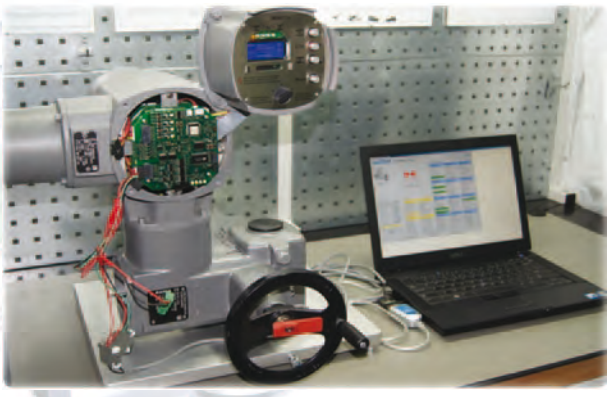
Fot. 2. Zrzut z ekranu diagnostycznego komunikacji Modbus RTU

Aplikacja pozwala na sprawdzenie bieżącego statusu urządzenia, wydawanie rozkazów za pomocą rozkazów dyskretnych oraz definiowanie wartości zadanej stopnia otwarcia przepustnicy/zaworu. Aby zacząć pracę z programem należy podłączyć napęd w sposób pokazany na fot. 3 i 4. Potwierdzeniem poprawnej współpracy podłączonych elementów jest pulsowanie na czerwono diod 8 (Data Ex) i 32 (Bus Act). Umożliwia to odczytanie parametrów i kontrolę komunikacji (fot. 5).

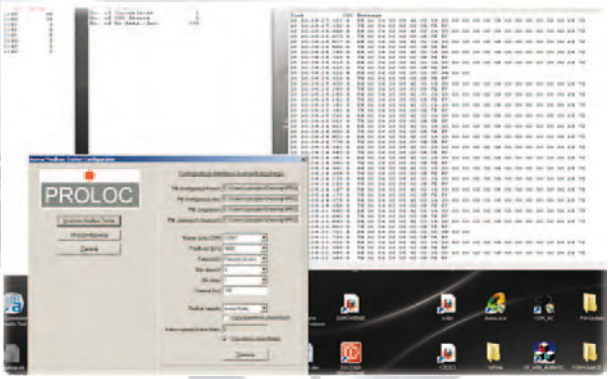
Aby wykonać funkcje testowe, proselektor wyboru musi być w położeniu „Zdalne". Układ testowy AM (fot. 3 i 6) lub AC (fot. 7 i 8) (karta testowa, laptop) ma możliwość przesterowania otwórz-stop-zamknij lub wykonania autotestu pracy napędu. Jest to grupa ikon



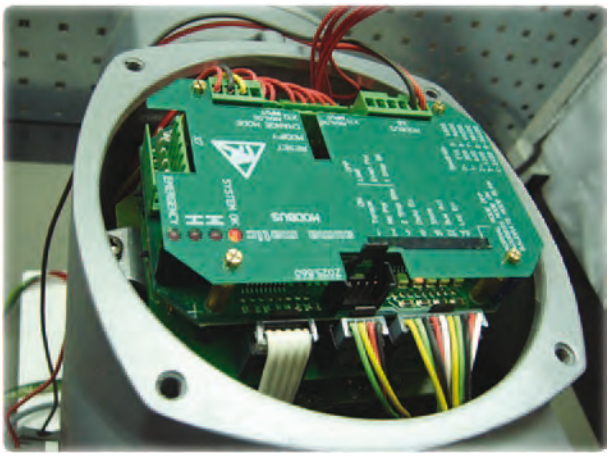
Fot. 3. Podłączony napęd SA 10.2-z AM01.1 na stanowisku badawczym antystatycznym z kartą RS-422/485 firmy MOXA do płyty interfejsu Modbus RTU Auma Matic, z kontrolą poprzez laptop serwisowy



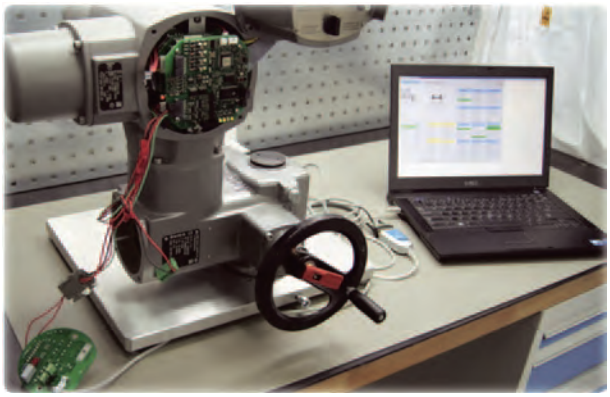
Fot. 4. Podłączony napęd SA 07.1 z AM01.1



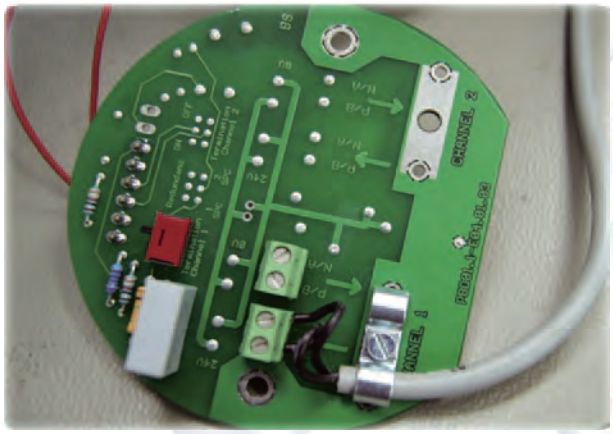
Fot. 5. Zrzut z ekranu głównej karty serwisowej PROLOC, w tle podgląd transmisji danych po omawianym protokole



Fot. 6. Widok płyty Modbus zamontowanej w sterowniku AM



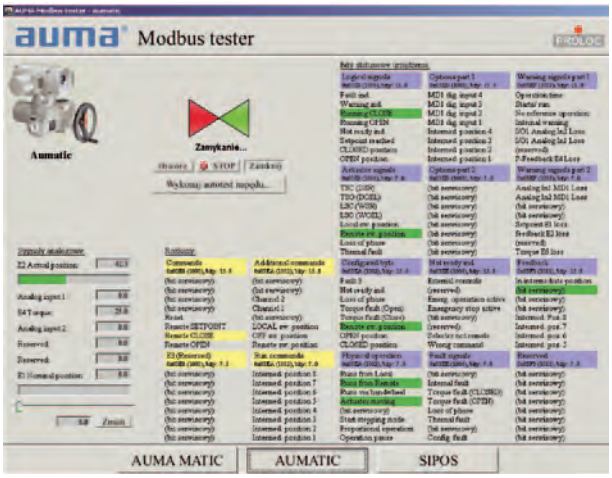
Fot. 7. Podłączony napęd SA 07.1 z AC01.1 z wykorzystaniem płyty interfejsu przyłączeniowego



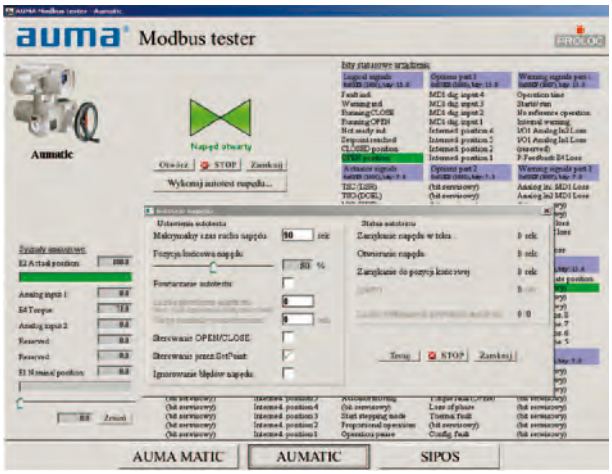
Fot. 8. Płytka przyłączeniowa magistrali (możliwa terminacja)

w lewym górnym roku okna głównego programu. Po naciśnięciu przycisku „Wykonaj autotest napędu" otworzy się okno z funkcjami autotestu (fot. 9). W tym oknie można nastawić (*i*) maksymalny czas ruchu napędu, (*ii*) powtarzanie autotestu, (*iii*) liczbę powtórzeń, (*iv*) odstęp pomiędzy powtórzeniami, (*v*) sterowanie poprzez „open-close", (*vi*) sterowanie poprzez „SetPoint" oraz (*vii*) ignorowanie błędów (do trzech prób transmisji).

Z programu AMT można wysłać rozkaz do napędu i skontrolować stan poprawności wykonania rozkazu. Wizualizację pracy napędu zrealizowano poprzez zastosowanie kolorów. Kolor zielony oznacza poprawną pracę, pojawia się też komunikat tekstowy o stanie wykonania zadania, kolor zielony naprzemiennie z czerwonym sygnalizuje wykonanie pracy napędu, a czarny oznacza brak wykonania rozkazu. Aby wykonać test pracy napędu do pozycji Zamknij, należy nacisnąć przycisk Zamknij i obserwować pracę napędu. Jeżeli wystąpił błąd, jest on sygnalizowany w bitach statusowych napędu (prawa strona okna; fot. 10).



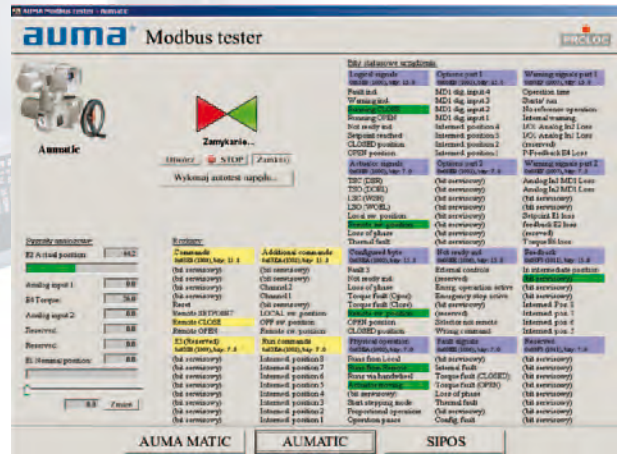
Fot. 9. Zrzut z ekranu karty serwisowej Auma Modbus Tester, do dyspozycji jest testowanie wg opcji Auma Matic, Aumatic i Sipos



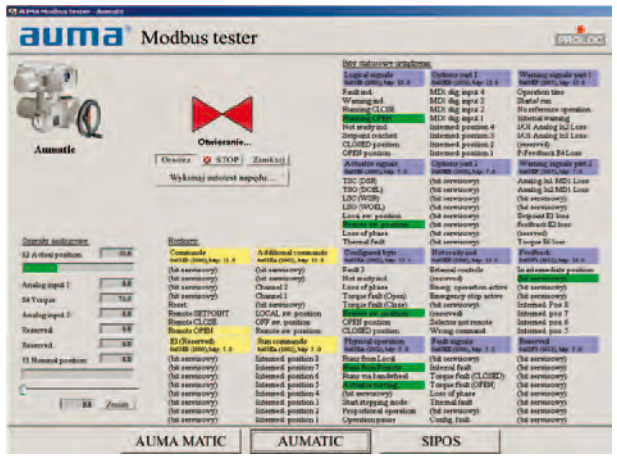
Fot. 10. Zrzut ekranu otwartego okna Autotestu

W zaistniałym przypadku zasymulowano wystąpienie przeciążenia w kierunku zamykania testerem mikrołącznika TSC, w wyniku czego na ekranie pojawiła się informacja o wystąpieniu konkretnego błędu zaznaczana zielonym tłem w komórkach o numerach 12289 (bit 7) TSC (DSR) i 12291 (bit 5) TSC (DSR). W napędzie, który nie posiada potencjometru lub RWG będzie w bitach informacja w postaci ostrzeżenia o niewystępowaniu opcji (komórka 12288, bit 7). Brak potencjometru powoduje w komórce 12292 (bit 3) podkreślenie Potencjometer V. Prawa strona okna przestawia informacje statusu autotestu napędu (czas zamykania napędu, czas otwierania napędu i czas zamykania do pozycji końcowej) oraz informację o liczbie wykonanych powtórzeń autotestu (fot. 11 a, b, c). W dolnym prawym roku okna umieszczone są trzy przyciski (Testuj, Stop, Zamknij). W oknie głównym w lewej dolnej części podawane są wartości sygnałów analogowych (o ile

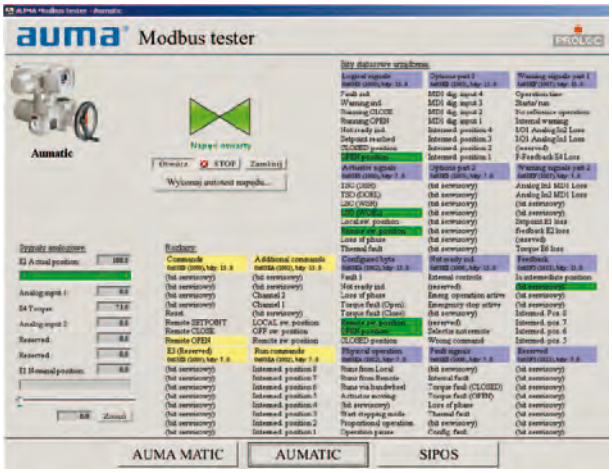
a)



b)



c)



Fot. 11 a, b, c. Zrzuty ekranu różnych stanów położenia napędu; wykonywanie rozkazu zamykania i otwierania, stan po wykonaniu zadania

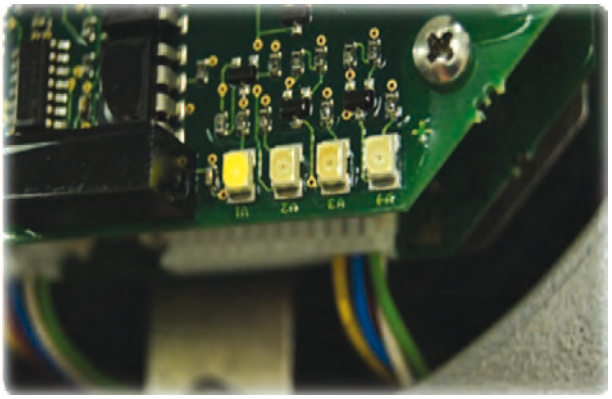
urządzenie ma odpowiednie wyposażenie); E2 – aktualna pozycja: analogowe wejście 1 i analogowe wejście 2 oraz wartość zadana E1 SetPoint. W dolnej części jest suwak lub cyfrowa możliwość wpisania wartości zadanej E1 SetPoint. Fotografia 12 przedstawia widok tych wartości na wyświetlaczu sterownika.

Fot. 12. Widok wartości E1 i E2 na wyświetlaczu sterownika (opcja sterowania SetPoint)

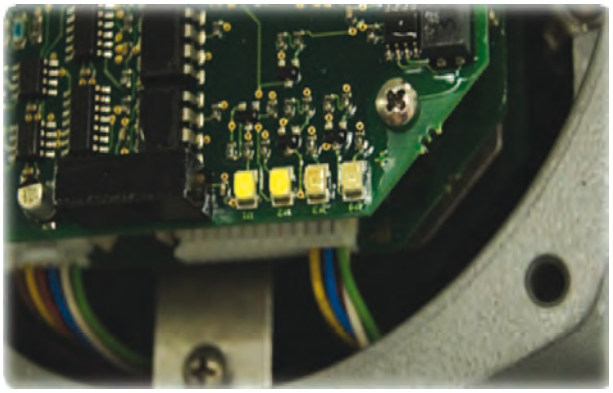
Oprócz sygnałów analogowych są jeszcze informacje o sygnałach wychodzących z programu do napędu. Rozkazy High 15 do 8 (nr adresu 4096), niewykorzystane w programie, i 0 do 7 (nr adresu 4096) Low, wykorzystane w programie w zakresie, 0 – Remote, Open, 1 – Remote, Close, 2 – Remote, SetPoint i 3 do 7 niewykorzystane. W prawej części okna programu znajdują się bity statusowe urządzenia (informacje wychodzące z napędu). Bity są pogrupowane w oznaczeniu wysokim 15–8 i niskim 7–0 pod adresem komórek 12288–12294 oraz 15–8 stany nieprzypisane (bity serwisowe).

Program ma możliwość testu sterowników Auma Matic, Aumatic i Sipos przy czym postępowanie jest analogiczne jak dla opisanego testu ze sterownikiem Auma Matic (płyta testowa fot. 13a,b,c) oraz Aumatic (płyta testowa fot. 8).

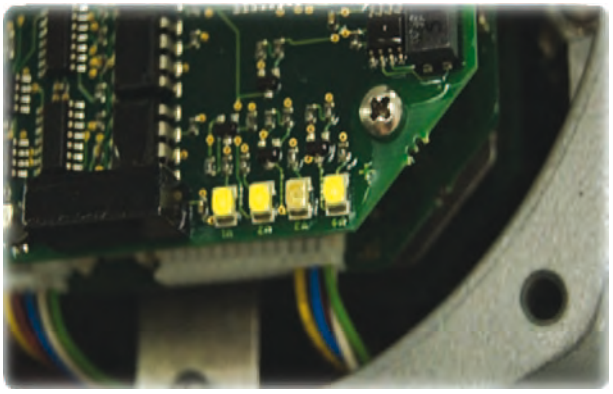
a)



b)



c)



Fot. 13 a, b, c. Płyta interfejsu Modbus RTU – Aumatic; diody V1 (LED system OK), V2 (LED DATA EX), V3 (LED CAN STATE) i V4 (LED STATE). Poszczególne stany: od włączenia do nawiązania prawidłowej komunikacji

Szymon Czyczyło, Robert Łudzień, Krzysztof Sobieraj
Auma Polska Sp. z o.o.