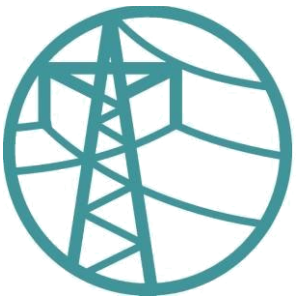


WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Hubert Dzieciuch

Laboratoryjny model bramy

Praca dyplomowa inżynierska

Opiekun pracy:
dr inż. Dariusz Rzońca

Rzeszów, 2022

Pragnę złożyć serdeczne podziękowania promotorowi
dr inż. Dariuszowi Rzońcy za
cierpliwość, wyrozumiałość oraz pomoc na
każdym etapie tworzenia tej pracy.

Spis treści

1. Wprowadzenie

Sterowniki programowalne pojawiły się pod koniec XX wieku zastępując dotychczasowe układy sterowania oparte na tradycyjnej technice przekaźnikowo-stycznikowej. Od tamtego czasu są one ciągle doskonalone i znajdują swoje zastosowanie w wielu dziedzinach: automatyce przemysłowej, automatyce domowej oraz w urządzeniach codziennego użytku. Główną przyczyną popularności sterowników PLC (*Programmable Logic Controllers*) jest możliwość reagowania na ciągle unowocześniające się wymagania aplikacji bez konieczności dokonywania całościowych zmian sprzętowych. Dodatkowym atutem sterowników, który powoduje że wiele osób skłania się do użycia ich w swoich projektach jest prostota konfiguracji sterownika oraz wdrożenia go do projektu, przy zachowaniu pełnej kompatybilności. Zawdzięczamy to dedykowanym narzędziom, które pozwalają na programowanie sterowników bez zagłębiania się w najniższą warstwę sprzętową procesora. Rosnąca popularność tego typu sterowników wymagała ujednolicenia metod zastosowanych w dedykowanych środowiskach programistycznych. Wynikiem standaryzacji jest norma IEC61131-3 [3] [4], która definiuje wiele właściwości jakie powinny być przestrzegane przez środowisko programistyczne. Jedną z kluczowych jest zalecenie użycia następujących języków programistycznych: LD (*Ladder Diagram*), FBD (*Function Block Diagram*), ST (*Structured Text*), IL (*Instruction List*), SFC (*Sequential Function Chart*) [3][4][18].

W dzisiejszych czasach ludzkość dąży do zautomatyzowania niemal każdego obszaru naszego życia. Nie inaczej sprawa wygląda w dziedzinie systemów bram automatycznych. Zautomatyzowane systemu parkingów nie są już żadną nowością, również coraz rzadziej spotykamy tradycyjne rozwiązanie w naszych domach. Upodobanie wygody coraz częściej prowadzi nas do stosowania systemów, które nie wymagają od nas większych działań. Spotkać możemy wszelakie rozwiązania dotyczące tego problemu gdzie swoje zastosowanie znajdują piloty lub dedykowane aplikacje internetowe.

Celem niniejszej pracy jest zbadanie poprawności programowania sterownika SU1.7 [7] za pomocą środowiska programistycznego CPDev. W tym celu na jego podstawie wykonane zostało stanowisko laboratoryjne symulujące działanie bramy automatycznej. Ma to na celu dokładną weryfikację pracę takowego systemu w oparciu o ten sterownik oraz ukazanie możliwości jego zastosowania w zamienniku dla gotowych lecz często wiele droższych gotowych rozwiązań. Sterownik ten został stworzony przez firmę E-TRONIX. W ramach współpracy z Politechniką Rzeszowską, firma ta udostępniła

jeden z egzemplarzy w celach naukowych. Sterownik ten programy jest za pomocą specjalnej wersji narzędzia CPDev (Control Program Developer), opracowanej w Katedrze Informatyki i Automatyki.

W drugim rozdziale skupiłem się na przedstawieniu środowiska inżynierskiego CPDev. Ukazane zostały dostępne języki programowania, budowa środowiska oraz jego główne funkcjonalności w programowaniu sterowników PLC.

Trzeci rozdział pracy zawiera informacje dotyczące sterownika E-TRONIX SU1.7. Ukazana została jego ogólna charakterystyka wraz z jego możliwościami, budowa zewnętrzna wraz z krótkim opisem poszczególnych wyjść oraz ich funkcjonalności. Opisane zostały czynności jakie trzeba podjąć, aby ten sterownik uruchomić na podstawie programu testowego dostarczonego ze sterownikiem, który ma za zadanie ukazać użytkownikowi dostępne możliwości.

W czwartym rozdziale został przedstawiony system bramy automatycznej. Ukazano wizję na podstawie której został ten model wykonany. Zawarte zostały zrzuty ekranu projektu modelu bramy, zdjęcia oraz opis stanowiska symulującego działanie bramy przesuwnej sterowanej za pomocą pilota. Wytłumaczone zostało działanie wszelkich komponentów elektronicznych oraz schematów połączeń.

Piąty rozdział to ukazanie rozwiązania problemu sterowania bramą za pomocą automatu stanowego. Zawarte zostały kody programów wszystkich zdefiniowanych bloków funkcyjnych oraz głównego programy wykonawczego wraz ze wszystkimi opisami pozwalającymi zrozumienie koncepcji działania.

Szósty rozdział zawiera całą dokumentację z testowania przygotowanego stanowiska oraz systemu sterowania. Mając gotowe wszelkie elementy składowe potrzebne do wykonania finalnej wersji bramy automatycznej zostały przetestowane: działanie systemu sterowania oraz poszczególnych elementów pozwalających na bezpieczne dla człowieka użytkowanie w przypadku użycia sterownika oraz programu w modelu rzeczywistym.

Ostatni rozdział to podsumowanie wykonanej pracy. Zawiera on wnioski oraz refleksje dotyczące zbudowanego systemu bramy automatycznej, działania sterownika oraz środowiska programistycznego CPDev.

2. Środowisko programistyczne CPDev

W rozdziale tym krótko przedstawione zostało środowisko inżynierskie CPDev, jego kompatybilność z normą IEC 61131-3, krótko opisane zostało budowanie projektu oraz wykorzystanie tego oprogramowania.

2.1. Przedstawienie środowiska

CPDev (Control Program Developer) [6] jest to inżynierskie środowisko programistyczne przeznaczone do programowania sterowników PLC oraz PAC. Służy ono również jako narzędzie dydaktyczne na Politechnice Rzeszowskiej. Pierwsza wersja została wykonana w latach 2006/2007, do programowania służył wtedy wyłącznie język ST. Przez lata narzędzie to było rozwijane pod okiem zespołu programistów z Katedry Informatyki i Automatyki.

Swoje zastosowanie środowisko to znalazło w kilku większych projektach jak np. zastosowanie w rozproszonym systemie kontrolno-pomiarowym w Zakładach Lumel SA w Zielonej Górze. Przykład zastosowania CPDev'a wraz ze sterownikami marki SMC LUMEL został opisany w [1]. CPDev został użyty również w Systemie Mega-Guard służącego do nawigacji oraz automatyzacji statków holenderskiej firmy Praxis Automation Technology B.V. Jak możemy wyczytać w [2] system ten składa się z kilkunastu podsystemów o różnych przeznaczeniach, które zawierają : sterowniki, moduły I/O, panele operatorskie, panele HMI połączone ze sobą za pomocą redundowanego Ethernetu. Przykłady te pokazują możliwości użycia CPDev'a w realnych, dużych projektach.

2.2. Zgodność z normą PN/EN 61131-3 (IEC)

Norma IEC 61131-3 [3] powstała w 1998 roku, jej głównym celem jest standaryzacja zasad programowania sterowników przemysłowych PLC. Pełni ona rolę zbioru wytycznych, których musi przestrzegać wykonawca sterownika lub oprogramowania. Jej nazwa została uznana w Europie, sama norma odwołuje się natomiast do 10 innych dokumentów. Składa się ona z pięciu części. Część pierwsza definiuje ogólne postanowienia, przykładowo: cykliczne przetwarzanie programu lub przydział czasu pracy na komunikację. Część druga definiuje natomiast jakie wymagania oraz badania powinien spełnić sprzęt oraz dokonano ich kwalifikacji. Kolejna część wspomina o językach programowania służących do programowania sterowników. Są to informacje na temat: struktury programu, podstawowych bloków funkcjonalnych oraz elementów konfiguracji wspomagających ich instalację. Część czwarta jest to przewodnik dla użytkownika

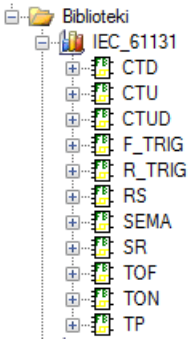
w sprawie budowy systemu automatyki, doboru sprzętu i jego zastosowań. Ostatnia część dotyczy zasad komunikacji w procesie produkcji, określa funkcje adresowania oraz wymiany danych.

Zgodność z normą PN/EN 61131-3 (IEC) jest największą zaletą pakietu inżynierskiego CPDev, pozwala to na zapewnienie kompatybilności z wieloma rodzajami sterowników oraz systemów sterowania ją wykorzystujących ze względu na powszechność jej stosowania [14][17]. Stosują się do niej wszystkie firmy świata, które chcą móc nazywać swój sprzęt sterownikiem PLC. Dzięki temu w środowisku mamy możliwość programowania za pomocą pięciu języków:

1) ST (Structured Text - Tekst strukturalny) – jest to odpowiednik języków algorytmicznych wysokiego poziomu, którego zbiór instrukcji jest porównywalny do tych z rodziny C oraz języka PASCAL.
2) IL (Instruction List - Lista rozkazów) – to drugi język tekstowy, będący odwzorowaniem języku niskopoziomowego typu asembler. Zbiór jego instrukcji zawiera operacje logiczne, arytmetyczne, relacji oraz bloków funkcyjnych np. czasomierzy, liczników lub przerzutników.
3) FBD (Function Block Diagram - Funkcjonalny schemat blokowy) – to odpowiednik schematu przepływu sygnałów dla obwodów logicznych, które ukazywane są jako połączenie bramek logicznych, bloków funkcjonalnych oraz funkcji. Często wybierany jest przez osoby nie mające doświadczenia w programowaniu.
4) LD (Ladder Diagram - Schemat drabinkowy) - często stosowany, ze względu na podobieństwo jest w systemach opartych na metodzie przekaźnikowej. Sięgają po niego osoby, które mają doświadczenie w wykonywaniu takich właśnie systemów. Polega zastosowaniu styków, cewek oraz połączeń między nimi.
5) SFC (Sequential Function Chart - Sekwencyjny schemat funkcjonalny) – jest to w zasadzie sieć językowa, która pozwala na opisywaniu zadań za pomocą grafu skierowanego. Grafy te zawierają etapy(kroki), które mogą zostać zdefiniowane za pomocą reszty języków oraz tranzycie (warunki przejścia).

Kolejnym faktem ukazującym że zgodność z normą IEC 61131-3 jest biblioteka dostarczona wraz z oprogramowaniem. Zawiera ona bloki funkcjonalne (rys. 2.2.1), które są charakterystyczne dla języków programowania wykorzystywanych w sterownikach

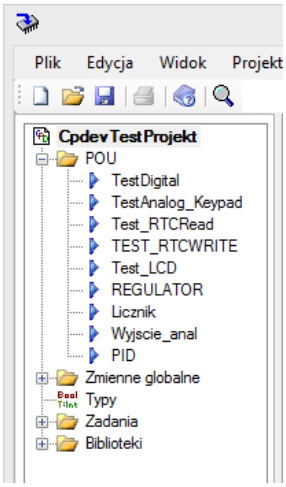
PLC. Są to kolejno: liczniki (CTD, CTU CTUD), bloki odpowiadające za wykrywanie zbocza (F_TRIG, R_TRIG), przerzutniki (RS, SR), czasomierze (TON, TOF), semafor (SEMA) oraz generator impulsu (TP) [15].



Rys. 2.2.1 Biblioteki CPDev zgodne z normą IEC 61131-3

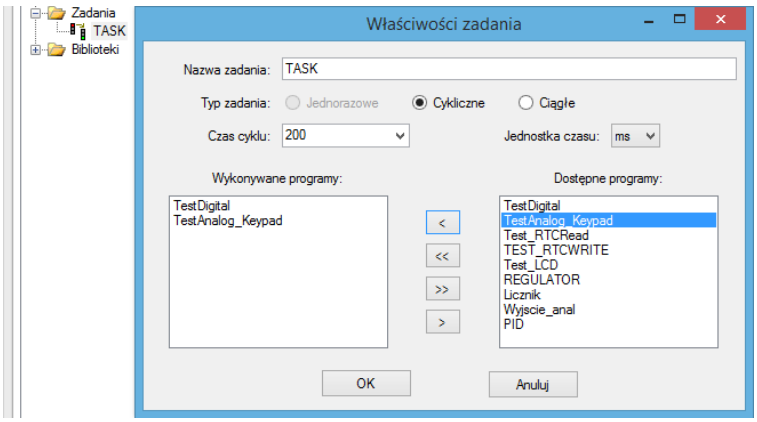
2.3. Budowa projektu w środowisku CPDev

Program wykonywany w środowisku inżynierskim CPDev, widoczny jest w postaci drzewa projektu (rys. 2.3.1). Składa się ono z POU, czyli programowych jednostek organizacyjnych. Są to wszystkie programy oraz bloki funkcjonalne napisane przy użyciu jednego z wcześniej wspomnianych języków. Takie podejście pozwala na łatwe połączenie projektu w całość, w sytuacji gdy jest on tworzony przez większą liczbę osób.



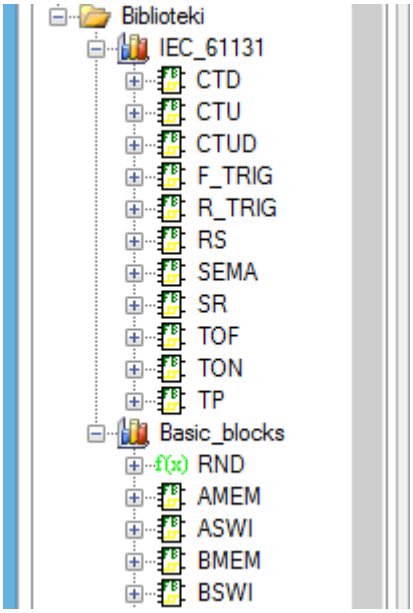
Rys. 2.3.1 CPDev – drzewo projektu

Tak utworzone jednostki POU, muszą zostać przydzielone do konkretnego zadania (ang. Task) (rys. 2.3.2). Te POU, które nie zostaną zdefiniowane jako zadanie zostaną pominięte podczas wgrywania na sterownik. Dodatkowo mamy możliwość wyboru typu dla danego zadania oraz zdefiniowania czasu cyklu z jakim ma się ono wykonywać.



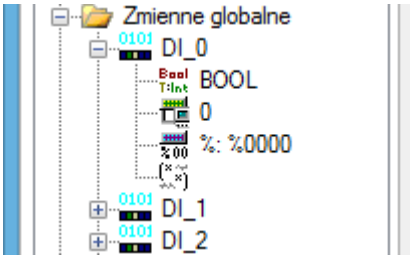
Rys. 2.3.2 CPDev – definiowanie zadań

Program POU możemy utworzyć za pomocą dowolnego języka również przy pomocy zawartych w bibliotekach środowiska bloków funkcjonalnych.



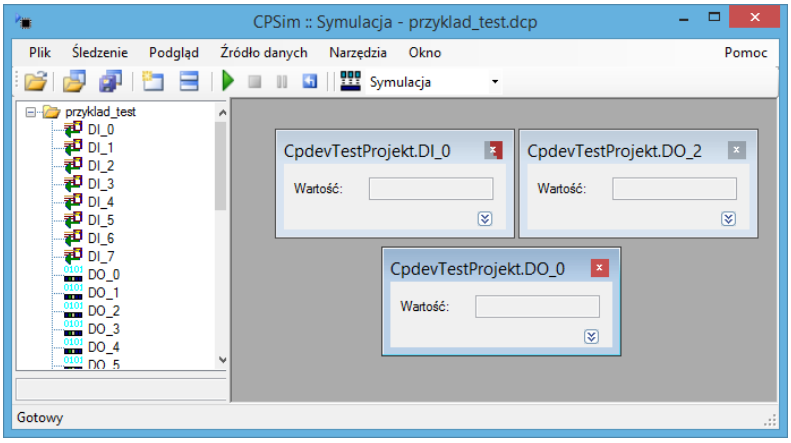
Rys. 2.3.3 CPDev - wbudowane bloki funkcjonalne

Ważnym krokiem jest zdefiniowanie adresy zmiennych globalnych (rys. 2.3.4). Jest to kluczowe działanie w celu nawiązania poprawnej komunikacji ze sterownikiem lub oprogramowaniem do przeprowadzania symulacji np. Wonderware InTouch.



Rys. 2.3.4 CPDev – Definiowanie zmiennych globalnych

Dodatkowym atutem jest wbudowany symulator CPSim (rys. 2.3.5) pozwalający na szybkie oraz wygodne przetestowanie utworzonego przez nas programu bez wgrywania na sterownik, unikając dzięki temu związanego z tym ryzyka uszkodzenia go oraz wyeliminowanie wszystkich niepoprawności.



Rys. 2.3.5 CPDev – Symulator CPSim

3. Programowalny sterownik E-TRONIX SU1.7

Przy współpracy Katedry Informatyki i Automatyki Politechniki Rzeszowskiej oraz firmy E-TRONIX zostało umożliwione programowanie sterownika SU1.7 przy pomocy odpowiednio zmodernizowanego środowiska CPDev. W rozdziale tym przedstawię charakterystykę tego sterownika oraz ukażę sposób programowania go.

3.1. Charakterystyka ogólna oraz budowa sterownika

Sterownik SU1.7 jest prototypowym sterownikiem firmy E-TRONIX [7] (rys. 3.1.1). Jednostką centralną stanowi mikroprocesor ATmega1284, firmy ATMEL.

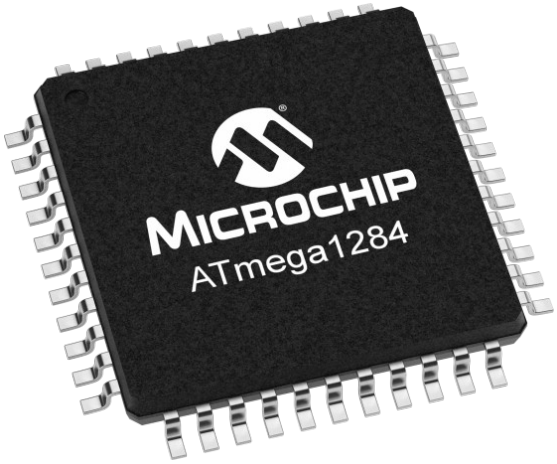
Rys. 3.1.1 Sterownik E-TRONIX SU1.7

Cechy charakterystyczne sterownika SU1.7 [19]:

1) Mikroprocesor ATmega1284
2) Wyświetlacz LCD 2x16
3) 8 wejść cyfrowych,
4) 8 wyjść logicznych,
5) 2 wejścia analogowe (napięciowe 0..10V),
6) 2 wejścia analogowe (temperaturowe - czujnik PT100),
7) 2 wyjścia analogowe 0..10V,
8) port komunikacyjny RS232/485,
9) zegar czasu rzeczywistego (RTC),
10) akustyczny sygnalizator (buzzer),
11) sygnalizacja stanów wejść cyfrowych i wyjść logicznych,

Sterownik ten składa się z następujących elementów:

Mikroprocesor ATmega1284 jest to 8-bitowy mikrokontroler CMOS małej mocy na bazie architektury RISC z ulepszeniem AVR [9]. Umieszczony jest on w 44-pinowej obudowie TQFP (rys. 3.1.2).. Jego prędkość taktowania wynosi 20MHz oraz posiada 128KB pamięci programów. Umożliwia korzystanie z następujących protokołów komunikacyjnych: I2C, SPI, UART/USART.



Rys. 3.1.2 ATmega1284

Wyświetlacz LCD 2x16 wbudowany w sterownik umożliwia użytkownikowi odróżnienie w jakim stanie pracy znajduje się sterownik, wyświetla on odpowiedni komunikat podczas trybu programowania oraz komunikaty ustalone przez użytkownika w trybie pracy (rys. 3.1.3). Wyświetlacz jest wielkim udogodnieniem, gdyż możemy na nim wyświetlić pożądane komunikaty, informacje o tym w jakim etapie programu znajduje się sterownik podczas jego testowania oraz wiele innych informacji.

Rys. 3.1.3 Wyświetlacz LCD

Komunikacja z komputerem oraz wgrywanie programu przy pomocy CPDev'a odbywa się za pomocą portu komunikacyjnego RS232, do wyboru możliwy jest również port RS485, którego użycia możemy dokonać przy pomocy zmiany odpowiedniej zworki. Dzięki tak zintegrowanemu interfejsowi szeregowemu, RS 232/485 sterownik ten może przy pomocy odpowiednich konwerterów zostać podpięty do już działającego systemu sterowania w sieci przemysłowej np. Profibus, DeviceNet, Ethernet, itp.

Wejścia cyfrowe w liczbie 8 pracujących w logice dodatniej dodatkowo odseparowanych galwanicznie od procesora. Są one podzielone na dwie grupy po 4 wejścia, a każda z nich posiada osobną masę. Stan wejść jest dodatkowo sygnalizowany za pomocą diod LED IN (rys. 3.1.4), które działają z pominięciem procesora.

Rys. 3.1.4 Diody sygnalizujące stan wejść – LED IN

Wyjścia logiczne są podzielone na dwie grupy:

1) wyjścia przekaźnikowe w liczbie 4 ze stykami normalnie otwartymi dla prądu stałego lub zmiennego 2 A, 250 VAC. Styki przekaźników nie są zabezpieczone a ich stan jest sygnalizowany na panelu za pomocą diod LED OUT (nr od 1 do 4) (rys. 3.1.5)..
2) wyjścia tranzystorowe również w liczbie 4 posiadających ochronę przepięciową, przeciążeniową oraz separacje galwaniczną. Stan tych wyjść jest sygnalizowany na panelu za pomocą diod LED OUT (nr od 5 do 8) (rys. 3.1.5).

Rys. 3.1.5 Diody sygnalizujące stan wyjść LED OUT

Wejścia analogowe(napięciowe) sterownika w liczbie dwóch to wejścia do pomiaru napięć 0..10 V lub 0..5 V, zakres pomiarowy możemy wybrać za pomocą zworki. Masa wejścia napięciowego jest połączona galwanicznie z masą mikrokontrolera.

Wejścia analogowe(temperaturowe) służące do pomiaru temperatury za pomocą czujników rezystancyjnych PT100 (o podłączeniu 2-przewodowym).

Wyjścia analogowe działające w zakresie napięć 0..+10 V, sterowane wyjściem PWM procesora. Przeznaczeniem ich jest umożliwienie sterowania silnikami za pomocą falowników.

Zasilanie sterownika może odbywać się na trzy różne sposoby:

1) napięcie zmienne 24 VAC
2) napięcie stałe niestabilizowane 28..38 VDC
3) napięcie stałe stabilizowane 24 VDC

Klawiatura programowalna w którą wyposażony jest sterownik zawiera 6 przycisków, są to kolejno: „UP”, „LEFT”, „RIGHT”, „DOWN”, „ESC”, „OK” (rys. 3.1.1). Przyciski połączone są z wejściami sterownika i można traktować je jako normalne wejścia cyfrowe, działające w logice dodatniej.

3.2. Uruchomienie oraz programowanie sterownika

Sterownik uruchamia się automatycznie po załączeniu zasilania i natychmiast przechodzi do trybu pracy programu. Jest to dodatkowo zasygnalizowane zaświeceniem się diody POWER (rys. 3.2.1).

Rys. 3.2.1 Dioda sygnalizująca uruchomienie sterownika

Programowanie jest możliwe za pomocą: assemblera (AVR STUDIO), języka C (WINAVR), BASIC-a (BASCOM AVR), ARDUINO (ARDUINO), języków normy IEC 61131-3 (CPDev) [5]. Wgrywanie programów bezpośrednio na procesor może odbyć się poprzez złącze ISP, zgodne z programatorami np. STK200/300 (złącze IDC 10). Możliwe jest również wgrywanie za pomocą programatora JTAG zgodnego z AVR JTAG ICE firmy Atmel. Domyślnie natomiast sterownik ten posiada wgrany program bootloader’a, dzięki któremu możliwe jest wgrywanie programów za pomocą złącza szeregowego RS232.

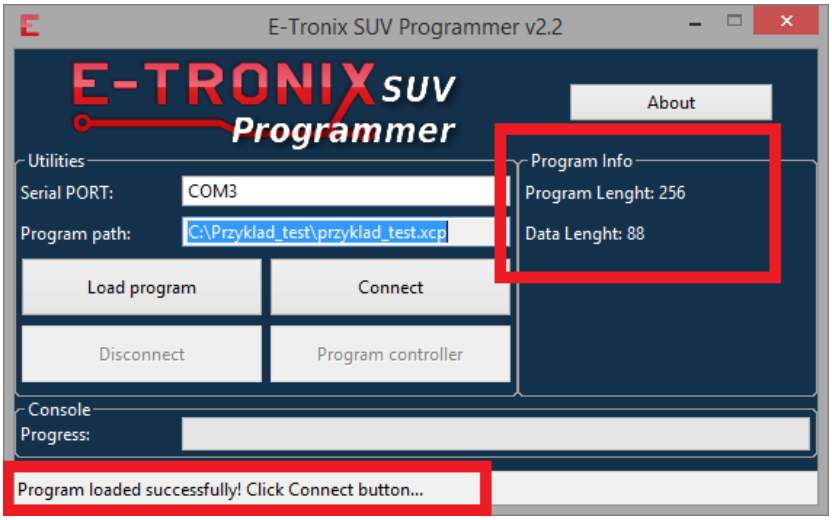
Aby wgrać program należy załączyć zasilanie jednocześnie trzymając wciśnięte przyciski „OK” oraz „ESC”. Sterownik przechodzi wtedy w tryb programowania (rys. 3.2.2).

Rys. 3.2.2 Komunikat wyświetlany na sterowniku w trybie programowania

Program napisany w CPDev’ie należy skompilować, a następnie za pomocą wbudowanego w środowisko Programmera wgrać go na sterownik (rys 3.2.3). Konieczne jest wybranie odpowiedniego portu COM, do którego podpięty jest sterownik w komputerze oraz wybrać ścieżkę dostępu do pliku z rozszerzeniem „.xcp”. Następnie nacisnąć przycisk „Connect”, a po nawiązaniu połączenia kolejno wcisnąć „Program Controller” aby rozpocząć wgrywanie programu. Po wykonaniu wszystkich działań i załadowaniu się programu, sterownik automatycznie przejdzie w tryb pracy.

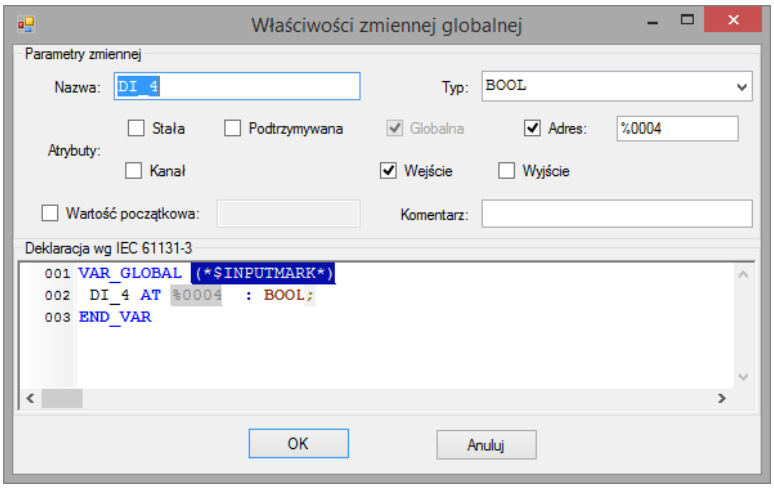
Rys. 3.2.3 Okno Programmera sterownika SU1.7

Dodatkowo po załadowaniu programu do Programmera, w prawej części okna uzyskujemy informacje o jego wielkości, a w dolnej sekcji widnieje informacja o tym czy program został załadowany z sukcesem (rys. 3.2.4).



Rys. 3.2.4 Programmer - wyświetlenie informacji o załadowanym programie

Wraz ze sterownikiem oprócz dokumentacji oraz wszystkich wymaganych do oprogramowania sterownika narzędzi załączony został przykładowy program. Zawiera on kilka jednostek POU, które ukazują podstawowe funkcjonalności sterownika. Najważniejszym elementem są jednak zdefiniowane zmienne globalne, które posiadają odpowiednie adresy określone na stałe w programie boatloader'a (rys. 3.2.5).



Rys. 3.2.5 Tworzenie zmiennych globalnych w programie

Ważne jest więc, aby nasz nowo napisany program posiadał zmienne o tych samych adresach aby procesor mógł odwoływać się do odpowiednich wejść oraz wyjść. Najlepszym sposobem jest jednak wyeksportowanie całej listy zmiennych do pliku o rozszerzeniu „.csv" i zaimportowanie ich do nowo utworzonego przez nas projektu. Takie rozwiązanie pozwoli na wyeliminowanie błędów oraz jest zdecydowanie szybsze niż definiowanie wszystkich zmiennych ręcznie.

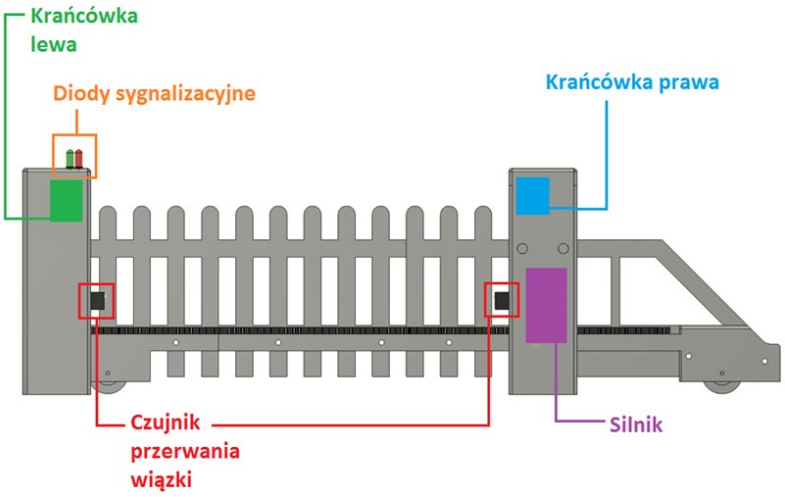
4. Model stanowiska laboratoryjnego

Rozdział ten ukazuje bramę w fazie projektowej. Przedstawiony został tu zaprojektowany model oraz przygotowana płytka PCB. W przypadku płytki, każdy komponent posiada odniesienia do dokumentacji technicznych producenta.

4.1. Koncepcja działania

Tematem przewodnim pracy jest zastosowanie wcześniej opisanego sterownika w systemie sterującym bramą. Aby dokonać analizy działania sterownika w takim systemie wykonany został model bramy, zaprojektowana została część elektroniczna, napisany został program, a ostatecznie wszystko to zostało połączone w całość uzyskując działający model laboratoryjny.

Działanie całego systemu ukażę na podstawie zrzutu ekranu modelu ukazanego na rysunku 4.1.1, wraz z oznaczonymi wszystkimi komponentami. Koncepcja działania jest następująca: użytkownik za pomocą pilota lub klawiatury steruje bramą mając możliwość jej otwierania, zamykania oraz zatrzymania w dwóch trybach: automatycznym i manualnym. Tryb manualny działa w taki sposób, że brama jest otwierana lub zamykana tylko w momencie gdy przycisk jest przytrzymywany przez użytkownika, nie zwracając uwagi na odczyty z czujnika przerwania wiązki. Tryb manualny bierze jednak pod uwagę odczyty z krańcówek. Tryb automatyczny działa na podstawie automatu stanowego opisanego w późniejszej części pracy, ukazanego na rysunku 5.1.1. Krańcówka lewa sygnalizuje zamknięcie bramy, a prawa otwarcie. Całość napędza silnik w zmodyfikowanym serwomechanizmie modelarskim. Dodatkowo dołączone diody LED sygnalizują w jakim momencie pracy jest aktualnie brama.



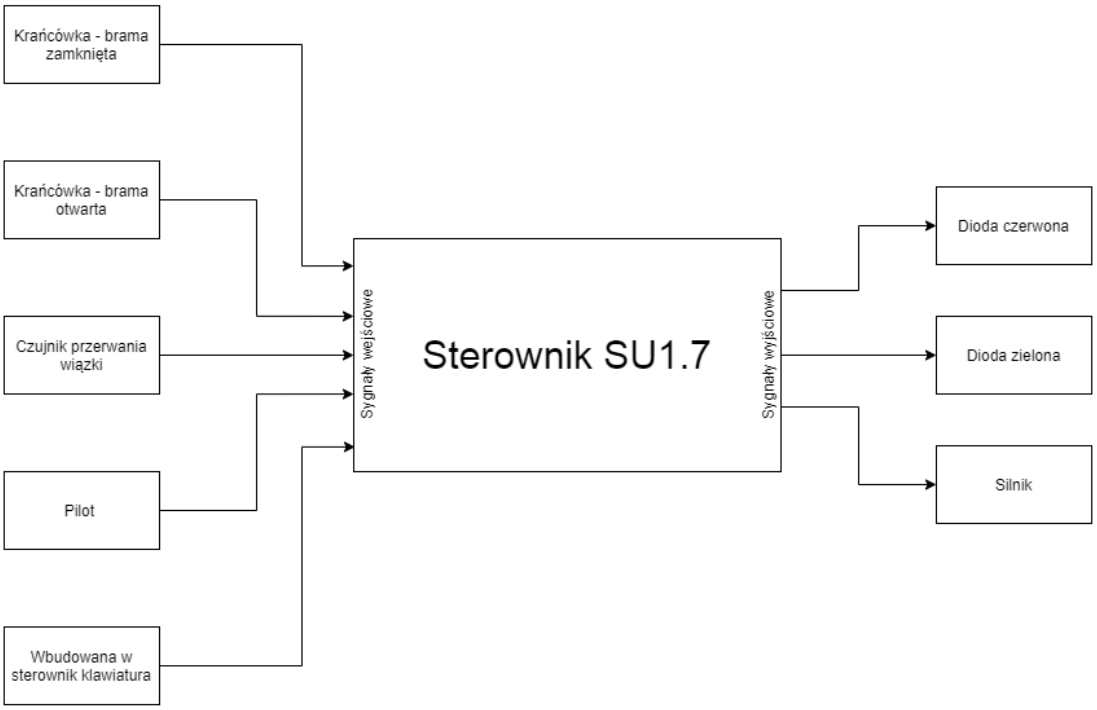
Rys. 4.1.1 Rysunek konceptualny działania modelu

Na rysunku 4.1.2 przedstawiającym blokowy schemat sygnałów wejściowych i wyjściowych widzimy, że jednostką centralną jest sterownik SU1.7. Sygnałami wejściowymi są kolejno:

1) Krańcówki – są to dwa transoptory szczelinowe, wykrywające czy brama jest w danym momencie otwarta lub zamknięta.
2) Czujnik przerwania wiązki – pełni role zabezpieczenia, czy nic nie znajduje się na drodze bramy podczas jej zamykania. Ma to za zadanie uchronić bramę przed kolizją z elementem trzecim, a w bramie realnego rozmiaru pełniłby rolę zabezpieczenia osób lub samochodów przed zniszczeniem.
3) Pilot – system ten będzie sterowany bezprzewodowo za pomocą pilota. Pilot posiada 4 przyciski, a każdy z nich będzie spełniał odpowiednio zadaną rolę, które dokładnie zostaną określone podczas opisywania działania programu sterującego.
4) Klawiatura wbudowana – jest to drugi sposób sterowania bramą, w przypadku gdy nie posiadalibyśmy ze sobą pilota i chcielibyśmy bezpośrednio przy pomocy sterownika bramę otworzyć lub zamknąć.

Sygnały wyjściowe stanowią natomiast:

1) Dioda LED czerwona – sygnalizująca, że brama jest aktualnie zamknięta jeśli świeci się światłem ciągłym. Dioda będzie natomiast mrugać podczas otwierania luz zamykania się bramy.
2) Dioda LED zielona – sygnalizująca, że brama jest otwarta i można przez nią przejechać.
3) Silnik – napęd odpowiadający za poruszanie się modelu.



Rys. 4.1.2 Schemat blokowy sygnałów sterujących bramą

Sterownik SU1.7 pełni rolę jednostki centralnej. Sczytuje on sygnały wejściowe pochodzące z czujników, odpowiednio przetwarza on te informację oraz na podstawie zawartego w nim programu odpowiednio ustala stany wyjść cyfrowych.

4.2. Wykorzystane narzędzia do wykonania modelu

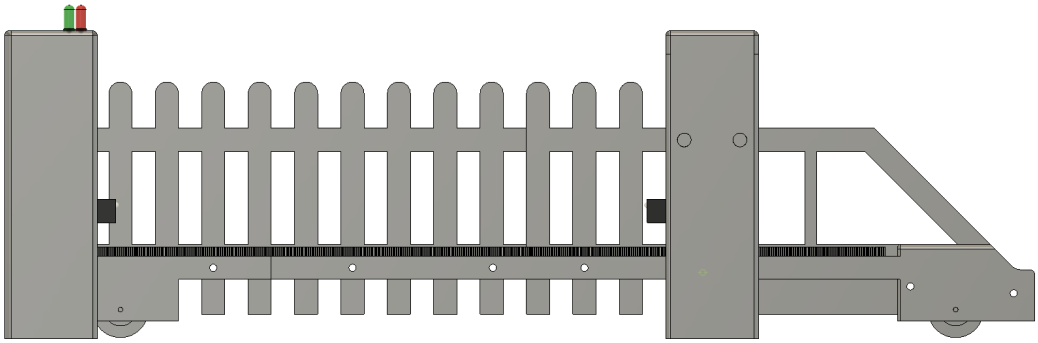
Model projektu został wykonany w oprogramowaniu Autodesk Fusion 360 [16]. Jest to kompletne środowisko do projektowania produktów CAD/CAM/CAE dla branży MFG, zawierającej szereg narzędzi do wykonania kompletnego modelu. Z wielu możliwości oprogramowania, użyto technik modelowania: bryłowego, powierzchniowego, swobodnego brył, komponentów oraz zespołów. Jest to oprogramowanie bardzo popularne pośród studentów ze względu na wszechstronność oraz możliwość integracji z innymi oprogramowaniami firmy Autodesk.

Schematy połączeń oraz projekt płytki PCB zostały zaprojektowane w oprogramowaniu EAGLE [17]. Jest to oprogramowanie firmy Autodesk. Wielkim jego atutem jest możliwość integracji go z oprogramowaniem Fusion360 wykorzystanego do wykonania modelu projektu. Popularność pośród studentów zapewnia istnienie wersji darmowej oraz edukacyjnej, wsparciu społeczności, ogromnej ilości poradników oraz popularności wśród producentów, dzięki czemu można znaleźć wiele odcisków elementów niedostępnych domyślnie we wbudowanych bibliotekach.

4.3. Model stanowiska

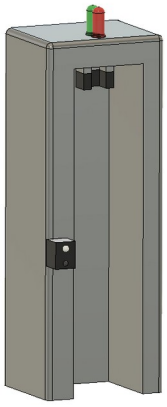
Na całym projekcie, a w szczególności na skrzydle widoczne są podziały modelu. Są one spowodowane faktem, że model został wytworzony za pomocą techniki druku 3D, a następnie złożony z wydrukowanych elementów za pomocą śrub oraz kleju. Do wykonania projektu wybrany został rodzaj bramy przesuwnej jednoskrzydłowej.

Na rysunku 4.3.1 ukazany jest widok na model od frontalnej strony. Z tej perspektywy możemy dojrzeć dwa słupki oraz skrzydło bramy, które stanowią jej główną część. Skrzydło bramy przesuwa się, dzięki zamontowanym w nim dwóch kołach. Dodatkowo dostrzec można, dwie diody znajdujące się na lewym słupku, które mają za zadanie sygnalizować w jakim momencie pracy jest brama oraz czy możliwy jest w danym momencie przejazd.



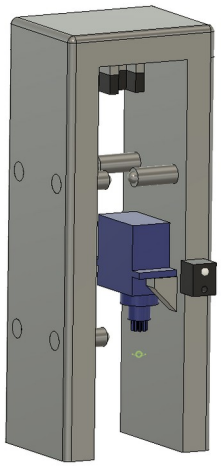
Rys. 4.3.1 Model bramy – widok frontalny

Lewy słup prezentuje się tak jak na rysunku 4.3.2. Na górze widoczne są wcześniej wspomniane diody LED. Wewnątrz dostrzec można zamodelowany jeden z transoptorów szczelinowych działający jako krańcówka. Ten czujniki informuje nas, że brama jest w tym momencie zamknięta. Kolejnym elementem widocznym jest jedna z dwóch części czujnika przerwania wiązki.



Rys. 4.3.2 Model bramy – słup lewy

Kolejny element bramy ukazany na rysunku 4.3.3 to prawy słup. Widzimy, że wewnątrz zamodelowana jest druga krańcówka, informująca nas o tym że brama jest otwarta. Na ścianie umiejscowiona jest druga część czujnika przerwania wiązki. Wewnątrz widzimy zamodelowany serwomechanizm modelarski, który służy jako napęd. Przymocowane jest do niego koło zębate, które z kolei porusza zębatką przymocowaną do skrzydła bramy. Dostrzec możemy też wystające z wewnętrznych ścian walce zakończone kulami imitujące prowadnicę, dzięki, którym brama przesuwa się po odpowiednim torze i pozostaje ciągle w pozycji pionowej.



Rys. 4.3.3 Model bramy – słup prawy

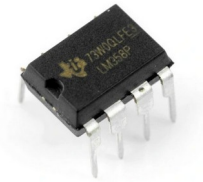
Skrzydło bramy zostało podzielone na kilka elementów. Po lewej stronie widzimy jeden dłuższy element, jest on wydłużony w celu ułatwienia wykrywania go przez czujnik szczelinowy. Wzdłuż bramy dostrzec można zębatkę, dzięki której poruszane jest całe skrzydło poprzez siłę przenoszoną z koła zębatego znajdującego się na serwomechanizmie.

Rys. 4.3.4 Model bramy – skrzydło

4.4. Użyte komponenty, schematy połączeń elektrycznych

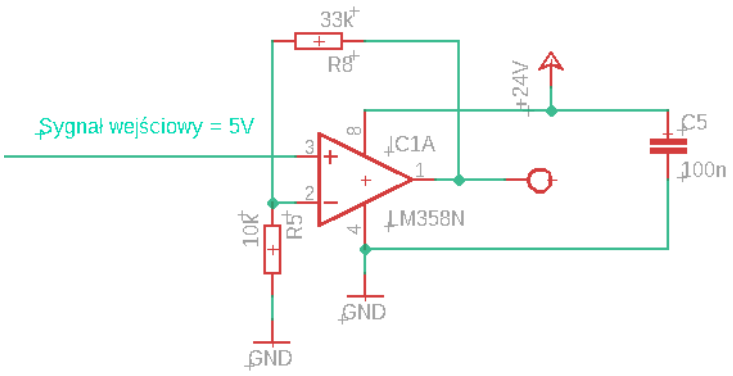
W tym podrozdziale opisane zostały wszystkie czujniki jakie zostały użyte w modelu bramy oraz wszystkie elementy elektroniczne jakie zostały wykorzystane przy tworzeniu płytki PCB. Głównym zadaniem wykonanej płytki jest konwersja sygnałów czujników działających w zakresie napięć do 5 V, natomiast wejścia sterownika SU1.7 działają w zakresie napięć do 24 V.

Wzmacniacz LM358P/N [8] – to bardzo popularny układ, przy którym odpowiedni dobór elementów pasywnych pozwala na modyfikowanie sygnału w wielu różnych możliwościach (rys. 4.4.1). Jest to wzmacniacz dwukanałowy, co pozwala na wzmocnienie jednocześnie dwóch sygnałów.



Rys. 4.4.1 Wzmacniacz operacyjny LM358P/N

Został on użyty na płytce PCB, aby wygenerować wzmocnienie sygnału z czujników. W odpowiedniej konfiguracji z rezystorami (rys. 4.4.2), pozwala on na wzmocnienie sygnału 5 V na 22.5 V, czyli napięcie które przez sterownik jest wykrywane jako stan wysoki. Do wejścia nieodwracającego został podłączony sygnał z czujnika o wartości 0 V lub 5 V, do wejścia odwracającego została podłączona masa układu poprzez rezystor 10 kΩ oraz sygnał wyjściowy wzmacniacza poprzez rezystor 33 kΩ. Wzmacniacz jest zasilany napięciem 24 V, dodatkowo zasilanie jest filtrowane poprzez kondensator monolityczny 100 nF, który ma odfiltrowywać wysokoczęstotliwościowe zakłócenia.

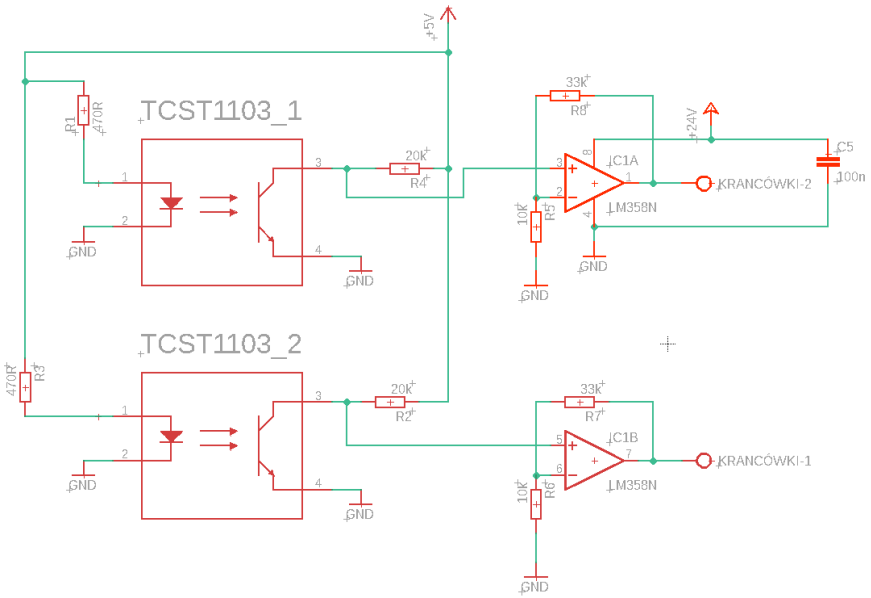


Rys. 4.4.2 Wzmacniacz operacyjny LM358P/N schemat połączeń

Czujnik optyczny TCST1103 [10] – użyte zostały dwa czujniki tego typu, działające jako krańcówki (rys. 4.4.3). Działają one na zasadzie diody świecącej na fototranzystor. Fototranzystor odpowiednio reaguje podczas zasłaniania światła generowanego przez diodę.

Rys. 4.4.3 Czujnik optyczny TCST1103

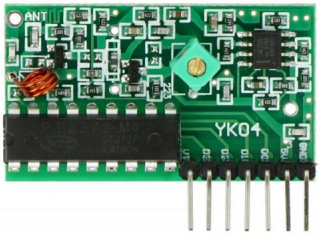
W konfiguracji pokazanej na rysunku 4.4.4, czujnik na wyjściu generuje napięcie 5 V, gdy pomiędzy diodą a fototranzystorem znajduje się przeszkoda. Dioda pozostaje zasilona poprzez rezystor 470 Ω, fototranzystor został podciągnięty do zasilania 5 V poprzez rezystor 20 kΩ do kolektora, natomiast emiter został podłączony do masy układu. Wspomniany wcześniej wzmacniacz LM368P/N dokonuje wzmocnienia sygnału.



Rys. 4.4.4 Czujniki szczelinowe TCST1103 - schemat połączeń

Moduł radiowy [11] - do sterowania bramą wykorzystany został 4-kanałowy moduł radiowy (rys. 4.4.5) wraz z pilotem (rys. 4.4.6). Pilot posiada 4 przyciski: „A”, „B”, „C”, „D”, a wciśnięcie każdego z nich powoduje ukazanie się stanu wysokiego

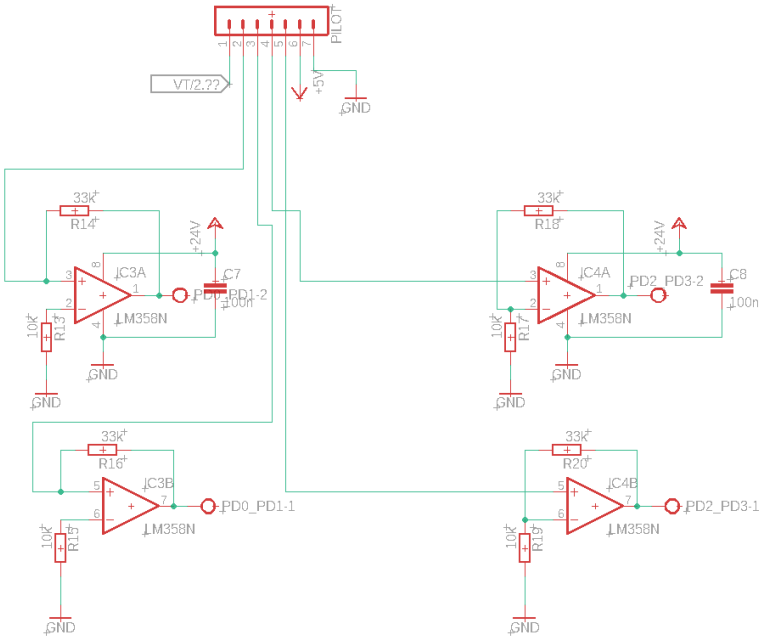
na odpowiednim wyjściu modułu. Dodatkowo w module znajduje się pin „VT” na którym pokazuje się stan wysoki, gdy jakikolwiek z przycisków zostanie wciśnięty.



Rys. 4.4.5 Moduł radiowy

Rys. 4.4.6 Pilot do modułu radiowego

Każde wyjście modułu zostało podpięte do wcześniej wspomnianego wzmacniacza operacyjnego. Wyjścia zostały pogrupowane po dwa, natomiast wyjście „VT” zostało pogrupowane razem z sygnałem przychodzącym z czujnika przerwania wiązki. Użyty moduł radiowy jest to element elektroniczny z gotowym modułem cyfrowym, więc nie jest wymagane dobieranie do niego jakichkolwiek elementów pasywnych. Na płytce znajdują się wszystkie takie elementy oraz kondensatory filtrujące zasilanie. Dodatkowo znajduje się tam dekoder PT2272-M4, który odpowiednio konwertuje sygnały odbierane z pilota przez wbudowaną antenę.

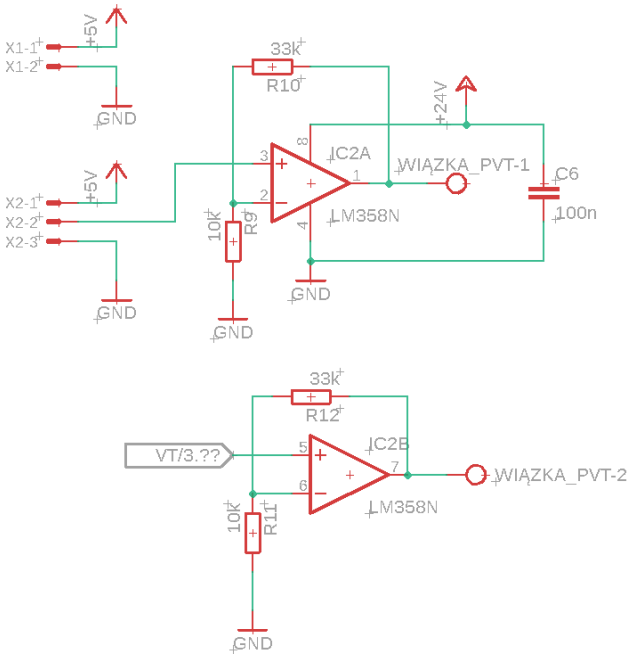


Rys. 4.4.6 Moduł radiowy - schemat połączeń

Czujnik przerwania wiązki [12] (rys. 4.4.7) użyty w projekcie ma zasięg do 50 cm. Pierwsza część to dioda LED IR 3 mm o kącie świecenia 10°. Odbiornik to fototranzystor. Czujnik reaguje na przerwanie wysyłanej z nadajnika do odbiornika wiązki światła podczerwonego.

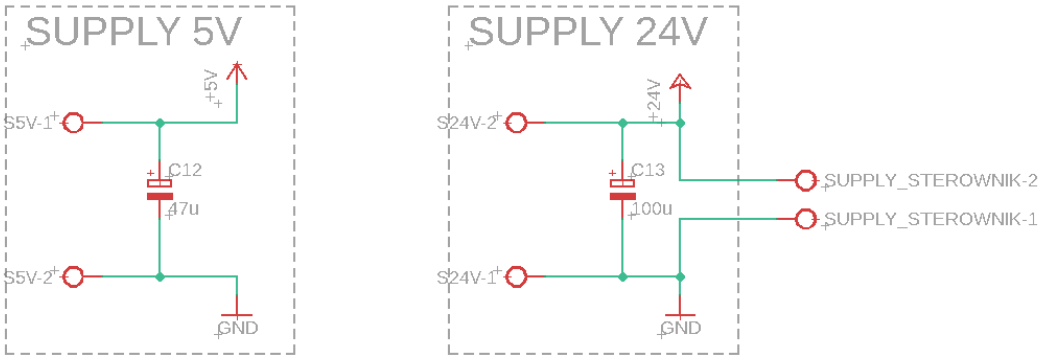
Rys. 4.4.7 Czujnik przerwania wiązki

Po przerwaniu wiązki przez obiekt trzeci, wyjście tranzystorowe typu otwarty kolektor odbiornika zostanie zwarte do masy. Urządzenie może być zasilane napięciem 5 V lub 3,3 V (dla 5 V zasięg będzie większy) (rys. 4.4.8). Odbiornik posiada wyjście tranzystorowe typu otwarty kolektor. Odbiornik generuje stan wysoki na wyjściu, natomiast, gdy wiązka światła zostanie przerwana na wyjściu pojawia się stan niski.



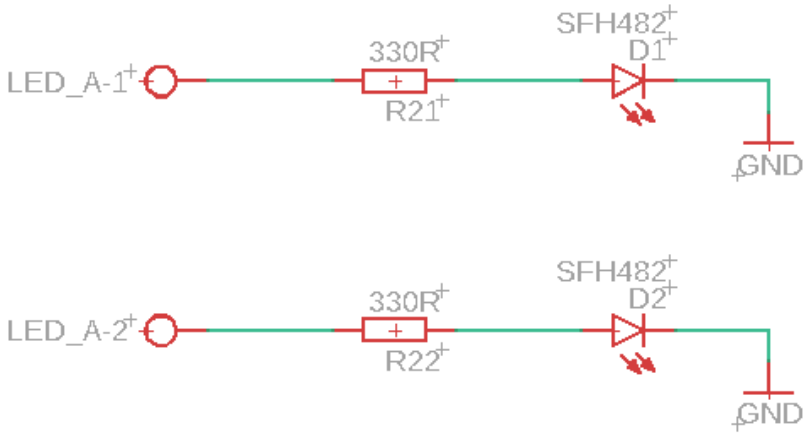
Rys. 4.4.8 Czujnik przerwania wiązki - schemat połączeń

Zasilanie płytki odbywa się z dwóch zasilaczy 24 V i 5 V. Napięciem 24 V zasilane są wzmacniacze oraz sterownik, natomiast napięciem 5 V zasilane są czujniki oraz silnik. Dodatkowo bezpośrednio przy linii zasilania zostały podpięte kondensatory elektrolityczne (47 µF dla 5 V, 100 µF dla 24 V), które mają za zadanie zapobiegać skokom napięć. Masy obydwu zasilaczy zostały ze sobą zwarte, aby sterownik posiadał jednakowy punkt odniesienia.



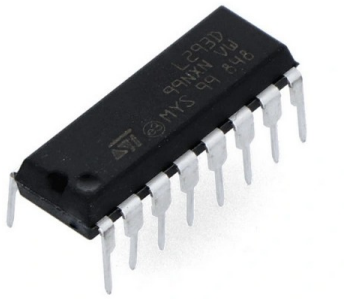
Rys. 4.4.9 Zasilanie układu elektronicznego

Diody sygnalizujące zostały podpięte do napięcia 5 V poprzez rezystory 330 Ω.



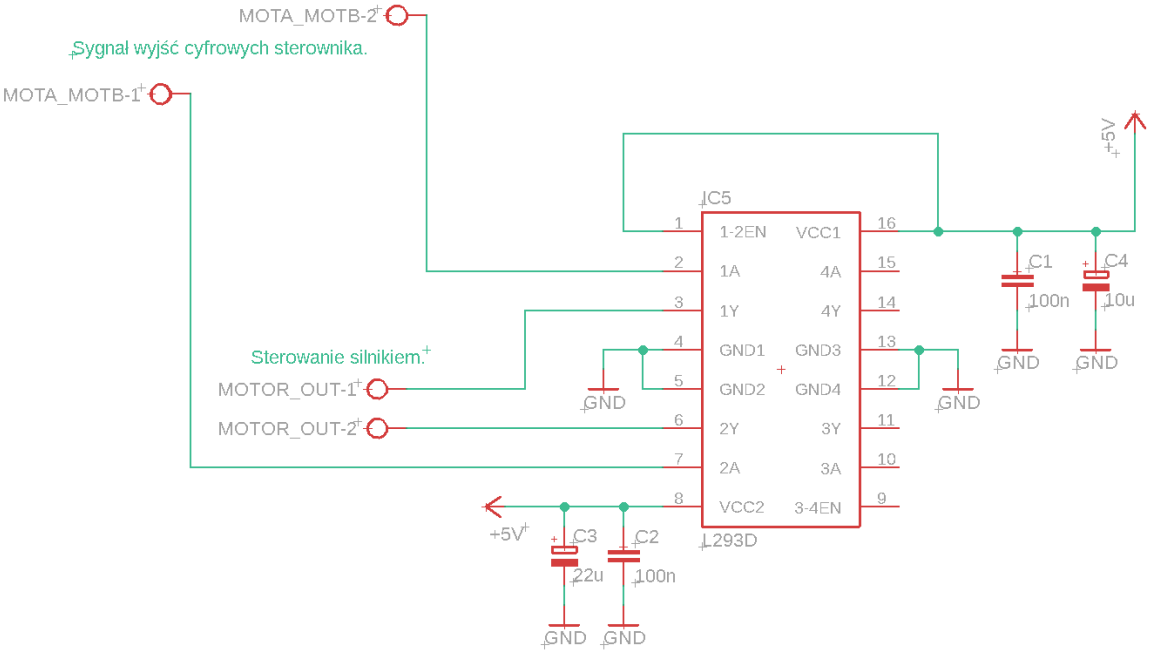
Rys. 4.4.10 Diody - schemat połączeń

Sterownik silników [13] użyty w projekcie to moduł L293D (rys, 4.4.11). Jest to bardzo popularny mostek H wykorzystywany w elektronice. Ten dwukanałowy układ scalony, przeznaczony jest do zastosowania z napięciami do 36 V i natężeniem nie większym niż 0,6 A na jeden kanał (chwilowe natężenie może sięgać maksymalnie 1,2 A na jeden kanał).



Rys. 4.4.11 Mostek H - L293D

Mostek H na wyjściach sterujących otrzymuje sygnał 5 V pochodzący z wyjść tranzystorowych sterownika. Zasilanie również pochodzi z napięcia 5 V. Jako, że sterownik nie posiada wyjść PWM, poprzez które można sterować prędkością silnika w użytym mostku, do wejścia PWM modułu został podpięty sygnał 5 V, który zastępuje sygnał z maksymalnym wypełnieniem, więc silnik działa ciągle na maksymalnej prędkości. Nie jest to przeszkodą, gdyż silnik użyty w modelu to odpowiednio przerobiony serwomechanizm modelarski, więc posiada on wewnątrz przekładnię, która na wyjściu generuje prędkość obrotową bliską porządnej. Prędkość obrotowa serwomechanizmu jest na tyle odpowiednia, że niepotrzebne było użycie jakichkolwiek przekładni do sterowania bramą. Na schemacie (rys. 4.4.12), widzimy dodatkową filtrację zasilania poprzez kondensatory 22 µF oraz 100 nF oraz filtrację zasilania części logicznej mostka poprzez kondensatory 10 µF i 100 nF. Wyjścia mostka są podpięte bezpośrednio do silnika w serwomechanizmie.



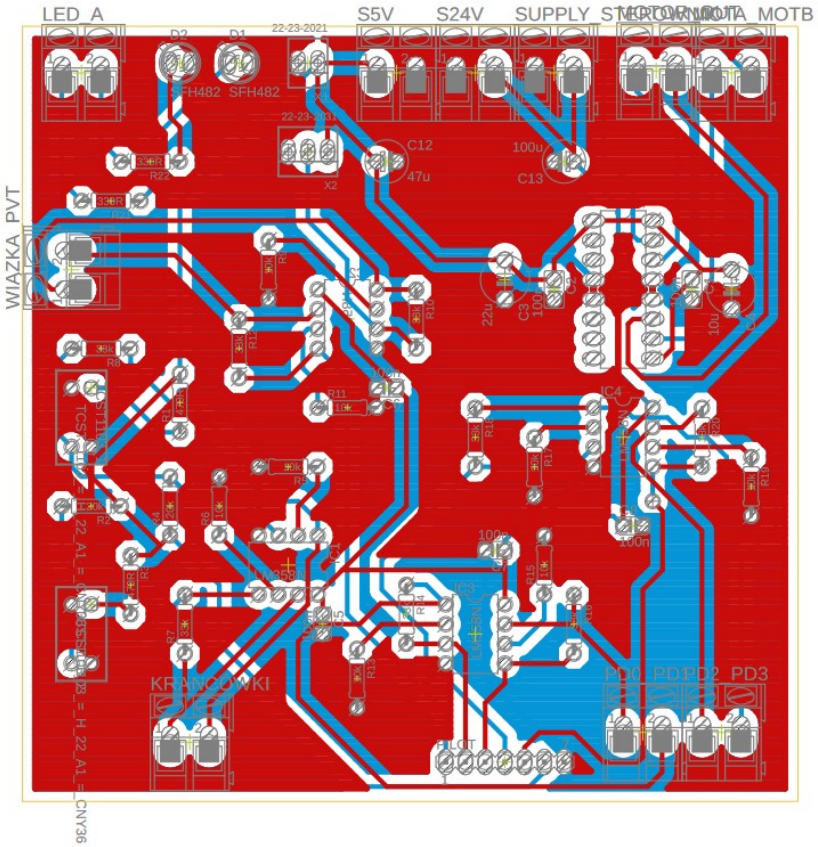
Rys. 4.4.12 Mostek H L293D - schemat połączeń

Serwomechanizm modelarski [22] – napęd bramy w projekcie stanowi Serwo SG-90 - micro – 180. Zostało ono odpowiednio dostosowane. Z serwomechanizmu wyciągnięty został układ sterujący, tak aby wewnątrz pozostał sam silnik oraz przekładnia. Dzięki temu można sterować silnikiem za pomocą mostka H. Dodatkowo przekładnia została tak zmodyfikowana, aby umożliwić pracę serwomechanizmu w pełnym zakresie.

Rys. 4.4.13 Serwomechanizm modelarski

4.5. Wykonana płytka PCB

Wcześniej ukazane schematy, wygenerowały na płytce PCB za pomocą oprogramowania EAGLE odpowiednie połączenia. Pozostało tylko odpowiednio rozmieścić elementy na płytce. Pierwszym krokiem było skorzystanie ze skryptu dostępnego w starszych wersjach oprogramowania. Jest to skrypt ULP o nazwie „place50”, który dokonuje pogrupowania elementów według wykonanych schematów. Konieczne jednak było samodzielne rozmieszczenie elementów na płytce oraz dobranie odpowiedniej grubości ścieżek. Za pomocą narzędzia „Autoroute” zbadane zostało według trzech różnych algorytmów czy możliwe jest wykonanie połączeń w takiej konfiguracji. Ostatecznie konieczne było poprawienie poprowadzonych ścieżek, tak aby poprowadzenia spełniały wszystkie wymogi firmy wykonującej płytkę PCB. Płytka jest dwustronna, czyli posiada ona dwie warstwy (rys. 4.5.1).

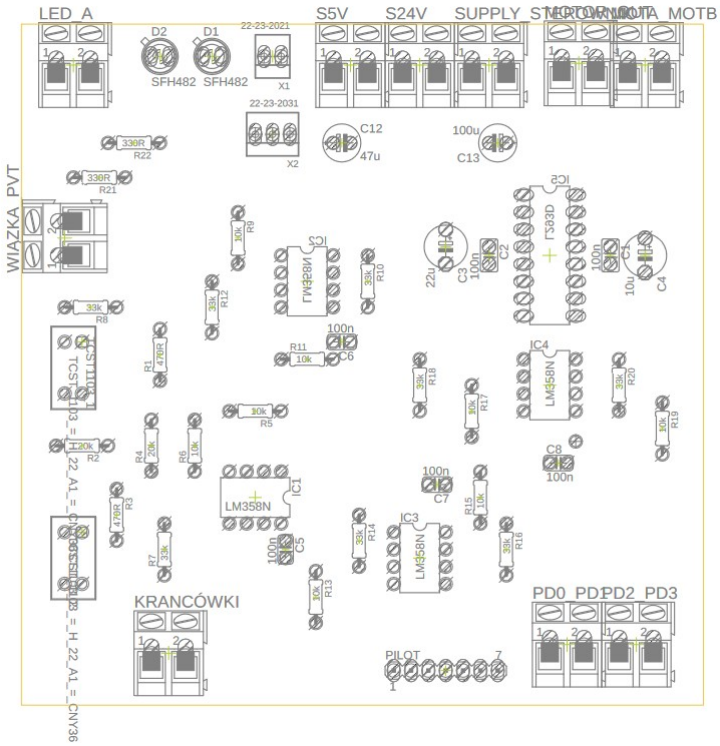


Rys. 4.5.1 Płytka PCB – widok całościowy

Na rysunku 4.5.2 widzimy rozmieszczone elementy bez połączeń, co pozwala na uzyskanie większej przejrzystości. Widzimy tam kolejno rozmieszczone:

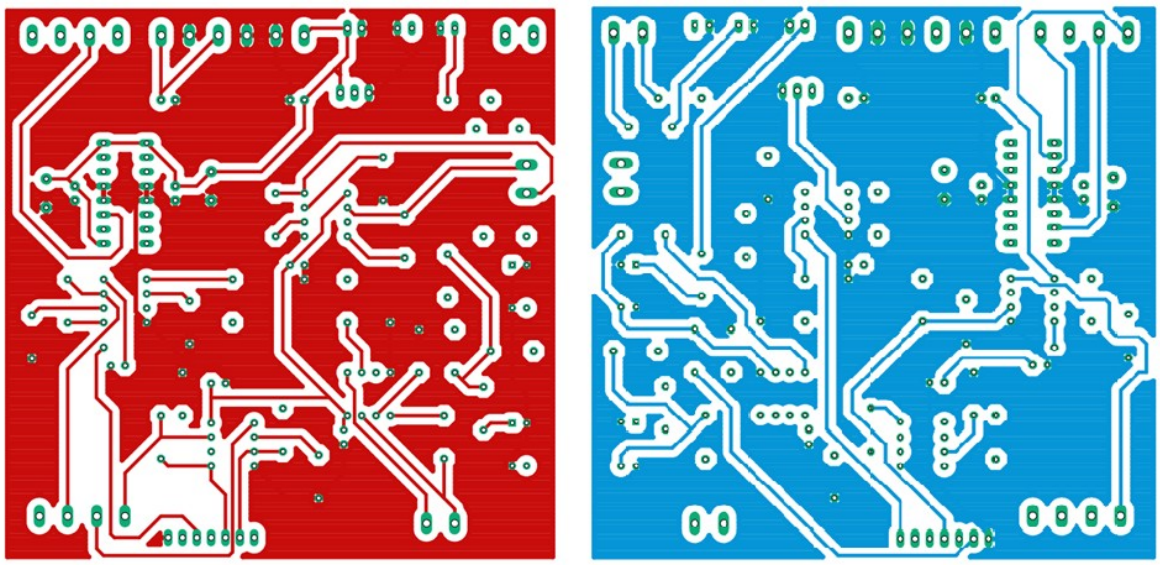
1) W prawym górnym rogu widzimy złącza śrubowe, kolejno od lewej: wejścia sterujące mostka H podłączone do wyjść tranzystorowych sterownika, wyjścia sygnałów sterujących silnikiem, zasilanie sterownika, podłączenie zasilania 24 V, podłączenie zasilania 5 V.
2) W lewym górnym rogu widzimy złącze śrubowe, gdzie podłączony jest sygnał wyjścia tranzystorowego sterownika odpowiadające za sterowanie diodami LED.
3) Po lewej stronie od złączy zasilania znajdują się diody LED.
4) Po centralnej lewej stronie widzimy złącza śrubowe, które są wyjściami sygnałów czujnika przerwania wiązki oraz wyjścia PVT modułu radiowego.
5) Dolne złącza śrubowe są to kolejno wyjścia sygnałów krańcówek oraz wszystkich czterech wyjść modułu radiowego.
6) Po lewej, dolnej stronie widzimy transoptory szczelinowe wraz ze wszystkimi elementami pasywnymi.
7) Po centralnej prawej stronie znajduje się mostek H, L293D.

8) W centrum płytki znajdują się cztery wzmacniacze wraz ze wszystkimi rezystorami oraz kondensatorami.



Rys. 4.5.2 Płytka PCB - rozmieszczenie elementów

Aby dojrzeć jak zostały wykonane ścieżki połączeniowe, poniżej zostały zamieszczone dwa wydruki widoku płytki, są to: warstwa górna po lewej stronie oraz warstwa dolna znajdująca się po prawej stronie (rys. 4.5.3).



Rys. 4.5.3 Płytka PCB – warstwa TOP i BOTTOM

Wszystkie ścieżki na obydwu warstwach płytki posiadają szerokość 24 mils (0,6096 mm), za wyjątkiem ścieżki sygnałowej GND, której szerokość wynosi 32 mils (0,8128 mm). Dodatkowo płytka została z dwóch stron wypełniona miedzią na wolnych przestrzeniach, w odpowiedniej odległości od innych ścieżek. Dodatkowa ilość miedzi jest połączona z sygnałem GND. Zostało to wykonane za pomocą funkcji „Polygon”, a następnie „Ratsnest”.

5. Program sterujący

W poniższym rozdziale przedstawione zostało rozwiązanie problemu sterowania bramą za pomocą automatu stanowego [18][20][21]. Zamieszczono najważniejsze fragmenty kodu programu głównego oraz bloków funkcjonalnych wraz z opisami koniecznymi do zrozumienia koncepcji działania.

5.1. Automat stanowy programu głównego

Program główny, został napisany na podstawie grafu automatu stanowego (rys 5.1.1). Jest to klasyczny automat stanowy, zaimplementowany w programie za pomocą instrukcji warunkowej *case*. Zauważyć można, że graf posiada sześć stanów, które oznaczają kolejno:

1 – brama jest aktualnie zamknięta

2 – otwieranie bramy

3 – brama jest aktualnie otwarta

4 – zamykanie bramy

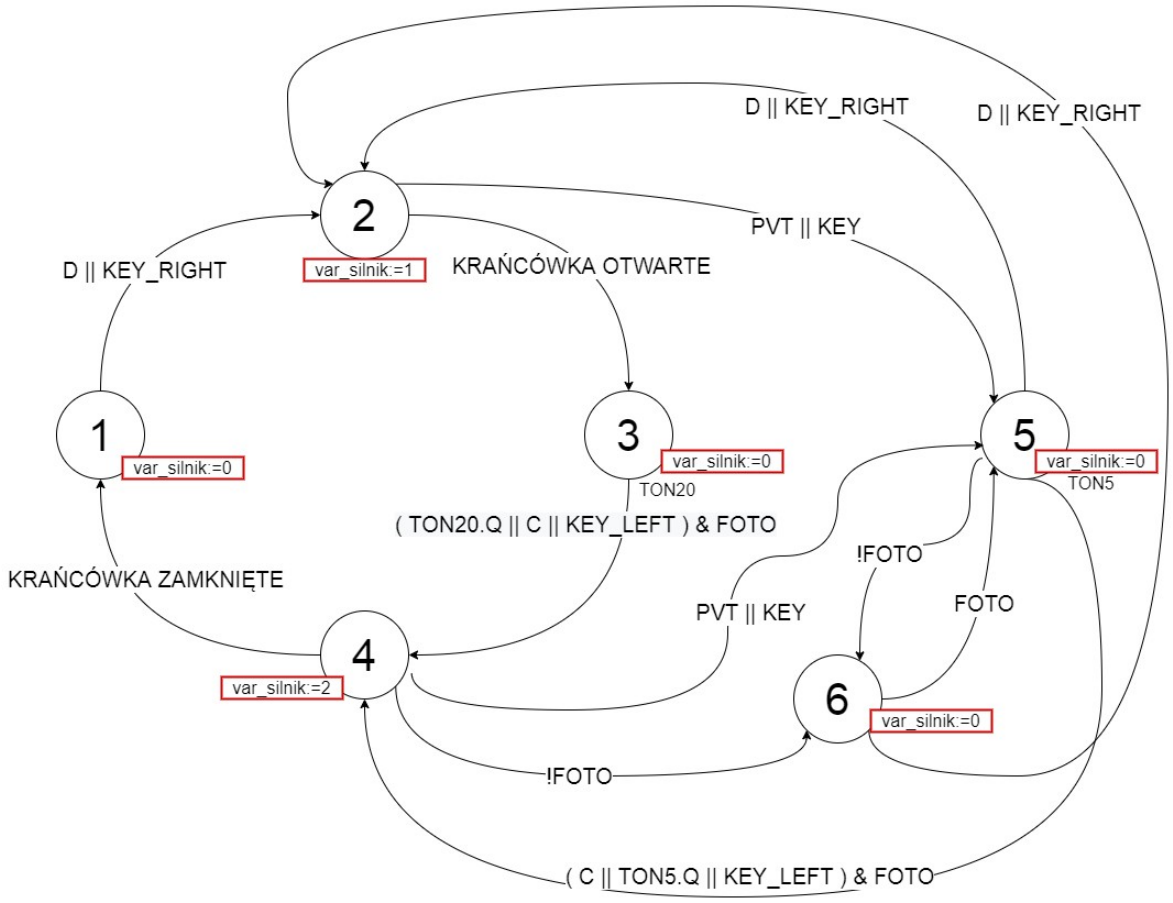
5 – brama oczekuje na wciśnięcie konkretnego przycisku na sterowniku, pilocie lub usunięcie przeszkody wykrywanej przez czujnik przerwania wiązki

6 - podczas zamykania bramy czujnik przerwania wiązki wykrył przeszkodę i ciągle znajduje się ona pomiędzy słupkami bramy

Jest to automat działania trybu automatycznego bramy. Idea działania jest następująca. Brama docelowo czeka jako zamknięta, automat znajduje się wtedy w stanie pierwszym. Po naciśnięciu na pilocie przycisku „D” lub prawego przycisku na klawiaturze sterownika brama zaczyna się otwierać i program przechodzi do stanu drugiego. Po odczytaniu sygnału z krańcówki prawej, która oznacza że brama jest otwarta następuje zakończenie pracy silnika i brama przechodzi w tryb oczekiwania na zamykanie. Tryb ten można przerwać naciskając przycisk „C” na pilocie lub lewy przycisk na klawiaturze. Dodatkowo zamykanie bramy spowoduje pojawienie się stanu wysokiego na wyjściu „Q” czasomierza „TON20”, który zostaje uruchomiony po wejściu automatu w stan numer 3, a czas jego zliczania trwa 20 sekund. Dodatkowo spełniony musi być warunek, że żadna przeszkoda nie może znajdować się na drodze skrzydła bramy, więc czujnik przerwania wiązki musi przesyłać sygnał o stanie wysokim. Brama w stanie czwartym będzie się zamykać do momentu, aż krańcówka sygnalizująca że brama jest zamknięta nie zmieni sygnału wyjściowego na wysoki. Zamykanie można przerwać dowolnym przyciskiem na klawiaturze lub pilocie, wtedy brama przechodzi do stanu piątego. Z tego stanu brama

może zacząć się ponownie zamykać po naciśnięciu przycisku „C” lub lewego przycisku na klawiaturze (konieczne jest aby nie było przeszkody pomiędzy słupkami bramy), otwierać po wciśnięciu przycisku „D” lub prawego klawisza z klawiatury. Jeśli żadna z tych czynności nie zostanie wykonana, to po upływie 5 sekund, czyli po pojawieniu się stanu wysokiego na wyjściu „Q” czasomierza „TON5”. Po pojawieniu się przeszkody na drodze skrzydła bramy automat ze stanu czwartego, czyli zamykania bramy, przechodzi do stanu szóstego. Dopiero po zniknięciu przeszkody, automat przejdzie we wcześniej wspomniany stan numer pięć.

Wyjścia diod LED sterownika są uwarunkowane instrukcją warunkową *if*, która zostanie opisana w dalszej części pracy. Automat stanowy programu głównego, bezpośrednio steruje tylko wyjściami tranzystorowymi, które warunkują pracę serwomechanizmu [18]. Służy do tego zmienna „var_silnik” typu integer, która po uzyskaniu wartości 0, dokonuje zahamowania silnika, po otrzymaniu wartości 1 dokonuje otwierania bramy, natomiast ustawienie tej zmiennej na wartość 2 powoduje zamykanie się bramy.



Rys. 5.1.1 Graf stanowy programu głównego

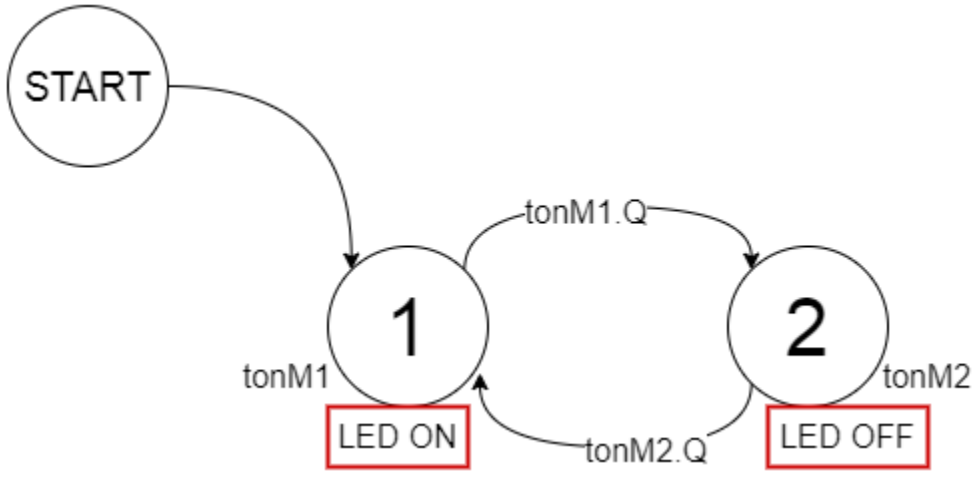
Ważnym aspektem systemu sterowania jest możliwość działania w dwóch trybach:

Automatycznym – jest to tryb działania uwarunkowany poprzez graf stanowy ukazany na rysunku 5.1.1.

Manualnym – jest to tryb działania, w którym użytkownik dowolnie operuje działaniem bramy.

Tryb manualny działa w sposób następujący. Podczas wciskania na pilocie przycisku „C” lub przycisku „LEFT” brama jest zamykana, wciskanie przycisku „D” lub przycisku „RIGHT” powoduje otwieranie się bramy. Ważnym aspektem jest fakt, że system sterowania będąc w trybie pracy manualnej nie reaguje na odczyty czujnika przerwania wiązki, brama jest całkowicie zależna od użytkującego. System reaguje jednak na odczyty krańcówek i nie pozwala na ich przekroczenie. Przełączenie się pomiędzy trybami odbywa się za pomocą następujących przycisków: wciśnięcie przycisku „A” na pilocie lub „UP” na sterowniku, powoduje włączenie trybu automatycznego, wciśnięcie przycisku „B” na pilocie lub „DOWN” na sterowniku powoduje włączenie trybu manualnego. Uwzględnione w programie zabezpieczenie pozwala na płynne przechodzenie pomiędzy dwoma trybami w dowolnym momencie działania bramy.

Drugim grafem jest graf automatu stanowego według którego działa blok funkcjonalny „MRUGANIE” (rys. 5.1.2). Odpowiada on za mruganie czerwoną diodą LED podczas, gdy brama jest w stanie poruszania się lub oczekiwania w momencie gdy nie jest ani zamknięta ani otwarta. Działanie automatu jest następujące. Blok funkcjonalny nie posiada żadnych zmiennych wejściowych, zostaje on wywołany w odpowiednim momencie działania programu, gdy żadna z krańcówek nie jest załączona. Po wywołaniu bloku, rozpoczyna się praca automatu, wynikiem czego jest mruganie czerwonej diody LED. Jej czas pracy wynosi 1 sekundę, natomiast wypełnienie 50%.



Rys. 5.1.2 Graf stanowy programu bloku funkcjonalnego MRUGANIE

5.2. Zmienne wejściowe oraz wyjściowe zdefiniowane w projekcie

Pierwszą rzeczą, która zostanie opisana przed opisem bloków funkcjonalnych będą zmienne globalne użyte w projekcie. Wiele zmiennych zostało zaimportowane do projektu z programu przykładowego dostarczonego wraz ze sterownikiem. Użyte zmienne wejściowe zostały zawarte w tabeli 5.2.1. W pierwszej kolumnie widzimy nazwę zmiennej, pod jaką używana jest w programie. Druga kolumna to typ zmiennej. Trzecia kolumna to opis zmiennej, czyli od jakiego czujnika pochodzi dana zmienna, natomiast w czwartej kolumnie widzimy adres po którym wgrany na procesor bootloader odnajduje wejścia podłączone do procesora.

Tabela 5.2.1 Zmienne wejściowe bloku funkcjonalnego MRUGANIE

Nazwa zmiennej	Typ	Opis	Adres
DI_0	BOOL	Krańcówka lewa – brama zamknięta	%0000
DI_1	BOOL	Krańcówka prawa – brama otwarta	%0001
DI_2	BOOL	Czujnik przerwania wiązki	%0002
DI_3	BOOL	Pilot – wyjście PVT	%0003
DI_4	BOOL	Pilot – przyciska „A”	%0004
DI_5	BOOL	Pilot – przyciska „C”	%0005
DI_6	BOOL	Pilot – przyciska „B”	%0006
DI_7	BOOL	Pilot – przyciska „D”	%0007
KEY_LEFT	BOOL	Klawiatura wbudowana w sterownik – przycisk LEFT	%0033
KEY_RIGHT	BOOL	Klawiatura wbudowana w sterownik – przycisk RIGHT	%0034
KEY_UP	BOOL	Klawiatura wbudowana w sterownik – przycisk UP	%0035
KEY_DOWN	BOOL	Klawiatura wbudowana w sterownik – przycisk DOWN	%0036

W tabeli 5.2.2 widzimy natomiast zmienne wyjściowe. Są to zmienne połączone z wyjściami tranzystorowymi sterownika. Wyjścia tranzystorowe sterownika mają możliwość zasilania z osobnego źródła, podpięte więc do nich zostało napięcie 5 V, które po ustawieniu zmiennej wyjściowej na stan wysoki zostaje przekazane na wyjście.

Tabela 5.2.2 Zmienne wejściowe bloku funkcjonalnego MRUGANIE

Nazwa zmiennej	Typ	Opis	Adres
DO_4	BOOL	Wejścia pierwsze mostka H – L293D	%0012

DO_5	BOOL	Wejście drugie mostka H – L293D	%0013
DO_6	BOOL	Czerwona dioda LED	%0014
DO_7	BOOL	Zielona dioda LED	%0015
LCD_LINE_1	STRING[16]	16-znakowa tablica znaków, której zmiana powoduje wyświetlenie adekwatnego komunikatu na pierwszej linii wyświetlacza cyfrowego	-
LCD_LINE_2	STRING[16]	16-znakowa tablica znaków, której zmiana powoduje wyświetlenie adekwatnego komunikatu na drugiej linii wyświetlacza cyfrowego	-

5.3. Budowa programu

Drzewo projektu jednostek POU, prezentuje się tak jak na rysunku 5.3.2. Widzimy tutaj utworzone dwa bloki funkcjonalne o nazwach „MRUGANIE”, „STEROWANIE” oraz plik POU programu głównego o nazwie „MAIN”.

Wszystkie programy, zarówno bloków funkcjonalnych jak i głównego programu zostały napisane w języku ST z powodów wymienionych w ostatnim rozdziale pracy. Dodatkowym atutem języka ST jest przejrzystość kodu. Wiele operacji, które znajdują się w wykonanych programach, w języku drabinkowym LD czy FBD zajęły by wiele linii, a wynikiem tego byłoby znaczne pogorszenie czytelności kodu programu.

Każdy program w języku ST ma narzuconą budowę, rozpoczyna się on od linii `PROGRAM PROG` w przypadku programu, gdzie `PROGRAM`, wskazuje na typ POU natomiast `PROG` jest to nazwa danego programu. Program kończy się natomiast linią `END_PROGRAM`, tak jak jest to ukazane na listingu 5.3.1.

```
PROGRAM PROG
 VAR_EXTERNAL (*$AUTO*) END_VAR

 VAR END_VAR
```

Listing 5.3.1 Domyślna budowa struktury programu

W przypadku bloku funkcjonalnego kod bloku rozpoczyna się od linii `FUNCTION_BLOCK BLOK`, gdzie analogicznie jak w poprzednim przypadku `FUNCTION_BLOCK` to zdefiniowanie typu POU, natomiast `BLOK` jest to nazwa danego bloku. Zakończenie struktury danego bloku jest linią `END_FUNCTION_BLOCK`, widoczną na listingu 5.3.2.

```
FUNCTION_BLOCK BLOK

VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
```

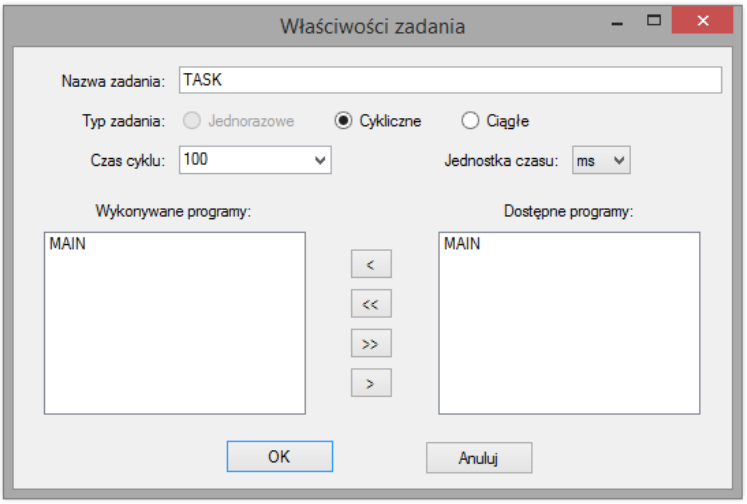
```
END_FUNCTION_BLOCK

END_PROGRAM
```

Listing 5.3.2 Domyślna budowa struktury bloku funkcjonalnego

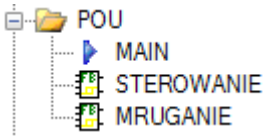
Częścią wspólną programów, funkcji oraz bloków funkcjonalnych są klamry pozwalające na odpowiednie rozmieszczenie kolejnych fragmentów kodu. Pierwszą klamrą VAR_EXTERNAL…END_VAR, w której znajdować powinny się wszystkie zmienne globalne użyte w danym programie lub bloku funkcjonalnym. Dla ułatwienia zastosować można następujące rozwiązanie VAR_EXTERNAL (*$AUTO*) END_VAR, gdzie zastosowanie komentarza (*$AUTO*) pozwala na użycie w programie wszystkich zmiennych globalnych. Kolejna klamra VAR…END_VAR, jest to miejsce przeznaczone na implementację zmiennych lokalnych danego POU. Wymienione klamry automatycznie generowane są tylko w programach, w blokach funkcjonalnych nie są one generowanie automatycznie, a programista musi je dopisać ręcznie w przypadku gdy będzie to konieczne. W blokach funkcjonalnych generowane są natomiast dwie inne klamry, są to VAR_INPUT…END_VAR, w której zamieszczamy wszystkie zmienne wejściowe, uwzględnione podczas wywoływania bloku oraz VAR_OUTPUT…END_VAR, w której umieszczane są zmienne wyjściowe.

Program główny jest jedynym programem działającym niezależnie w całym projekcie (rys. 5.3.1). Zadanie o nazwie „TASK”, które zostało utworzone działa cyklicznie, a czas cyklu wynosi 100ms.



Rys. 5.3.1 Zdefiniowany TASK projektu

Całościowo utworzone drzewo jednostek organizacyjno programowych POU, prezentuje się tak jak na rysunku 5.3.2. Widzimy tam utworzony program główny o nazwie „MAIN” oraz dwa bloki funkcjonalne: „MRUGANIE” oraz „STEROWANIE”.



Rys. 5.3.2 Drzewo projektu - jednostki organizacyjno-programowe POU

5.4. Programy bloków funkcjonalnych

Pierwszy blok funkcjonalny o nazwie MRUGANIE, definiuje działanie czerwonej diody LED, podczas gdy brama nie jest ani otwarta ani zamknięta. Do obliczenia czasu wykorzystane zostały dwa czasomierze TON, zdefiniowane jako bloki funkcjonalne we wbudowanych w środowisko CPDev bibliotekach.

Jako pierwsze konieczne jest zdefiniowanie czasomierzy TON oraz zmiennej „stan_mruganie” typu integer, która domyślnie przyjmuje wartość początkową „1” a w programie pełni funkcje zmiennej przechowującej informacje o stanie w jakim właśnie znajduje się automat stanowy.

```
VAR
 tonM1:TON;
 tonM2:TON;
 stan_mruganie:int:=1;
END_VAR
```

Listing 5.4.1 Blok funkcjonalny MRUGANIE - implementacja zmiennych lokalnych

W kodzie programu kolejnym ważnym elementem jest wywołanie bloków funkcjonalnych czasomierzy TON, zdefiniowanie w jakim momencie powinny zostać wywołane oraz czas PT. Ważnym elementem do dodania jest, że CPDev nie pozwala ,a implementację zmiennej PT, która nie jest całkowita więc konieczne jest aby tą zmienną ustawić za pomocą jednostek „ms”.

```
tonM1(IN:=stan_mruganie=1, PT:=t#500ms);
tonM2(IN:=stan_mruganie=2, PT:=t#500ms);
```

Listing 5.4.2 Blok funkcjonalny MRUGANIE - wywołanie bloków czasomierzy TON

Automat stanowy jest zaimplementowany za pomocą instrukcji warunkowej typu *switch*, znanej z języka „C”. W języku ST nazywa się ona *case* i delikatnie różni się składnią niż te w językach wysokopoziomowych.

```
case stan_mruganie of
1: DO_6:=true;
   if tonM1.Q then stan_mruganie:=2; end_if;
2: DO_6:=false;
   if tonM2.Q then stan_mruganie:=1; end_if;
end_case;
```

Listing 5.4.3 Blok funkcjonalny MRUGANIE - program automatu stanowego

Kolejnym blokiem funkcjonalnym jest blok „STEROWANIE”. Odpowiada on za wysterowanie wyjść DO_4 oraz DO_5 w taki sposób, aby odpowiednio wysterować silnik. Posiada on wejściową zmienną „silnik_input”, która przyjmuje odpowiednią wartość podczas wywoływania bloku. Może ona uzyskać wartość:

0 – wtedy program zatrzymuje silnik

1 – wtedy program powoduje otwieranie się bramy

2 – wtedy program powoduje zamykanie się bramy

```
VAR_INPUT
 silnik_input:INT;
END_VAR
```

Listing 5.4.4 Blok funkcjonalny STEROWANIE - implementacja zmiennych wej.

W przypadku tego bloku funkcjonalnego, cały program został zdefiniowany za pomocą instrukcji warunkowej *if*. Na podstawie wartości zmiennej „silnik_input” zostają odpowiednio wysterowane wyjścia tranzystorowe DO_4 oraz DO_5.

```
if silnik_input=0 then DO_4:=false; DO_5:=false;
elsif silnik_input=2 then DO_4:=true; DO_5:=false;
elsif silnik_input=1 then DO_4:=false; DO_5:=true;
end_if;
```

Listing 5.4.5 Blok funkcjonalny STEROWANIE - Instrukcja warunkowa "if"

5.5. Program główny

Program główny jest napisany w oparciu o graf stanowy (rys. 5.1.1). Napisany jest zgodnie z domyślną struktura budowy programu. Wszystkie zmienne globalne zostały

zadeklarowane automatycznie za pomocą klamry VAR_EXTERNAL (*$AUTO*) END_VAR. W miejscu przeznaczonym na deklaracje zmiennych lokalnych zadeklarowane zostały instancje bloków funkcjonalnych „STEROWANIE” o nazwie „silnik” oraz „MRUGANIE” o nazwie „mrug”. Kolejno zadeklarowane zostały instancje dwóch czasomierzy TON o nazwach „ton5” oraz „ton20”. Kolejnym krokiem było zadeklarowanie zmiennych pozwalających na sczytywanie w programie zbocza narastającego wejść cyfrowych dla zmiennych za pomocą bloków funkcjonalnych R_TRIG:
DI_3, DI_4, DI_5, DI_6, DI_7, KEY_RIGHT, KEY_LEFT, KEY_UP, KEY_DOWN.

```
VAR
 silnik:STEROWANIE;
 mryg:MRUGANIE;

 ton5:TON;
 ton20:TON;

DI_3_Narastajace:R_TRIG;

.
.
.

KEY_DOWN_Narastajace:R_TRIG;

END_VAR
```

Listing 5.5.1 Program główny - deklaracja zmiennych, instancji bloków funkcjonalnych oraz czasomierzy

Kolejno przechodząc do fragmentów programu wykonywanych w pętli widzimy zdefiniowane zadeklarowane w listingu 5.4.1 czasomierze. Linia `ton5(IN:=stan=5, PT:=t#5s);` definiuje czasomierz „ton5” o czasie PT równym 5 sekund, wywoływanym w momencie, gdy zmienna „stan” osiągnie wartość 5. Zmienna „stan” jest zadeklarowana jako zmienna globalna, przechowuje ona informację o tym w jakim stanie znajduje się obecnie automat stanowy. Deklaracja czasomierza „ton30”, działa analogicznie do poprzednio opisanego czasomierza „ton5”.

```
ton5(IN:=stan=5, PT:=t#5s);
ton20(IN:=stan=3, PT:=t#20s);
```

Listing 5.5.2 Program główny - definiowanie czasomierzy

Kolejny fragment programu, odpowiada za sczytywanie zbocza narastającego wcześniej wspomnianych zmiennych. Jest to rodzaj zabezpieczenia, działający przeciwko sytuacjom gdy użytkownik przytrzyma jakiś przycisk zbyt długo, a wtedy automat stanowy mógłby zareagować niepożądaną zmianą stanu.

```
(*zbocze narastające*)
 DI_3_Narastajace(CLK:=DI_3);

.
.
.

KEY_DOWN_Narastajace(CLK:=KEY_DOWN);
```

Listing 5.5.3 Program główny - wywołanie instancji bloków R_TRIG

Fragment kodu ukazany na listingu 5.5.4 odpowiada za zmianę wyświetlanych znaków na pierwszej linii wyświetlacza sterownika. w zależności od tego w jakim trybie pracy działa obecnie sterownik wyświetlany zostanie komunikat „AUTOMATYCZNY” lub „MANUALNY”. Jest to sprawdzane na podstawie zmiennej „automatyczny”, która jeśli posiada wartość „true” oznacza, że sterownik znajduje się w trybie pracy automatycznej, wartość „false” natomiast oznacza, że sterownik znajduje się w trybie pracy manualnej.

```
if DI_4 or KEY_UP then automatyczny:=true;
LCD_LINE_1:='AUTOMATYCZNY';
elsif DI_6 or KEY_DOWN then automatyczny:=false;
LCD_LINE_1:='MANUALNY';
end_if;
```

Listing 5.5.4 Program główny - zmiana tekstu na pierwszej linii wyświetlacza

Automat stanowy został zaimplementowany w kodzie za pomocą instrukcji warunkowej *case*. Jak widać na listingu 5.5.5 przejścia warunkowe pomiędzy stanami są wykonywane za pomocą instrukcji warunkowej *if*. Tryb manualny działa na podstawie kodu widocznego w listingu 5.5.6. Zmienna „var_silnik” zostaje zmieniona odpowiednio w każdym stanie oraz podczas działania trybu manualnego. Jest ona również zmienną wejściową instancji bloku funkcjonalnego „silnik:”, wywoływanego w każdym cyklu sterownika co możemy dojrzeć na listingu 5.5.7. Dodatkowo w każdym stanie trybu

automatycznego oraz w dowolnym momencie działania trybu manualnego druga linia wyświetlacza wyświetla odpowiedni komunikat, są to:

1) „BRAMA OTWARTA” – gdy brama jest aktualnie otwarta
2) „BRAMA ZAMNIETA” – gdy brama jest aktualnie zamknięta
3) „OTWIERANIE” – gdy brama jest aktualnie otwierana lub automatycznie otwiera się
4) „ZAMYKANIE” – gdy brama jest zamykana lub automatycznie zamyka się
5) „OCZEKIWANIE” - gdy automat stanowy znajduje się w stanie 5, czyli brama czeka 5 sekund i rozpoczyna proces zamykania się lub reakcję ze strony użytkującego i odpowiednio reaguje.
6) „PRZESZKODA” - gdy automat stanowy znajduje się w stanie 6, czyli brama jest w trybie oczekiwania, aż przeszkoda zniknie spomiędzy słupków lub użytkujący podejmie działanie aby ją otworzyć.

```
(*działanie trybu automatycznego*)
if automatyczny=TRUE then
 case stan of
 1: var_silnik:=0;
    LCD_LINE_2:='BRAMA ZAMKNIETA';
    if  (DI_7_Narastajace OR KEY_RIGHT_Narastajace) then
stan:=2; end_if;

.
.
.

  6:var_silnik:=0;
    LCD_LINE_2:='PRZESZKODA';
    if DI_2 then stan:=5;
    elsif (DI_7_Narastajace or KEY_RIGHT_Narastajace) then
stan:=2;
    end_if;

  end_case;
```

Listing 5.5.5 Program główny - implementacja automatu stanowego, działanie trybu automatycznego

```
(*działanie trybu manualnego*)
 else
    if DI_1 then stan:=3; var_silnik:=0; LCD_LINE_2:='BRAMA
OTWARTA';
    elsif DI_0 then stan:=1; var_silnik:=0;
LCD_LINE_2:='BRAMA ZAMKNIETA';
    else stan:=0; end_if;

    if (DI_5 or KEY_LEFT) and not DI_0 then var_silnik:=2;
LCD_LINE_2:='ZAMYKANIE';
    elsif (DI_7 or KEY_RIGHT) and not DI_1 then
var_silnik:=1; LCD_LINE_2:='OTWIERANIE';
    else var_silnik:=0;
    end_if;

 end_if;
```

Listing 5.5.6 Program główny - działanie trybu manualnego

Na listingu 5.5.7 widzimy wywołanie instancji bloku funkcjonalnego o nazwie „silnik”, ze zmienną wejściową „var_silnik”. Kolejne trzy linie warunkują działanie diod LED widocznych w modelu. Diody są odpowiednio włączane, wyłączane lub wywołana zostaje instancja bloku funkcjonalnego „MRUGANIE” o nazwie „mrug”.

```
silnik(silnik_input:=var_silnik);

if DI_0 then DO_6:=true; DO_7:=false;
elsif DI_1 then DO_7:=true; DO_6:=false;
else mrug(); DO_7:=false; end_if;
```

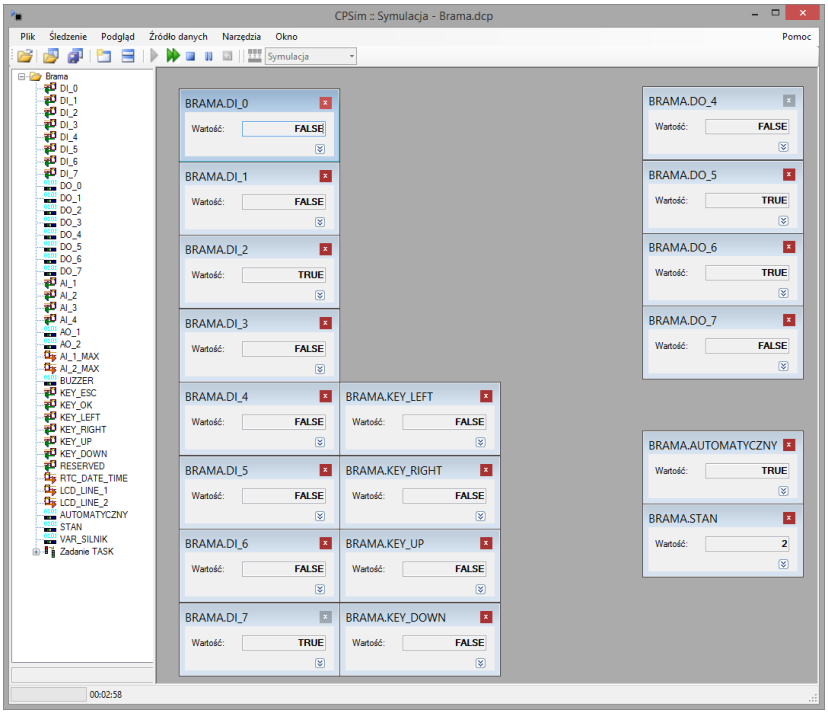
Listing 5.5.7 Program główny - załączanie diod LED

6. Badanie opracowanego stanowiska

W tym rozdziale ukazany został finalny rezultat badań nad modelem laboratoryjnym bramy opisanym w pracy. Pokazane zostało testowanie programu, wykonany model fizyczny oraz testowanie jego działania.

6.1. Testowanie oprogramowania przy pomocy symulacji

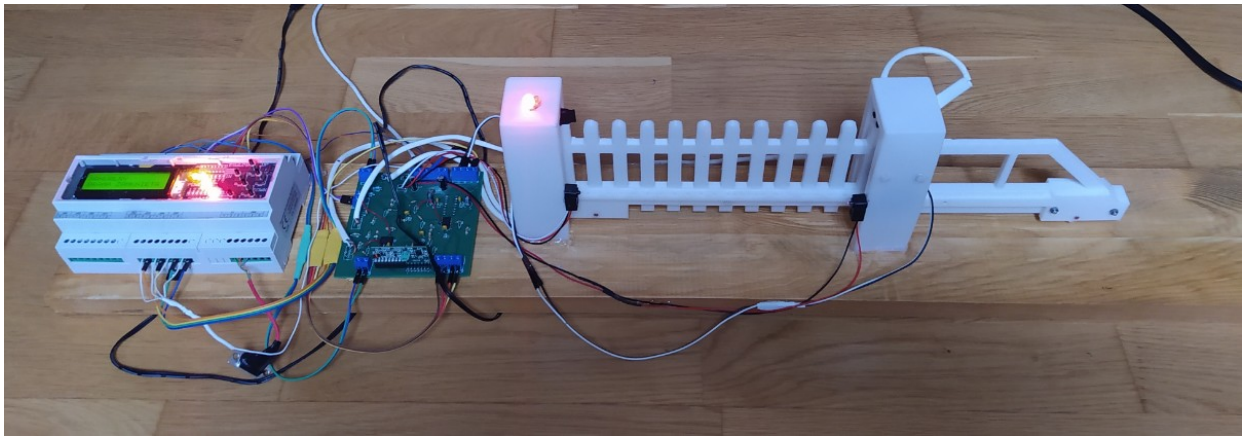
Pierwszym etapem było testowanie programu. Badanie przeprowadzono przy pomocy wbudowanego w pakiet CPDev symulatora CPSim. Przeprowadzona symulacja została w trybie offline, jeszcze przed wgraniem finalnego programu na sterownik. Jak można zauważyć na rysunku 6.1.1 po lewej stronie widoczne są zmienne wejściowe, których wartości musiały być zmieniane w odpowiedniej kolejności tak, aby odpowiadało to działaniu modelu rzeczywistego. Są to wszystkie cyfrowe zmienne wejściowe: DI_0, DI_1, DI_2, DI_3, DI_4, DI_5, DI_6, DI_7 oraz zmienne wejściowe sygnałów pochodzących z klawiatury wbudowanej w sterownik PLC: KEY_RIGHT, KEY_LEFT, KEY_UP, KEY_DOWN. Po prawej stronie widzimy natomiast zmienne, które pokazują stan wyjść tranzystorowych sterownika: DO_4, DO_5, DO_6, DO_7. Dodatkowo na symulacji wyświetlone zostały dwie zmienne „AUTOMATYCZNY”, aby mieć pogląd na to w jakim trybie działania jest aktualnie sterownik oraz zmienna „STAN”, aby mieć pogląd w jakim aktualnie stanie znajduje się automat stanowy programu głównego.



Rys. 6.1.1 Testowanie programu - symulacja CPSim

6.2. Przedstawienie opracowanego systemu

Model przedstawiony na rysunku 6.2.2 został wydrukowany za pomocą techniki druku 3d. Pozwoliło to jak najbardziej zbliżyć przewidywany efekt z wykonanym modelem rzeczywistym. Widzimy tam kolejno od lewej: Sterownik E-TRONIX SU1.7, płytkę PCB oraz model bramy. Widoczne okablowanie, świadczy o tym, że wszystkie konieczne połączenia pomiędzy tymi elementami zostały dokonane. Jedyną wymaganą zmianą było zaklejenie fragmentu bramy, który jest sczytywany poprzez transoptory szczelinowe za pomocą czarnej izolacji. Jest to spowodowane faktem, że czujniki nie działają przy białym kolorze plastiku z jakiego został wykonany model.

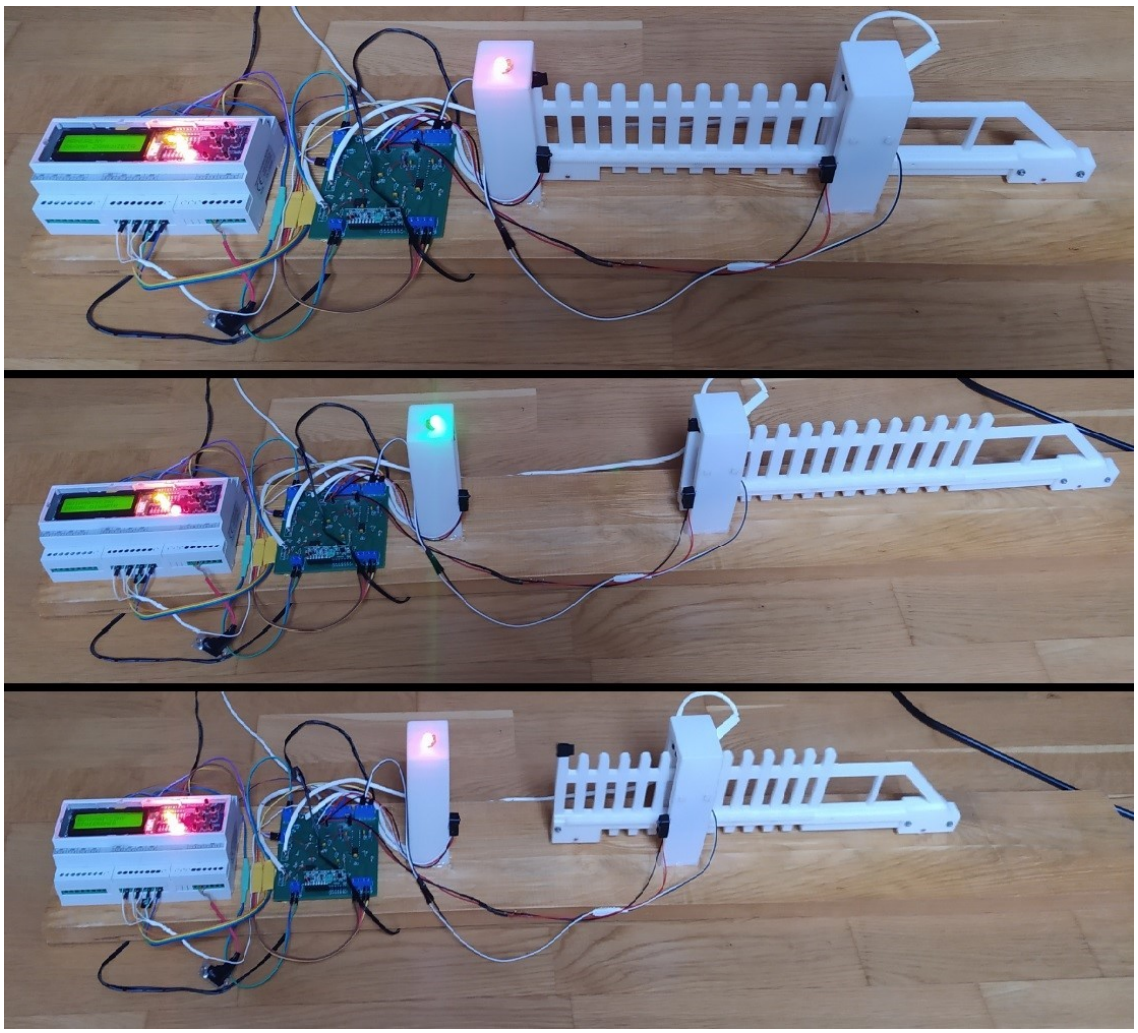


Rys. 6.2.1 Wykonany model rzeczywisty

6.3. Testowanie działania rzeczywistego modelu

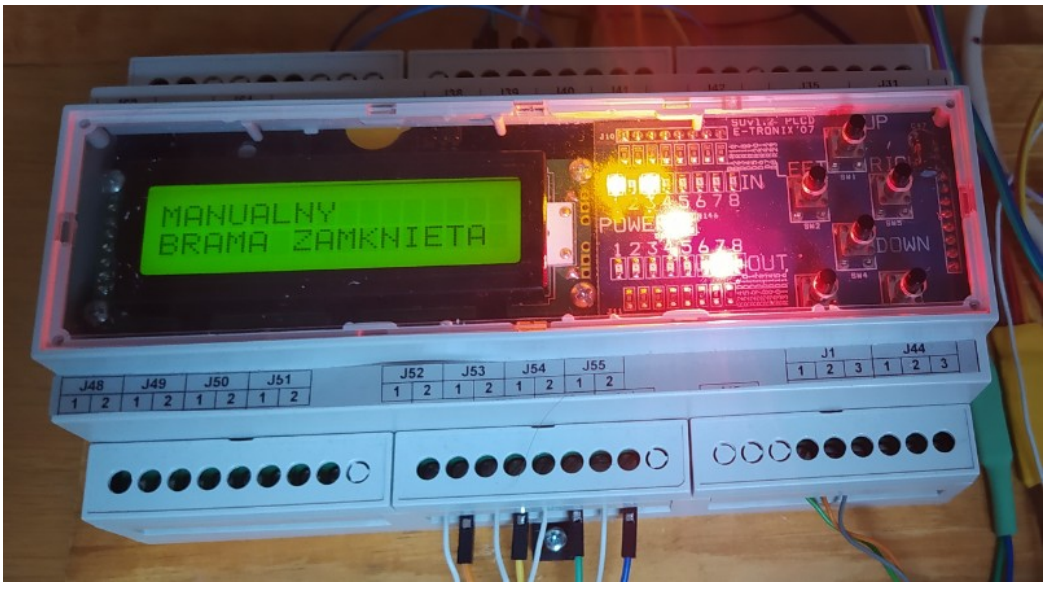
Wszystkie testy działania modelu, przebiegły prawidłowo. Działa sterowanie zarówno za pomocą pilota jak i przycisków na sterowniku. Sterownik odpowiednio wyświetla komunikaty na wyświetlaczu LCD. Nie zauważono, też żadnych błędów pomiędzy przechodzeniem między różnymi trybami pracy sterownika. Warto jednak, zauważyć że jest to prototypowy model a nie realna brama, więc duże znaczenie w niektórych niedociągnięciach pracy modelu ma jakość użytych komponentów. Sam sterownik jednak sprawuje się bardzo dobrze.

Na rysunku 6.3.1 zamieszczone zostały połączone trzy zdjęcia ukazujące sterownik podczas pracy. Są to kolejno od góry: brama zamknięta, brama otwarta oraz brama w trakcie zamykania.



Rys. 6.3.1 Model rzeczywisty – działanie

Dodatkowo na rysunku 6.3.2, ukazane zostało w jaki sposób działa wyświetlanie się komunikatów na wyświetlaczu sterownika.



Rys. 6.3.2 Model rzeczywisty – wyświetlacz

7. Podsumowanie i wnioski końcowe

Inspiracją do napisania pracy była chęć zapoznania się z innymi rozwiązaniami ze świata sterowników PLC. Jak dotąd rynek sterowników był zdominowany przez kilka wielkich firm, których rozwiązania pojawiały i nadal pojawiają się wszędzie. Obecnie powstaje coraz więcej mniejszych firm chcących wyjść naprzeciw wielkim korporacjom, oferując rozwiązania sprzętowe oraz oprogramowanie. Opracowanie systemu sterowania bramą było świetnym przykładem ukazania działania sterownika.

Najważniejszymi elementami pracy było napisanie programu sterującego w środowisku uruchomieniowym CPDev. Program główny dzielił się na dwie zasadnicze części: sterowanie automatyczne oraz manualne. Dodatkowym utrudnieniem było bezproblemowe zmienianie trybu pracy w dowolnym momencie działania bramy. Kolejnym kluczowym elementem pracy było zaprojektowanie oraz wykonanie modelu 3d, które pozwala na zobaczenie efektów pracy nie tylko za pomocą symulacji. Ważnym aspektem było też zaprojektowanie płytki PCB pośredniczącej pomiędzy sterownikiem, a użytymi czujnikami ze względu na różnice zakresu napięć na jakich działały.

W wyniku przeprowadzonych prac dokonano obserwacji niepożądanego zachowania sterownika w postaci wolniejszego reagowania na odczyt transoptorów szczelinowych działających jako krańcówki w momencie działania trybu automatycznego pracy, co nie powinno mieć miejsca w przypadku stałego cyklu pracy sterownika.

Kolejnymi wadami jest brak mozliwości skorzystania z dwóch szybkich wejść cyfrowych sterownika w środowisku CPDev. Są to wejścia podłączone do pinów przerwań mikroprocesora, jednak z poziomu CPDev'a nie mamy możliwości skonfigurowania ich, tak jak robi się to w przypadku programowania mikroprocesorów z rodziny ATmega.

Kolejną niedogodnością w pakiecie CPDev znalezioną podczas wykonywania projektu jest fakt, że zmienne wejściowe oraz wyjściowe w blokach funkcjonalnych pisanych w językach graficznych mogą być wyłącznie typu bool. Możliwość używania innego typu zmiennych występuje tylko w przypadku używania języków tekstowych. Drugim rozwiązaniem problemu jest używanie w bloku funkcjonalnym zmiennej globalnej, której wartość nadamy w programie głównym przed wywołaniem bloku funkcjonalnego.

Autor za własny wkład pracy uważa:

1) Napisanie programu systemu sterującego
2) Zaprojektowanie modelu bramy w oprogramowaniu Autodesk Fusion360

3) Zaprojektowanie płytki PCB w oprogramowaniu Autodesk EAGLE.
4) Wykonanie części fizycznej pracy: lutowanie elementów na płytce PCB, wykonanie połączeń oraz złożenie modelu
5) Wdrożenie do projektu sterownika, jego konfigurację oraz oprogramowanie go.

Załączniki

Płyta CD zawierająca:

1) Projekt wykonany w środowisku CPDev
2) Plik wizualizacji CPSim

Wykaz listingów

Spis tabel

Wykaz ilustracji

Literatura

[1] Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L.: Programowanie w języku ST sterownika SMC Lumel dla minisystemu rozproszonego, XIII Konferencja Automation, Pomiary Automatyka Robotyka, (CD) str. 606-614, 2/2009.

[2] Trybus L., Jamro M., Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B.: Uzupełnienia środowiska inżynierskiego CPDev dla programowania holenderskiego systemu sterowania statków Mega-Guard, Napędy i sterowanie 6/2012, s. 98-103.

[3] Norma PN-EN 61131-3:2013-10.

[4] Katedra Informatyki i Automatyki PRZ, Sterowniki_IEC61131-3. http://automatyka.kia.prz.edu.pl/attachments/article/13/Sterowniki_IEC61131-3.pdf Dostęp 15.01.2022

[5] Baza wiedzy – Pakiet inżynierski CPDev. https://www.e-tronix.eu/14,pakiet-cpdev-i-codesys.html Dostęp 19.01.2022

[6] Pakiet inżynierski CPDev - Instrukcja programowania, Rzeszow University of Technology, Department of Computer and Control Engineering

[7] Programowalny sterownik SU1.7 https://www.e-tronix.eu/46,sterownik-plc-programowalny-su-1-7.html Dostęp 10.01.2022

[8] ATMEGA1284P-AU. Atmel Corporation. https://www.farnell.com/datasheets/2048001.pdf Dostęp 15.01.2022

[9] LM358, LM258, LM158, LM2904 Dual Operational Amplifiers, Texas Instruments. https://botland.com.pl/index.php?controller=attachment&id_attachment=1973 Dostęp 23.01.2022

[10] Transmissive Optical Sensor with Phototransistor Output, VISHAY. https://botland.com.pl/index.php?controller=attachment&id_attachment=62 Dostęp 23.01.2022

[11] Czterokanałowy moduł radiowy 433 MHz + pilot. https://botland.com.pl/moduly-radiowe/16729-czterokanalowy-modul-radiowy-433-mhz-pilot-5904422326449.html Dostęp 23.01.2022

[12] Czujnik przerwania wiązki IR – LED. https://botland.com.pl/czujniki-ruchu/18689-czujnik-przerwania-wiazki-ir-led-3mm-0-50cm-5904422366476.html Dostęp 23.01.2022

[13] PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES, SGS-THOMSON MICROELECTRONICS.

https://botland.com.pl/index.php?controller=attachment&id_attachment=32
Dostęp 23.01.2022

[14] Programowanie PLC zgodnie z normą IEC61131 – standardy i środowisko programistyczne. ELEKTRONIKA PRAKTYCZNA 6/2014. https://ep.com.pl/files/10707.pdf Dostęp 25.01.2022

[15] Trybus L.: Automatyka i Sterowanie. http://automatyka.kia.prz.edu.pl/attachments/article/9/UkladyCzasowe.pdf Dostęp 15.01.2022

[16] Autodesk Fusion360. https://www.autodesk.pl/products/fusion-360/overview Dostęp 21.12.2021

[17] Autodesk EAGLE. https://www.autodesk.com/products/eagle/overview Dostęp 21.12.2021

[18] Jerzy Kasprzyk, Programowanie sterowników przemysłowych, WNT, Warszawa 2006.

[19] Sterownik programowalny E-TRONIX SU1.7, specyfikacja. https://www.e-tronix.eu/download/specyfikacja_SU_1.2.pdf Dostęp 10.01.2022

[20] Trybus L.: Automatyka i Sterowanie. http://materialy.prz-rzeszow.pl/pracownik/pliki/24/W3%20-%20Uk%C5%82ady%20czasowe%20AiS%202017.pdf Dostęp 15.01.2022

[21] Trybus L.: Automatyka i Sterowanie. http://materialy.prz-rzeszow.pl/pracownik/pliki/24/W2%20-%20Uk%C5%82ady%20sekwencyjne%20AiS%202016.pdf Dostęp 15.01.2022

[22] Serwo SG-90 - micro – 180. https://botland.com.pl/serwa-typu-micro/13128-serwo-sg-90-micro-180-5904422350338.html?cd=1564049911&gclid=EAIaIQobChMI7MjkiJTd9QIVjrmyCh1GAQzQEAQYASABEgICEfD_BwE&sskey=0cfe621530884d5c9610b833a5ddfccf Dostęp 23.01.2022

Sygnatura:

POLITECHNIKA RZESZOWSKA im. I. Łukasiewicza Rzeszów, 2022
Wydział Elektrotechniki i Informatyki

STRESZCZENIE PRACY DYPLOMOWEJ INŻYNIERSKIEJ

Laboratoryjny model bramy

Autor: Hubert Dzieciuch, nr albumu: EA-DI-160833
Opiekun: dr inż. Dariusz Rzońca
Słowa kluczowe: sterownik PLC, norma IEC 61131-3, języki tekstowe, stanowisko laboratoryjne, pakiet inżynierski CPDev

Celem pracy było przetestowanie oraz przedstawienie możliwości sterownika E-TRONIX SU1.7. Zaprojektowane oraz wykonane zostało stanowisko laboratoryjne pozwalające na przetestowanie sterownika w systemie sterowania bramą automatyczną. w pracy zamieszczono informacje o środowisku programistycznym Control Program Developer, sterowniku oraz sposobie jego programowania. Opisany został projekt bramy wykonany w oprogramowaniu Fusion360, projekt płytki PCB wykonany w oprogramowaniu EAGLE oraz utworzony został model rzeczywisty. Pokazano również weryfikację programu za pomocą symulacji przy użyciu narzędzia CPSim. Zaprezentowano również wnioski wyciągnięte na podstawie badania całego systemu, tj. rzeczywistego modelu wraz z pełnym oprogramowaniem.

RZESZOW UNIVERSITY OF TECHNOLOGY Rzeszow, 2022
Faculty of Electrical and Computer Engineering

DIPLOMA THESIS (BS) **ABSTRACT**

Laboratory model of a gate

Author: Hubert Dzieciuch, code: EA-DI -160833
Supervisor: Dariusz Rzońca, PhD, Eng.
Key words: PLC Controler, IEC 61131-3 standard, textual languages, laboratory station, CPDev engineering package

The aim of the study was to test and present the capabilities of the E-TRONIX SU1.7 controller. A laboratory station was designed and constructed to test the controller in an automatic gate control system. The thesis contains information about the Control Program Developer environment, the controller and the way of programming it. The design of the gate was made in Fusion360 software, the PCB design was made in EAGLE software and the real model were built. Verification of the program by simulation using the CPSim tool is also shown. The conclusions drawn from the study of the whole system, i.e. the real model with full software, are also presented.