Mikrosystemy w motoryzacji – laboratorium

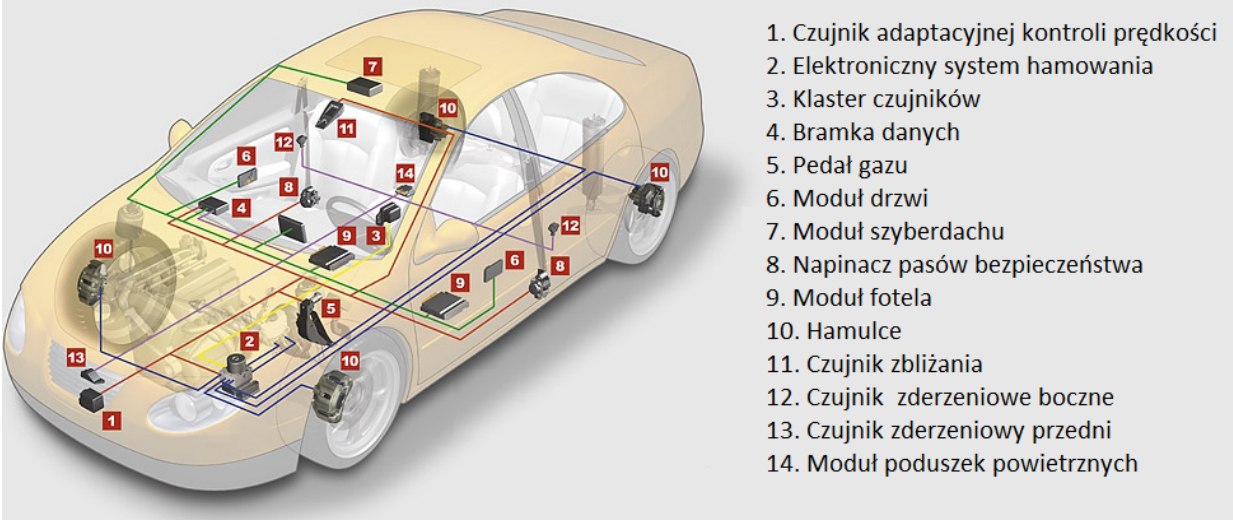
Ćwiczenie nr 3
Komunikacja CAN – sterowanie oświetleniem

Ćwiczenie nr 4
Komunikacja CAN – sterowanie lusterkiem

opracował: dr inż. Paweł Knapkiewicz

Wprowadzenie:

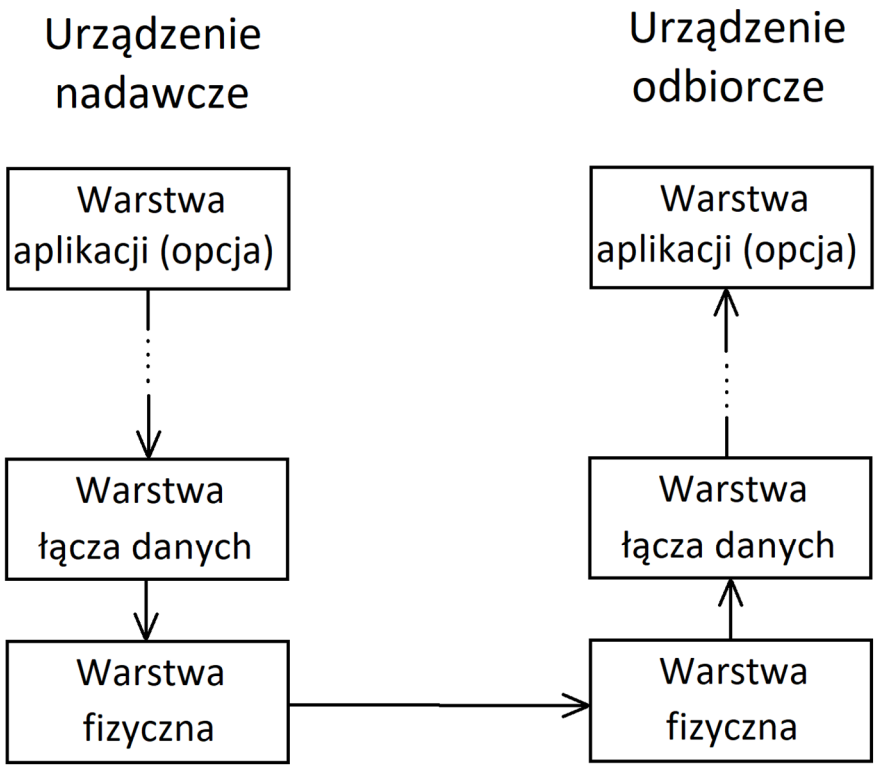
W latach 80 XX wieku firmy motoryzacyjne zaczęły montować urządzenia elektroniczne oparte na mikrokontrolerach. Dedykowane układy sterowały pracą zarówno silnika jak i mniej istotnej klimatyzacji. Ten kolejny etap w ewolucji samochodów pozwalał na dalszą optymalizację parametrów wpływających na atrakcyjność użytkowania. Dzięki współpracy z czujnikami takimi jak sonda lambda [1] zmniejszało się zużycie paliwa, przy jednoczesnym wzroście mocy silników, możliwe stało się wprowadzenie poduszek powietrznych, systemu kontroli trakcji, czy zaawansowanych systemów komfortu. jazdy. Do coraz to bardziej zaawansowanych zastosowań koniecznym stało się zapewnienie komunikacji między modułami (Rys. 1). Producentom zależało na niskokosztowym, niezawodnym i zapewniającym dostateczną prędkość transmisji rozwiązaniu. W 1981r. firma Bosch wyszła naprzeciw tym oczekiwaniom opracowując interfejs komunikacyjny CAN (ang.Controller Area Network) [2].



Rysunek 1 - Przykładowy schemat połączenia sieci CAN[3]

Nowa, szeregowa architektura szybko zyskiwała popularność także w innych niż auta osobowe zastosowaniach jak samochody ciężarowe, autobusy, samoloty, statki, maszyny rolnicze, czy nawet w systemach automatyki budynkowej, systemach rozproszonego sterowania w zakładach przemysłowych, sieciach sensorowych i wielu innych [4]. Za sukcesem sieci opartych na

standardzie CAN stoi ich prosta aczkolwiek bardzo przemyślana konstrukcja. Rozpatrując występujący w nim przekaz informacji na podstawie modelu OSI ISO [5] można wyróżnić warstwę fizyczną, warstwę łącza danych oraz warstwę aplikacji. W komunikacji CAN pozostałe warstwy nie występują (Rys. 2).

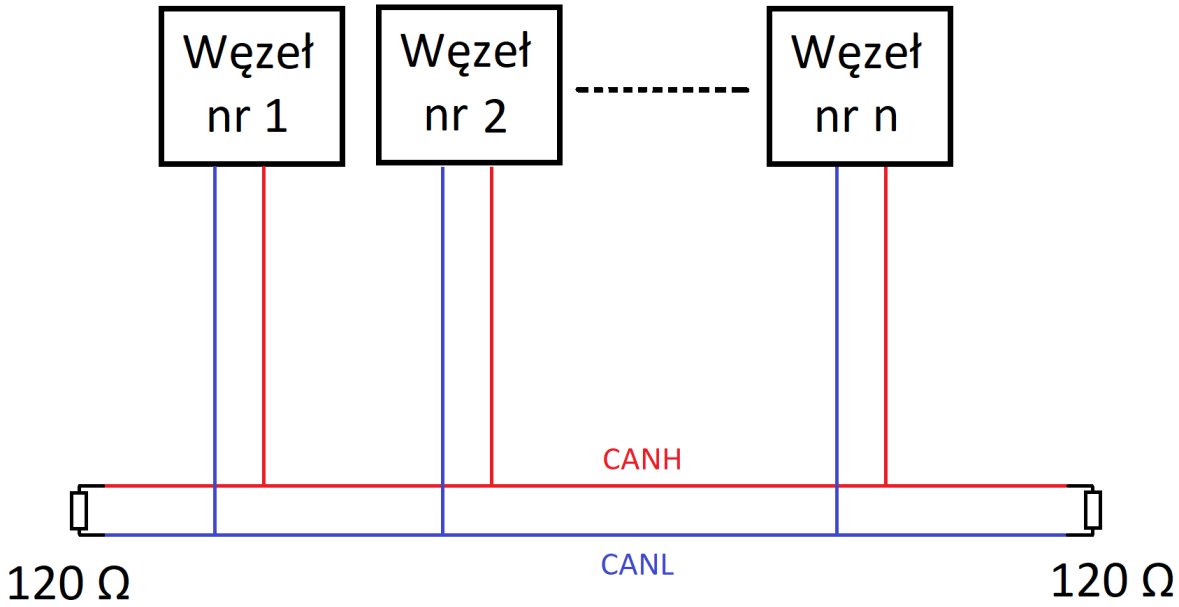


Rysunek 2 - Schematyczny przepływ między warstwami modelu OSI -opracowanie własne.

Warstwa łącza danych formatuje nadawany sygnał do postaci ramki CAN, decyduje ona także o tym czy to jest odpowiedni moment na wysłanie ramki, sprawdzając czy taka transmisja nie jest już dokonywana. Rozpoczynając proces nadawania sprawdzana jest także poprawność przesyłania danych. Warstwa fizyczna dokonuje przekształcenia ramki na odpowiednie sygnały elektryczne reprezentujące wartości logiczne. W odbiorniku następuje proces który można opisać jako odwrotny do nadawania: impulsy elektryczne zostają przekształcone do postaci ramki CAN, a po sprawdzeniu jej poprawności komenda może być dalej procedowana.

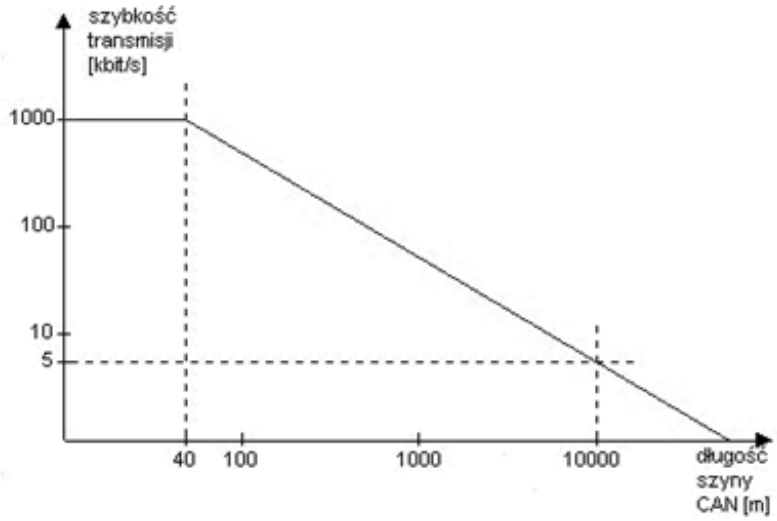
Warstwa fizyczna.

Fizyczne połączenie następuje przy użyciu magistrali, skrętki dwóch przewodów, zakończonych rezystorami terminującymi 120 Om, które zapobiegają odbiciom sygnałów [6]. Podłączone do magistrali węzły odbierają w czasie rzeczywistym ten sam sygnał nadawany przez jeden z nich (Rys. 3).



Rysunek 3 - Schematyczne podłączenie węzłów do szyny CAN- opracowanie własne.

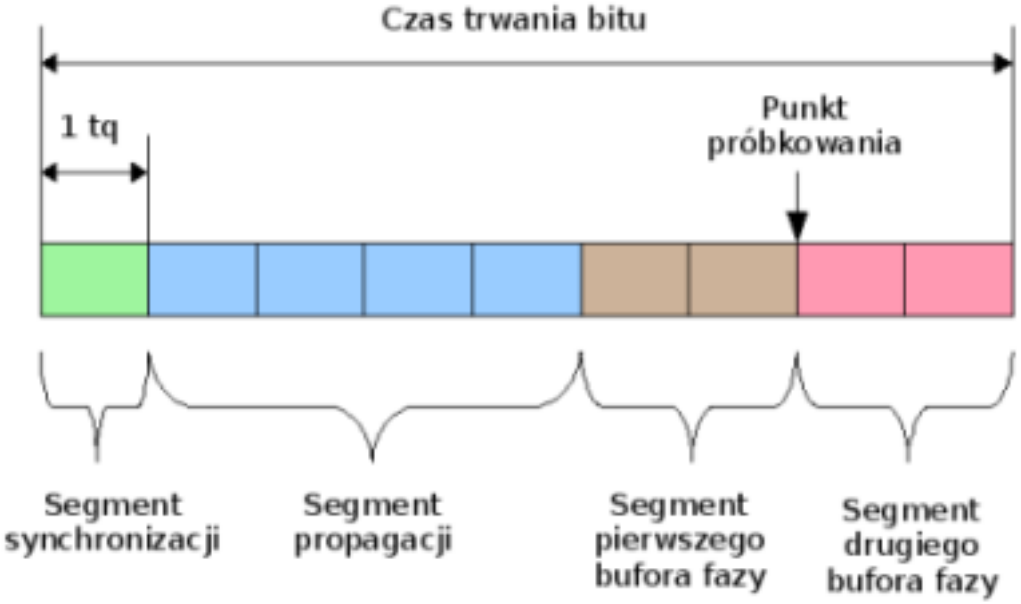
Maksymalna prędkość nadawania wynosi 1 Mbit/s, zgodnie z zaleceniami producenta jest ona ograniczona ze względu na długość magistrali. Odwrotnie proporcjonalną zależność zalecanej szybkości transmisji do długości szyny CAN przedstawia wykres na rysunku 4.



Rysunek 4 - Zależność między szybkością transmisji, a długością magistrali.[20]

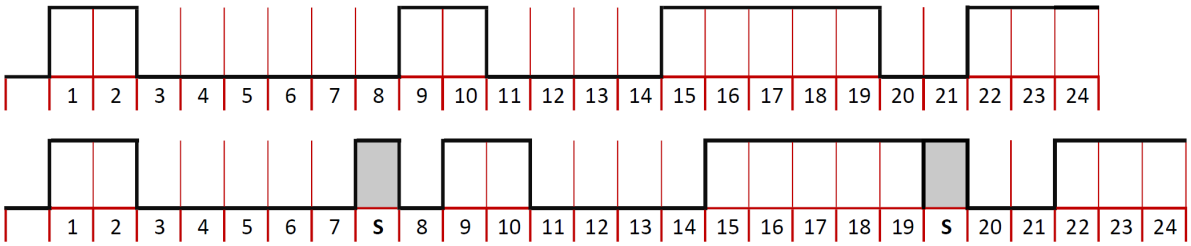
Przesył wiadomości może być rozpoczęty w dowolnym momencie. W celu zapewnienia prawidłowego działania każdy węzeł w danej magistrali musi mieć ustawione takie same taktowanie. W innym przypadku urządzenia odbiorcze nie będzie w stanie odczytać wszystkich podanych bitów. Próbkowanie, a więc odczytanie jego wartości odbywa się w ściśle określonym momencie. Czas przeznaczony na wysłanie jednego bitu jest podzielony na mniejsze odcinki, które według zamysłu twórców standardu przeznaczone są kolejno na: synchronizację, czas propagacji sygnału, pierwszy bufor czasowy i drugi bufor czasowy. Próbkowanie odbywa się pomiędzy pierwszym a drugim buforem. Ma to zapewnić najoptymalniejszy, redukujący możliwość popełnienia błędu moment na odczytanie informacji.

Na potrzeby danej aplikacji czas trwania poszczególnych segmentów przewidzianych przez standard może być programowany. Na rysunku 5 przedstawiono przykładową konfigurację.



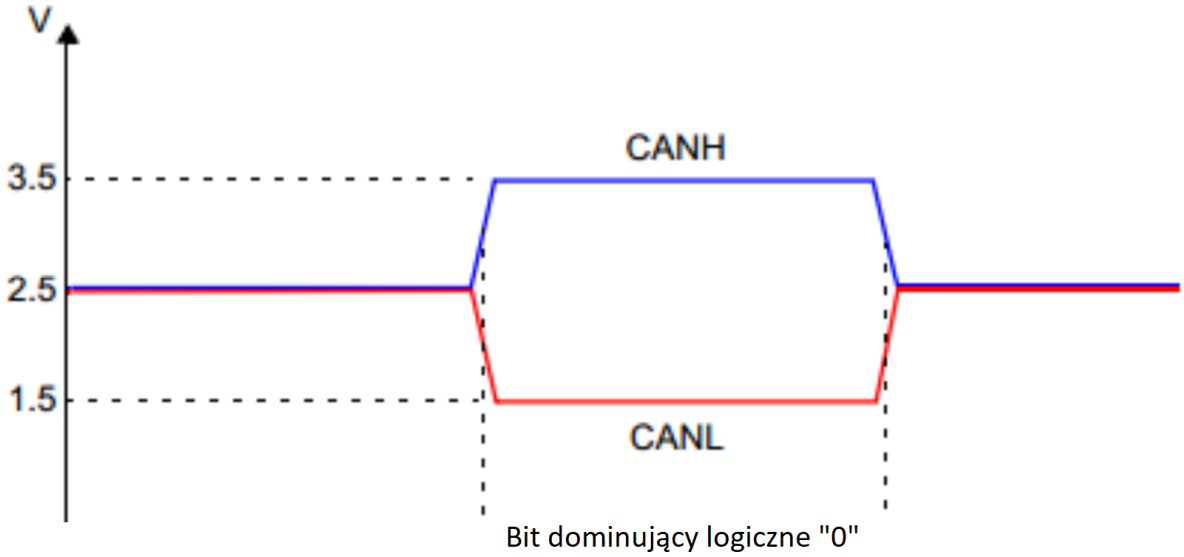
Rysunek 1 - Podział czasu trwania nadawania bitu na segmentu.[7]

Dodatkowym mechanizmem zapobiegania błędom powstałym przy złej synchronizacji odczytu z nadawaniem jest Bit stuffing (Rys. 6). Nieuniknione różnice w częstotliwości taktowania zegarów nadajnika i odbiornika mogą prowadzić do desynchronizacji. W chwili, gdy wartości logiczne wysyłane przez nadajnik zmieniają swoją wartość można pomocą dodatkowego próbkowania ustalić kiedy następuje początek przesyłu danych. W przypadku gdy nadawana wiadomość niesie ze sobą jeden po drugim tą samą wartość logiczną odbiornik nie jest w stanie rozpoznać kiedy zaczął się bit kolejny, a kiedy zakończono nadawanie poprzedniego. Mechanizm bit stuffing w przypadku wystąpienia pięciu kolejnych bitów o tej samej wartości wstawia jeden bit o wartości przeciwnej [8].
Komplikuje to odczyt oraz nadawanie komunikatów CAN, ale dzięki zastosowaniu tego mechanizmu wzrasta niezawodność przesyłu danych bez konieczności doprowadzenia dodatkowego przewodu z sygnałem zegarowym.



Rysunek 6 - Przykład wiadomości nadanej z i bez użycia mechanizmu Bit stuffing - opracowanie własne.

W zamyśle projektantów eliminujący zakłócenia elektromagnetyczne przewód w postaci skrętki przesyła sygnał różnicowy. Domyślną wartością napięcia zarówno na przewodzie CANH (ang. High) jak i CANL (ang. Low) jest potencjał 2,5V względem masy. Stan ten określany jest mianem recesywnego, oznacza on również przesył wartości logicznej „1”. Stanowi dominującemu, a więc logicznemu “0” odpowiada wartość napięcia 3,5V na przewodzie CANH i 1,5V na CANL [6] (Rys. 7).



Rysunek 7 - Graficzne przedstawienie nadania bitu dominującego "0" [4]

Dane te dotyczą najpowszechniej stosowanej specyfikacji CAN 2.0 High speed, Oprócz tego istnieje jeszcze wersja specyfikacji low-speed, zwanej też fault tolerant (tłum. odpornej na błędy). Odporniejsza na zakłócenia wersja oferuje przesył z prędkością do 125 kbit/s co w niektórych zastosowaniach może mieć znaczenie drugorzędne względem korzyści jak np. odporność na pojawienie się dużych napięć na magistrali. W tym przypadku urządzenia pracujące w standardzie low-speed powinny wytrzymać napięcia od -27V do 40V mogące się pojawiać na liniach CANL i CANH. Odbywa się to jednak kosztem wyższych napięć sygnałowych i zwiększonego poboru prądu [7, 9].

Warstwa łącza danych

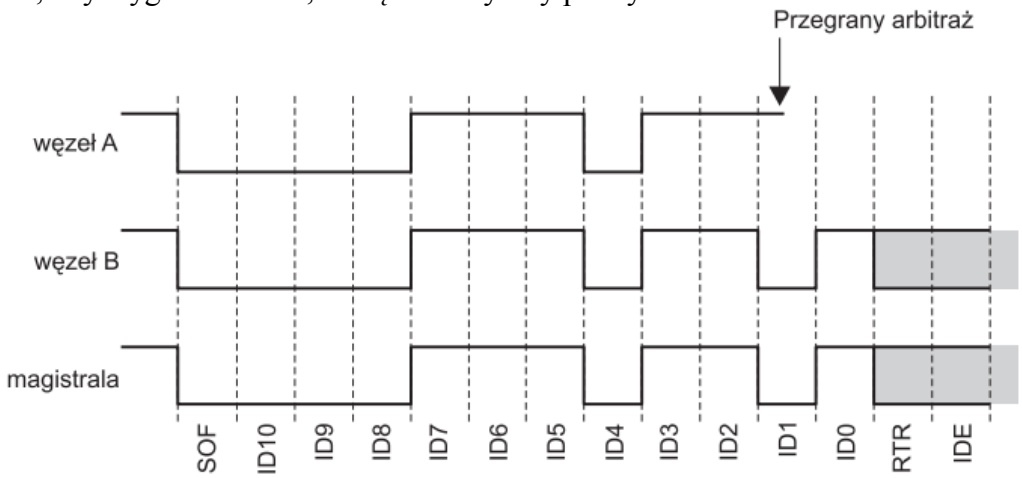
Opierając się na opisanych zasadach realizacji warstwy fizycznej jednocześnie rozpatrując standard na wyższym poziomie abstrakcji jego twórcy ustalili zasady jakimi mają się cechować komunikaty CAN. W celu zrealizowania stawianych przed nowym interfejsem komunikacyjnym zadań określone zostały 4 rodzaje ramek komunikacyjnych:

- Data Frame - podstawowa ramka składająca się z ustalonego komunikatu dodatkowo mogąca realizować przesył danych,
- Remote Frame - typ ramki mający stanowić odpytanie innego modułu. Ma ona wywoływać reakcje w postaci przekazania z powrotem określonego. Na przykład wartości temperatury. dzięki czemu przekazanie tej wartości odbywa się „na żądanie" tylko wtedy, gdy ona będzie potrzebna do pracy innego modułu,
- Error Frame - ramka informująca o błędzie, nieprawidłowym odczycie,
- Overload frame - ramka informująca o zbyt dużym obciążeniu odbiorcy i niemożności wykonania polecenia. Obecnie rzadko występująca ze względu na dużą szybkość wykonywania poleceń przez moduł – odbiorcę oraz zaimplementowaną w nim opcje kolejkowania [10].

Wspomniana specyfikacja CAN 2.0 rozróżnia sytuację, w której wykorzystywana jest ramka posiadająca 11 bitowy unikalny adres (opisany przez standard CAN 2.0a), bądź przez 29 bitowy adres. Dla dobrego zrozumienia opisywanego tematu należałoby się tu jednak zatrzymać i wyjaśnić czym jest owy adres, bądź zamiennie: komunikat czy tez message ID. Twórcy

standardu zakładają, że w tym przypadku adres nie ma odpowiadać węzłowi sieci, a reprezentować funkcje jak na przykład włączenie wycieraczek. Jeden moduł miałby reagować na wiele różnych komunikatów. Tak też wygląda implementacja w rzeczywistych, komercyjnych zastosowaniach. Na przykładzie pozycji fotela: jeden komunikat mógłby określać jego przesunięcie w danej płaszczyźnie. Tu już w zależności od implementacji konkretnego producenta pole data mogłoby określać np. pozycję fotela do której moduł-aktuator miałby ją przesunąć. Wydaje się to komplikować stosunkowo proste zadanie, ale jednocześnie wprowadza ogromną elastyczność umożliwiając wprowadzenie nowych funkcjonalności jak pamięć pozycji foteli, czy po zastosowaniu dodatkowej warstwy aplikacji stworzenie oryginalnych interfejsów.

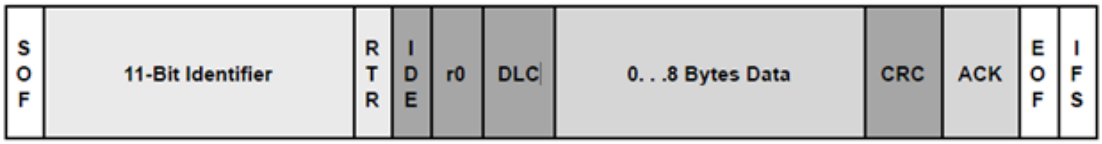
Standard CAN 2.0a o 11 bitowej długości adresu pozwala na stworzenie teoretycznie 2048 (2^{11}) unikalnych wiadomości CAN. Z doświadczeń własnych autora nabytych przy realizacji pracy wynika, że ta liczba jest trudna do osiągnięcia w rzeczywistych zastosowaniach, wynika to z potrzeby uwzględnienia mechanizmu arbitrażu. Inżynierowie firmy Bosch projektując działanie sieci CAN musieli uwzględnić sytuację, w której następuje jednoczesny przesył danych z dwóch modułów. Można było to rozwiązać tworząc model komunikacji Master-Slave, w którym węzeł nadrzędny decydowałby, któremu modułowi w danej chwili "udzielić głos". Problem pojawia się, gdy moduł chciałby poinformować o wydarzeniu które powinno być jak najszybciej obsłużone, jak np. czujnik kolizji informujący moduł poduszki powietrznej o zderzeniu pojazdu. Zbędna zwłoka mogłaby decydować nawet o czyimś życiu. Dodatkową wadą tego rozwiązanie jest, że w przypadku uszkodzenie jednostki nadrzędnej oznaczałoby także sparaliżowanie działania całej magistrali. Zdecydowano się na użycie architektury multi-master, w której sama wiadomość decydowałaby o jej ważności. Moduł oczekuje na moment, w którym inne urządzenie przestanie nadawać swój komunikat, gdy tak się stanie może zostać podjęta próba wysłania wiadomości, jednocześnie odbywa się monitorowanie rzeczywistego stanu linii. W przypadku, gdy dwa urządzenia równocześnie zaczną nadawać różne komunikaty, jeden z nich wykryje niezgodność własnego komunikatu ze stanem linii. Nastąpi to w module, który wyślę logiczną wartość "1" (Bit recesywny) podczas, gdy drugie urządzenie wyślę wartość "0" czyli bit dominujący. Moduł, który "wygrał" ten arbitraż kontynuuje nadawanie, podczas gdy drugi musi poczekać na swoją kolej (Rys. 8). W ten sposób sama wiadomość jest także informacją o jej priorytecie. Im wiadomość reprezentuje niższą wartość tym ma większe szanse, aby wygrać arbitraż, a więc ma wyższy priorytet.



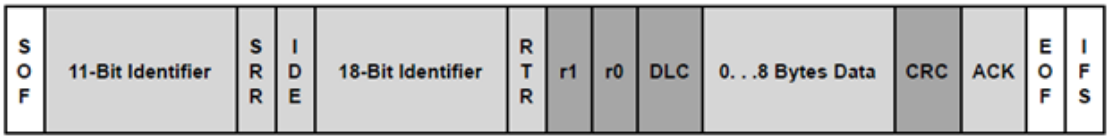
Rysunek 8 - Arbitraż dostępu do medium.[11]

Przy uwzględnieniu tej priorytyzacji mogą wystąpić wspomniane wcześniej problemy z wykorzystaniem wszystkich dostępnych adresów. Przypisanie wartości binarnych do danego komunikatu jest stosunkowo proste, gdy z góry znane są wszystkie wykorzystywane komendy. W przypadku gdy tak nie jest, warto "zarezerwować" część adresów pod przyszłe użycie. Dla bardziej skomplikowanych, mogących być rozwijanymi w przyszłości implementacji, można wykorzystać standard CAN 2.0b rozszerzający ramkę CAN o kolejnych 18 bitów adresu. Poniżej porównanie struktury obu ramek wraz z oznaczeniem bitów (Rys. 9,10).

- SOF (*Start Of Frame*) – początek ramki,
- RTR (*Remote Transmission Request Bit*) – rodzaj ramki,
- IDE (*Identifier Extension*) – format ramki danych (podstawowy lub rozszerzony),
- r0 – zarezerwowany bit,
- DLC (*Data Length Code*) – liczba wykorzystywanych bajtów danych,
- CRC (*Cyclic Redundancy Check)* – suma kontrolna,
- ACK (Acknowledgement) – potwierdzenie wysłania/odebrania danych,
- EOF (*End of Frame*) – zakończenie ramki,
- IFS (*Intermission*) – przerwa przed następną ramką.



Rysunek 9 - Struktura ramki danych (format ramki standardowej – CAN 2.0A)[20]



Rysunek 10 - Struktura ramki danych (format ramki rozszerzonej – CAN 2.0B)[4]

Zarówno format ramki rozszerzonej jak i ramki standardowej pozwala na przesłanie 8 bajtów informacji. Umożliwia to przekazanie bardzo precyzyjnych danych. Przykładowo w przypadku czujników pole data można wykorzystać do przesłania interesujących nas informacji z ogromną (2^{64}) rozdzielczością. Można także przesłać precyzyjne ustawienia wysokości świecenia świateł mijania. Oczywiście w praktycznych rozwiązaniach nie trzeba wykorzystywać wszystkich 8 bajtów. standard pozwala przesłać także sam komunikat bez pola data.
Oprócz opisywanego standardu istnieją także inne rozwiązania. W przypadku, gdy wyśrubowane zalecenia dotyczące kontroli błędów i ich zapobiegania, nie są tak istotne można wykorzystać sieć opartą na strukturze LIN (Local Interconnect network). Konkurencyjne względem interfejsu CAN Rozwiązanie oparte jest na strukturze Master-slave, wymaga ono do działania tylko jednego przewodu, a obsługiwać go mogą nie wyspecjalizowane mikrokontrolery z obsługą UART, a więc system ten jest o wiele tańszy w implementacji [12].

Warstwa aplikacji

Dodatkową opcjonalną warstwą którą mogą wprowadzić w swoich produktach producenci aut jest warstwa aplikacji. Należy przez to rozumieć warstwę najbliższą użytkownikowi, mogącą pełni rolę pośrednika między aplikacjami nie mieszczącymi się w ramach modelu OSI (jak np. interfejs użytkownika na ekranie dotykowym), a warstwami niższymi [http://cisco.sadzer.pl/model-osi/]. W tej warstwie powstało wiele protokołów ułatwiających wykorzystanie zmodyfikowanej sieci CAN w specyficznych aplikacjach jak np. standard

NMEA 2000. Rozbudowana o dodatkowe przewody zasilające magistrala, działając analogicznie do pierwotnego rozwiązania zapewnia komunikacje urządzeń jednostkach pływających [13]. W tabeli 1 przedstawiono protokoły wykorzystujące założenia sieci CAN i obszary ich wykorzystania.

Tabela 1. Obszary wykorzystujące założenia sieci CAN

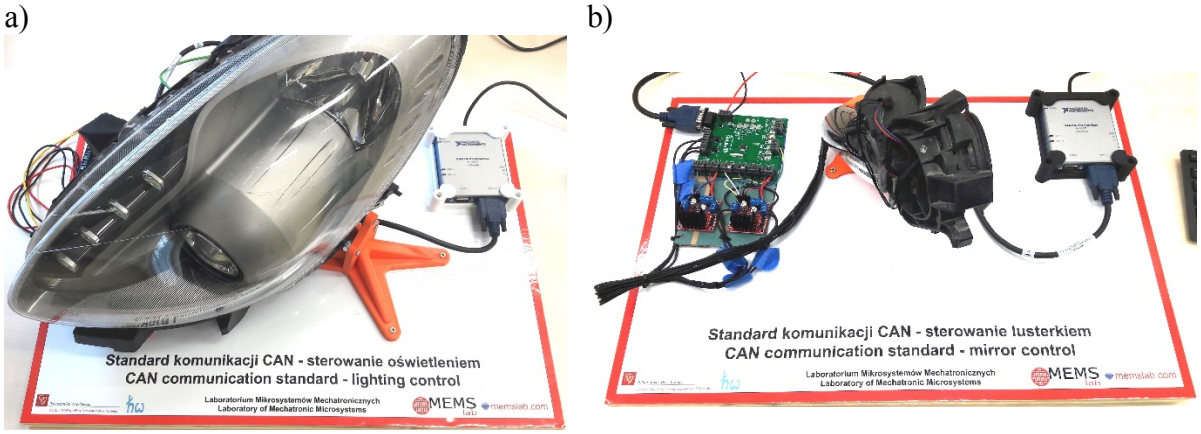
Nazwa protokołu	Główne obszary wykorzystania
NMEA 2000	Statki
J1939	Pojazdy kołowe
DeviceNet	Zastosowanie przemysłowe np. fabryki
CANopen	Systemy wbudowane
ISOBUS	Maszyny rolnicze
MILCAN	Zastosowania wojskowe

Sam użytkownik nie musi być świadom istnienia i działania przytoczonych warstw, ale uzyskana dzięki nim elastyczność, może pozwolić na bezpośrednie sterowanie komponentami auta. Dodawanie nowych elementów zwiększających bezpieczeństwo jak: systemy reagujące na oblodzenie jezdni, zwiększających komfort użytkowania jak masażery w fotelach, czy też będące wydawałoby się zbędnym gadżetem podświetlenie wnętrza na różne kolory, może stać się istotną przewagą danego producenta nad konkurencją. Wykorzystanie interfejsu komunikacyjne CAN pozwala dodatkowo na implementacje wygodnego dla użytkownika innowacyjnego sterowania na przykład poprzez dołączony do samochodu tablet czy sterowanie gestami.
Zaprezentowana w latach 80-tych idea sieci łączącej podzespoły samochodu wykorzystywana jest obecnie na bardzo szeroką skalę. Każde załączenie świateł mijania czy świateł drogowych może być sterowane poprzez komunikaty. Odpowiednio określając przekaz i reakcje modułu na niego można wykorzystać komunikaty CAN do wskazania które lampy mają być ściemnione, komunikatami można nawet włączać pojedyncze diody LED jeżeli tylko konstruktorzy uznają to za przydatne. Rozbudowana komunikacja między zainstalowanymi w aucie urządzeniami jest niezbędna do stworzenia zaawansowanego oświetlenia, jak wspomniane wcześniej, automatycznie reagujące na inne pojazdy reflektory przednie.

Cel i zakres ćwiczenia:

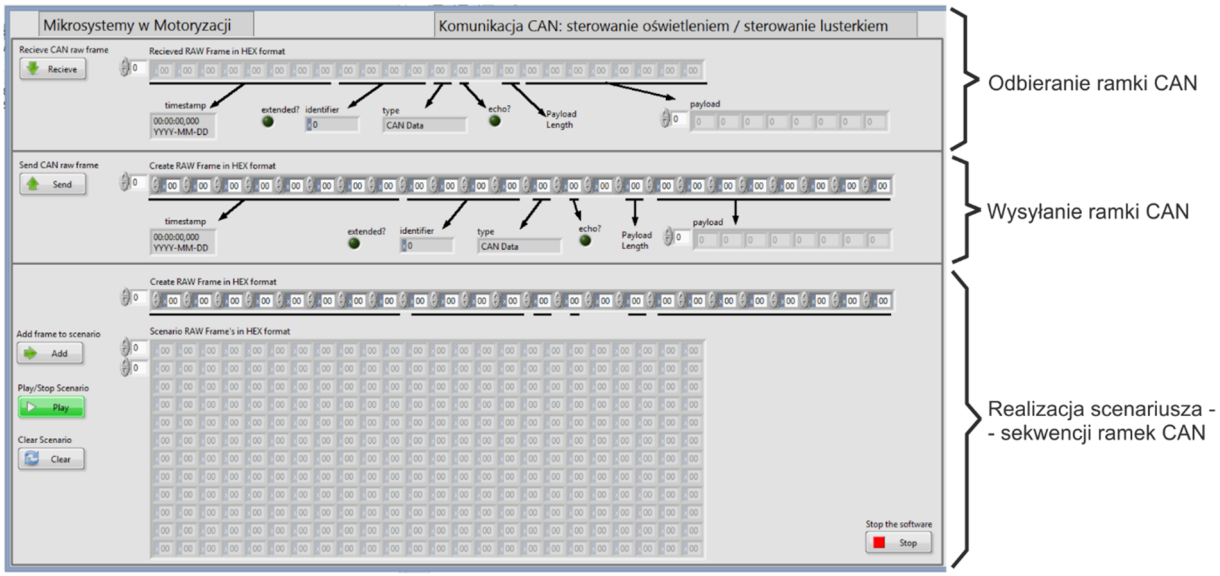
Stanowiska pomiarowe składają się z elementu badanego (lampy: AlfaRomeo Julietta / lusterka: Peugeot 407), połączonego z płytką PCB – nodem/węzłem CAN. Węzły CAN połączone są z kartą NI USB-8502, która stanowi interfejs komunikacyjny pomiędzy komputerem klasy PCB z oprogramowaniem LabView (Rys. 11).

a) b)



Rys. 11. Makieta edukacyjna / stanowisko pomiarowe: a) z lampą, b) z lusterkiem.

Panel użytkownika (Rys. 12), składa się z trzech bloków: Odbierania ramki CAN / Wysyłania ramki CAN / Przygotowania scenariusza.



Rys. 12. Panel użytkownika.

Celem ćwiczenia jest przyswojenie zasad komunikacji CAN, zrozumienie układu ramki danych transmisji CAN, i na podstawie nabytego doświadczenia zrealizowanie wybranego scenariusza sterowania.

Zadanie 1: Odebranie i rozszyfrowanie ramki danych CAN

Po odebraniu ramki (przycisk Recieve „Recieve CAN raw frame"), wyświetlona zostanie „surowa" ramka danych w notacji szestnastkowej (HEX) (Rys. 13).

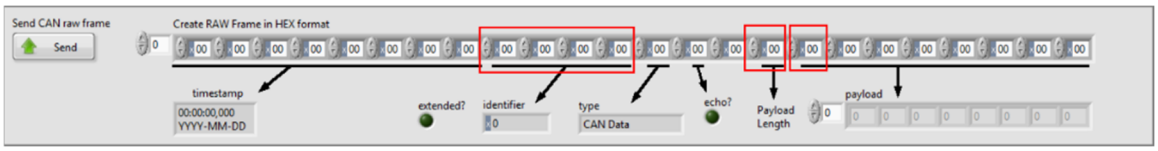
Rys. 13. Panel użytkownika – fragment odpowiadający za odbieranie ramki danych CAN.

Obierz wielokrotnie ramkę i wynotuj wartości stałe lub zmiany w ramce a szczególnie:

- zmiany dla „timestamp" (na których bajtach zmieniają się wartości, o ile zmieniają się wartości),
- wartość „Payload Lenght",
- zmiany dla „Payload" (na których bajtach zmieniają się wartości, o ile zmieniają się wartości). W wyodrębnionej części „payload" wartości przedstawione są w notacji dziesiętnej.

Zadanie 2: Wysyłanie ramki CAN

Należy zestawić wartości w ramce danych CAN i przypisać wartość ramki do funkcji lampy lub lusterka (Rys. 14).



Rys. 14. Panel użytkownika – fragment odpowiadający za wysyłanie ramki danych CAN.

Przygotowanie ramki danych CAN dla „Payload Lenght" = x00:

- przepisz wartości „indentifier" z ramki odebranej,
- ustaw wartość „Payload Lenght" = x00 (dla tego ustawienia wartości w części „payload" nie są istotne),
- wyślij kolejno ramki zmieniając baj '0' wartości „indentifier" w zakresie od x20 do x2F,
- zanotuj, jaka akcja lampy lub lusterka została wywołana przez daną ramkę CAN.

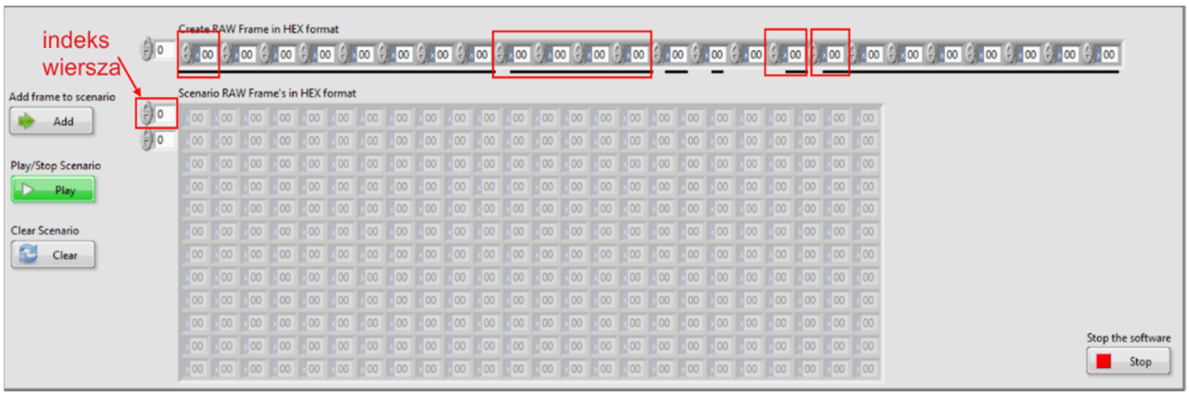
Przygotowanie ramki danych CAN dla „Payload Lenght" = x01:

- przepisz wartości „indentifier" z ramki odebranej,
- ustaw wartość „Payload Lenght" = x01 (dla tego ustawienia w części „payload" ważny jest – zostanie odczytany, bajt '0'),
- ustaw wartość bajtu '0' „payload" = x64 (100 DEC),
- wyślij kolejno ramki zmieniając baj '0' wartości „indentifier" w zakresie od x20 do x2F,
- zanotuj, jaka akcja lampy lub lusterka została wywołana przez daną ramkę CAN,
- ustaw wartość bajtu '0' „payload" = x00 i wyślij kolejno tylko te ramki (wartości „indentyfier"), które wcześniej wywołały jakąś akcję,

- odpowiedz, co powoduje zmiana wartości bajtu ‘0’ „payload” z x64 na x00 przy stałej wartości „indentyfier”?
- ustaw wartość bajtu ‘0’ „payload” = na dowolna z zakresu x01 do x63 i wyślij kolejno tylko te ramki (wartości „indentyfier”), które wcześniej wywołały jakąś akcję; znajdź komendy, w których zmiana wartości bajtu ‘0’ „payload” ma znaczenie – jakie znaczenie?

Zadanie 3: Przygotowanie scenariusza

Wykorzystując znajomość ramek danych i przypisanych im funkcji lampy lub lusterka, przygotuj scenariusz „wizualny” / „ruchowo-wizualny”. Do dyspozycji jest tablica, do której należy wpisać wybrane funkcje; ramki danych wysyłane będą w pętli (Rys. 15).



Rys. 15. Panel użytkownika – fragment odpowiadający za wysyłanie ramki danych CAN.

Przygotowanie scenariusza:

- uzupełnij wartości w ramce danych CAN „Create RAW Frame in HEX format” tak, jak w Zadaniu 2; pozostaw wartość „Timestamp” =x00 na wszystkich bajtach,
- przyciskiem „Add” ‘Add frame to scenario’ dodasz ramkę danych CAN to tablicy,
- możesz dodawać nieskończenie wiele ramek danych CAN; tablicę przesuwa się wskazując na indeks wiersza wyświetlanego na samej górze tablicy.
- komendy zapisane w tablicy wysyłane są co 100 ms – można wykorzystać tę funkcjonalność, jednak zalecane jest stosowanie opóźnienia.

Opóźnienie wykonywania funkcji z tablicy:

- jeżeli wartość w bajcie ‘0’ „Timestamp” będzie różna od zera, program nie wykona komendy CAN (niezależnie od wpisanych w ramkę wartości),
- wartość w bajcie ‘0’ „Timestamp” określa ilość sekund opóźnienia, np. x02 odpowiada 2 s opóźnienia,
- ustaw zatem wartość w bajcie ‘0’ „Timestamp” odpowiadającą pożądanej ilości sekund późnienia i dodaj do tablicy scenariusza.

Zakończenie tablicy / zainicjalizowanie pętli:

- przygotuj ramkę danych CAN, aby WSZYSTKIE wartości = x00,
- dodaj „wyzerowaną” ramkę na koniec tablicy.

Opracowanie wyników:

Opis wyników powinien być tożsamy z wykonanymi zadaniami.

Literatura

1 https://autokult.pl/15862,rozstanie-z-gaznikiem-ewolucja-zasilania-silnikow-cz-1] [Dostęp:25.11.2017]
2 https://www.can-cia.org/can-knowledge/can/can-history/] [Dostęp:25.11.2017]
3 https://auobd2.com/service/controller-area-network-can-bus-diagnostics-456.html [Dostęp:25.11.2017]
4 http://mikrokontroler.pl/2013/06/10/interfejs-komunikacyjny-can-podstawy/ [Dostęp:25.11.2017]
5 Andrew S. Tanenbaum– „Sieci komputerowe', wyd. Helion 2004 Warszawa
6 Bosch CAN specification ver. 2.0
7 http://canbus.pl/index.php?id=3 [Dostęp:23.11.2017]
8 Introduction to the Controller Area Network (CAN) Application Report SLOA101B– August 2002–Revised May 2016 Steve Corrigan Texas Instruments
9 https://en.wikipedia.org/wiki/CAN_bus [Dostęp:5.12.2017]
10 https://www.kvaser.com/about-can/the-can-protocol/can-messages-23 [Dostęp:6.12.2017]
11 „Sieci CAN – część 1" – Elektronika praktyczna 7/2005
12 https://pl.wikipedia.org/wiki/Local_Interconnect_Network [Dostęp:23.12.2017]
13.http://www.nmea.org/Assets/nmea-2000-digital-interface-white-paper.pdf[Dostęp:26.12.2017]