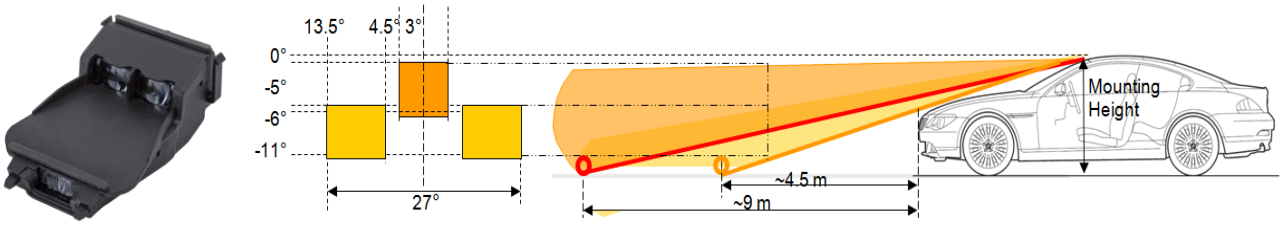
Uruchamianie systemów ADAS
(Advanced Driver Assistance Systems)

1. Wprowadzenie.

Celem ćwiczenia jest prezentacja narzędzi wykorzystywanych projektowania i weryfikacji zaawansowanych systemów wspomagających kierowców klasy ADAS (Advanced Driver Assistance System). W takcie realizacji ćwiczenia zostaną wykorzystane następujące elementy:

- Short Range Lidar (SRL) – urządzenie pozwalające na pomiar dystansu i prędkości pojazdu za pomocą trzech wiązek laserowych



- Środowisko uruchomieniowe CANoe, które pozwala na tworzenie i testowanie aplikacji ADAS oraz na symulację i weryfikację przygotowanych aplikacji ADAS. W trakcie laboratorium środowisko CANoe uzyskuje dane z symulatura

- Aplikacja pozwalająca na zdalne podłączenie do lidaru uruchamiana skrótem „LIDAR" umieszczonym na pulpicie. Po uruchomieniu należy wprowadzić adres IP serwera (157.158.38.214), wybrać Socket (w zakresie 10003 – 10008, inny dla każdego stanowiska laboratoryjnego), a następnie nawiązać połączenie za pomocą przycisku „Connect". Po nawiązaniu połączenia będą widoczne odległości i prędkości mierzone przez trzy wiązki lidaru.

- Przygotowany szkielet aplikacji, który będzie rozbudowywany w trakcie ćwiczenia. Szkielet znajduje się na pulpicie w folderze „ADAS" a umieszczony poniżej skrót „ADAS" otwiera aplikację w środowisku CANoe. Środowisko CANoe może pracować w trybie testowym (uruchamianym za pomocą żółtej błyskawicy umieszczonej w górnej części ekranu. I trybie projektowania aplikacji, aktywnym po zatrzymaniu trybu testowego (czerwony przycisk stop w górnej części ekranu).

Przygotowany szkielet aplikacji zawiera:
- panel graficzny stworzony za pomocą narzędzia Program Panel Designer (Tools->Panel Designer), który w trybie testowym prezentuje odległość mierzoną dla każdej z 3 wiązek lidaru,

- moduł ECU (otwierany z poziomu okna Simulation Setup) zawierający procedurę dekodowania odległości z ramek wysyłanych przez lidar. Skrypt przygotowany jest w języku CAPL.
Należy zwrócić uwagę na procedurę odpowiedzialną za odbiór danych: `on ethernetPacket msgChannel1` przeanalizować operację dekodowania ramek wykorzystując opis ramki pomiaru odległości z dodatku A.

2. Przygotowanie środowiska testowego dla aktywnego tempomatu:

Dodać następujące zmienne środowiskowe (Environment/System Variables), które będą wykorzystawane w projekcie aktywnego twmpomatu:

- ***Speed_Setup*** (Double) – zmienna odpowiadająca za zadawanie prędkości pojazdu w trybie aktywnego tempomatu (zakres 0 – 130 km/h)
- ***Tempomat*** (Tablicowa, Active_Inactive) - zmienna odpowiadająca za załączenie (Active) lub wyłączenie(Inactive) układu aktywnego tempomatu
- ***ShortDistance*** (Tablicowa, On_Off) – zmienna odpowiadająca za ostrzeżenie o zbyt bliskiej odległości za poprzedzającym pojazdem
- ***EmergencyBreak*** (Tablicowa, On_Off) –) zmienna odpowiadająca za wymuszenie awaryjnego hamowania
- ***EmergencyLights*** (Tablicowa, On_Off) - zmienna odpowiadająca za załączenie świateł awaryjnych
-***VeloLeftBeam***, ***VeloRightBeam***, ***VeloCentralBeam*** (Double) – zmienne odpowiadające za pomiar prędkości dla trzech wiązek lidaru
-***MinDist*** (Double) - wartość minimalna odległości dla trzech wiązek lidaru
-***MaxVelo*** (Double) - wartość maksymalna prędkości dla trzech wiązek lidaru

Utworzyć nowy panel graficzny (Tools/Panel Designer) pozwalający na:
- Zadawanie i wizualizację prędkości zadanej aktywnego tempomatu (Speed_Setup)
- Załączanie / wyłączanie tempomatu (Tempomat)
- Wizualizację odległości na postawie pomiarów z trzech wiązek „DistLeftBeam", „DistRightBeam", „DistCentralBeam „ oraz wizualizację odległości minimalnej „minDist"
- Wizualizację prędkości na postawie pomiarów z trzech wiązek „VeloLeftBeam", „VeloRightBeam", „VeloCentralBeam „ oraz wizualizację prędkości maksymalnej „maxVelo"
- Wizualizację sygnałów załączania świateł awaryjnych i awaryjnego hamowania „ADAS_EmergencyLights", "ADAS_EmergencyBreak"

3. Rozbudowa funkcjonalna modułu ECU o odczyt pomiaru prędkości dla trzech wiązek lidaru:

Zmodyfikować moduł odpowiedzialny za dekodowanie danych odbieranych siecią Ethernet `on ethernetPacket msgChannel1`
Opierając się na procedurze dekodowania odległości z sieci Ethernet dodać dekodowanie zmiennych odpowiedzialnych za pomiar prędkości dla trzech wiązek lidaru,: VeloLeftBeam", „VeloRightBeam", „VeloCentralBeam
Uwaga: wartości zmiennych środowiskowych przypisuje się bezpośrednio za pomocą znaku (=) poprzedzając ich nazwę symbolem (@)

4. Przygotowanie funkcjonalności aktywnego tempomatu:

a. W oknie Simulation Setup dodać nowy węzeł „Tempomat" (Insert CAPLTestModule). Podczas otwarcia skryptu należy podać nazwę pliku GxxSyy (xx – grupa, yy- sekcja) w którym unieszczony będzie kod tempomatu. Za pomocą edytora CAPL Browser dodać nową funkcję `on timer` (przykład użycia timera podany jest w treści programu odpowiedzialnego za dekodowanie ramek – fragment kodu wyłączony za pomocą komentarza).

Uwaga: timer nie jest wywoływany automatycznie (cyklicznie) lecz należy go aktywować w funkcji „on start" i powtarzać wywołanie w kodzie „on timer".

b. Zaimplementować następującą funkcjonalność tempomatu:
   - ♦ Tempomat powinien kontrolować odległość od poprzedzającego pojazdu (wartość minimalna pomiaru dla 3 wiązek lidaru) i generować sygnał :
     - ADAS_ShortDistance –w przypadku gdy odległość od poprzedzającego pojazdu (w mm) jest mniejsza niż Speed_Setup[km/h]^2 * 0,4
     - ADAS_EmergencyBreaking –w przypadku gdy odległość od poprzedzającego pojazdu (w mm) jest mniejsza niż Speed_Setup[km/h]^2 * 0,2

   - ♦ Tempomat powinien reagować na hamowanie pojazdu poprzedzającego (wzrost prędkości maksymalnej –dla trzech wiązek) i generować sygnał :
     - ADAS_EmergencyLights sygnał powinien przyjąć wartość 1 w przypadku gdy prędkość poprzedzającego pojazdu przekracza wartość (Speed_Setup[km/h]/ 10
   - ♦
   - ♦ Tempomat powinien być wyłączany ręcznie (przyciskiem z panelu graficznego) oraz w sposób automatyczny: w chwili wykrycia sygnału ADAS_EmergencyBreaking lub ADAS_ EmergencyLights.

   - ♦ W oknie wykresów prezentujące przebiegi czasowe dodać przebiegi minimalnej odległości, maksymalnej prędkości i sygnałów: ADAS_ShortDistance, ADAS_EmergencyBreaking, ADAS_EmergencyLights
   - ♦ Przeprowadzić testy stworzonego układu ADAS. Zrzut ekranu przedstawiający prawidłowe działanie sygnałów: ADAS_ShortDistance, ADAS_EmergencyBreaking, ADAS_EmergencyLights należy dołączyć do sprawozdania.

5. Sprawozdanie

Spakowany plik sprawozdania powinien zawierać:
- ♦ Skład sekcji laboratoryjnej i nazwę ćwiczenia.
- ♦ Dla ćwiczenia 2: zrzut ekranu ilustrujący panel graficzny tempomamtu (wraz z opisem poszczególnych elementów)
- ♦ Dla ćwiczenia 3: fragment kodu w języku CAPL odpowiadający za dekodowanie pomiarów prędkości wraz z komentarzami.
- ♦ Dla ćwiczenia 4: zrzut ekranu „simulation setup" pokazujący przebiegi czasowe zaimplementowanych sygnałów.
- ♦ Spakowany projekt (folder ADAS na pulpicie).

Kryteria oceniania:
- ♦ kompletność wykonanych ćwiczeń,
- ♦ przejrzystość kodu i komentarzy.

Appendix A – CAN frames

The lidar works with usage of three channels: left, central and right. Channels are directly connected with laser beams transmitted by device. According to manufacturer's documentation beams are emitted with 15-degree angular difference in horizontal plane.
Lidar provides three type of measurement forms. Each is contained in one CAN frame:

- 0x1E5 – relative distance to objects

Table 1.1. – list of fields in CAN frame 0x1E5

Field	Data length	Startbit	Variable type and Unit
RltvDistnc_LBeam	16	8	Long Ing, milimeters
RltvDistnc_CBeam	16	24	Long Ing, milimeters
RltvDistnc_RBeam	16	40	Long Ing, milimeters

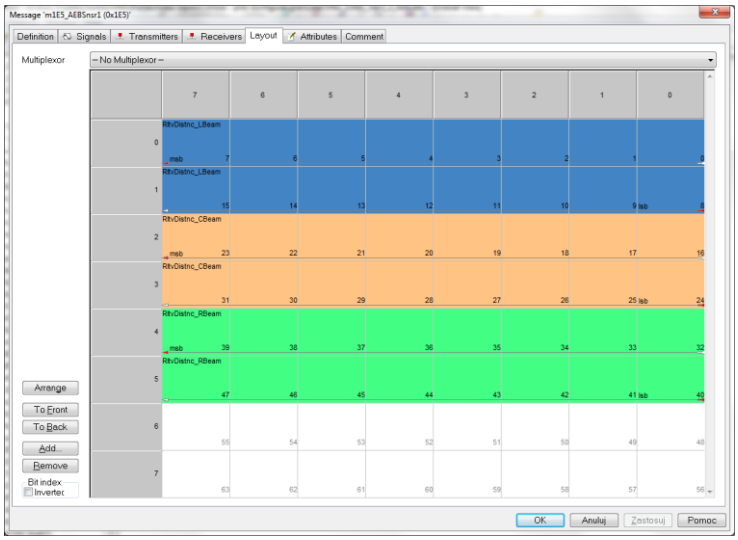


Fig. 1.1. – layout of frame 0x1E5

- 0x1E6 – relative velocity to objects

Table 1.2. – list of fields in CAN frame 0x1E6

Field	Data length	Startbit	Variable type and Unit
RltvVelo_LBeam	16	8	Long Int, km/h
RltvVelo_CBeam	16	24	Long Int, km/h
RltvVelo_RBeam	16	40	Long Int, km/h

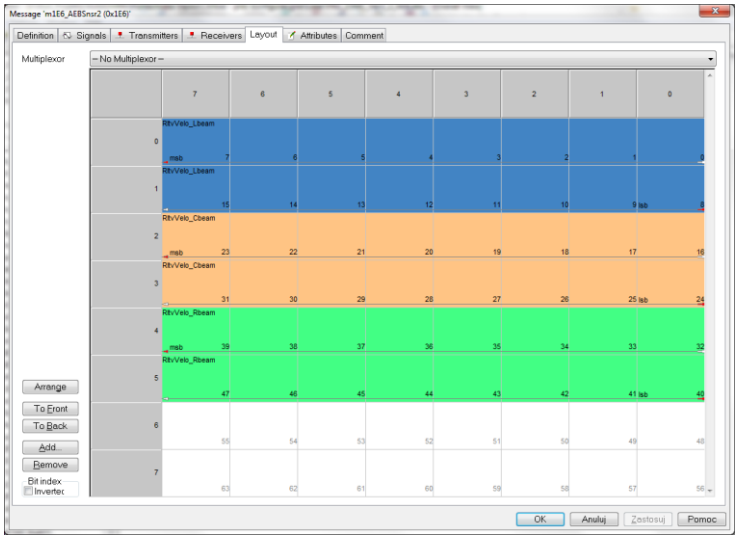


Fig. 1.2. – layout of frame 0x1E6

- 0x1E7 – reflection intensity and additional information

Table 1.3. – list of fields in CAN frame 0x1E7

Field	**Data length**	**Startbit**	**Variable type and Unit**
ReflecIntns_LBeam	8	0	Unknown
ReflecIntns _CBeam	8	8	Unknown
ReflecIntns _RBeam	8	16	Unknown
RltvDistnc_AEB	16	32	Long Int, meters
RltvVelo_AEB	16	48	Long Int, m/sec
CVSnsrQualInfo	4	60	Unknown

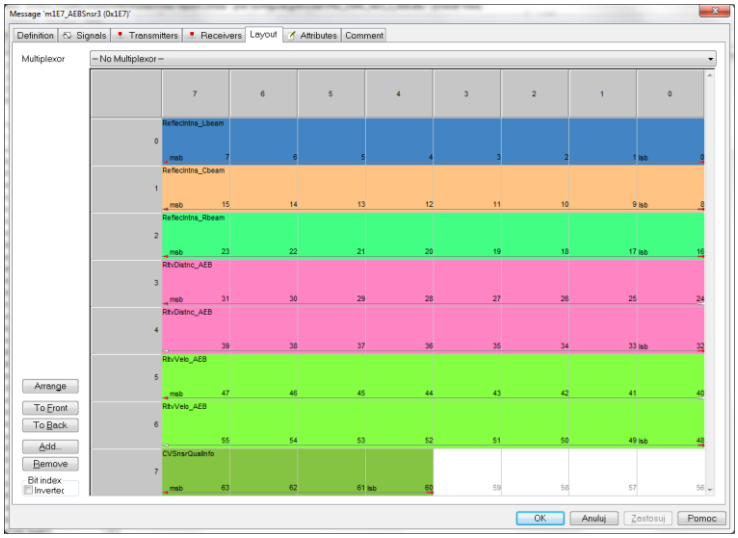


Fig. 1.3. – layout of frame 0x1E7