Escuela Politécnica Superior de Jaén

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN

Trabajo Fin de Grado

ADAPTACIÓN DE SEÑALES HARDWARE IN THE LOOP

Alumno: José María González León

Tutor: Prof. D. Florencia Almonacid Cruz

D. Xavier Sellart Ortega

Dpto: Electrónica

Junio, 2017

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Doña Florencia Almonacid Cruz y Don Xavier Sellart Ortega, tutores del Proyecto Fin de Carrera titulado: ADAPTACIÓN DE SEÑALES HARDWARE IN THE LOOP, que presenta JOSÉ MARÍA GONZÁLEZ LEÓN, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2017

El alumno: Los tutores:

JOSÉ MARÍA GONZÁLEZ LEON

XAVIER SELLART ORTEGA

FLORENCIA ALMONACID CRUZ

Índice

RESUMEN

En este trabajo se va a llevar a cabo el desarrollo de la primera fase de un proyecto interno de la compañía Applus IDIADA, el cual comenzó en 2017. Esta es compañía situada en el polígono de l'Arbonar, Tarragona. Applus IDIADA es una empresa multinacional que ofrece servicios de diseño, pruebas, ingeniería y certificación para la industria del automóvil en todo el mundo.

El proyecto consistirá en la **validación de una Unidad electrónica de Control (ECU) de un vehículo con funcionalidades de asistencia al conductor, Advanced Driver-Assistance Systems (ADAS),** y para ello se usará la tecnología de simulación **Hardware in the Loop (HiL)**. La finalidad de este proyecto es implementar, en un hardware capaz de trabajar en tiempo real, la planta de un vehículo para más adelante poder conocer, en un entorno realista simulado y bajo estímulos reales, cómo responde el controlador que se desea validar.

La novedad que presenta este **HiL** en concreto frente a otros desarrollados previamente en la industria de la automoción es que, en éste, se validarán funciones **ADAS**, como **Frenada de Emergencia (AEB)**, **Aviso de colisión frontal (FCW), Seguidor de carril (LKA)** o **Velocidad de crucero adaptativa (ACC)**.

Debido a la extensión de este proyecto, en este TFG se describirá solamente la primera fase de este proyecto, la cual se describe brevemente a continuación.

Se desea llevar a cabo la simulación de los sensores que integra el vehículo, y de los distintos escenarios donde éste será sometido a validación.

1. INTRODUCCIÓN

En esta sección se presenta en primer lugar, una breve introducción del proyecto que va a ser llevado a cabo en IDIADA, y a continuación, se define el alcance de este TFG.

1.1 Origen del Proyecto

Este proyecto nace en enero de 2017 de la colaboración de dos compañías; **National Instruments**, una empresa dedicada al desarrollo y venta de productos de software, hardware y servicios, e **IDIADA Automotive Technology S.A.,** una compañía líder especializada en diseño, ingeniería, ensayos y servicios de homologación para la industria del automóvil internacional.

De la unión de estas 2 grandes empresas surge la idea desarrollar un banco de pruebas **ADAS-HiL**, donde ambas mostrarán sus capacidades en el mundo de la automoción. National Instrument aportará sus herramientas hardware y software, e IDIADA aportará su experiencia y conocimientos en este tipo de tecnología de validación.

Este proyecto tiene como finalidad realizar pruebas de validación, donde todo el sistema excepto el dispositivo a validar será simulado. Esto permitirá desarrollar infinidad de casos de uso y test, que antes no eran posibles debido a su complejidad, coste o riesgo que éstos implicaban.

La tecnología HiL, es por tanto un método de validación de gran utilidad, y complementario a los ya existentes, como las pruebas en pista.

1.2 Motivación

En el desarrollo de sistemas de automoción, es necesario realizar una gran cantidad de pruebas, hasta hace pocos años, la única forma que se tenía de realizar este tipo de pruebas, era sobre la planta real del sistema, es decir, en pistas y sobre el propio vehículo que se desea validar.

En muchos casos, la forma más efectiva de desarrollar un sistema integrado de validación es conectar el sistema embebido a la **planta real**. En otros casos, la

simulación HiL es más eficiente. Las variables que determinan el desarrollo y la eficiencia de una prueba es una fórmula que incluye los siguientes factores:

- **Coste**
- **Duración**
- **Seguridad**

Coste: el coste de la realización de pruebas en un vehículo real es muy elevado, ya que además del propio vehículo, es necesario una infraestructura donde éste pueda ser testado.

Duración: para realizar pruebas en un vehículo real es necesario un prototipo de vehículo, o el propio vehículo. Utilizando la simulación HiL, las pruebas de conducción pueden realizarse mucho antes de que el vehículo esté disponible.

Seguridad: el uso de la prueba de conducción para el desarrollo de funciones críticas, tiene una importante implicación en la seguridad. Si hay errores en el diseño del prototipo, el resultado podría ser un accidente.

Por lo tanto, debido a las ventajas anteriormente nombradas, se ha considerado la tecnología de HiL una solución útil, económica y viable para la simulación de funciones **ADAS**.

1.3 Alcance del TFG

En este Trabajo de Fin de Grado se realizará la primera fase del sistema de validación HiL, la cual se encargará del sistema de simulación.

En primer lugar, se desea simular los sensores que incluye el vehículo a validar, concretamente un radar y una cámara.

En segundo lugar, se configurarán los distintos escenarios simulados donde será validado el funcionamiento de la ECU ADAS.

Se profundizará más adelante (véase Sección 3.2) acerca del alcance tanto del proyecto, como de este TFG.

2. ESTADO DEL ARTE

En esta sección del documento, se explican las diferentes tecnologías, dispositivos, metodologías y protocolos. El propósito de esta sección es poner el proyecto en un contexto científico y tecnológico.

2.1 Metodología

En el enfoque clásico del diseño de un sistema, la metodología empleada consiste en acercarse a las tareas en un proceso de prueba y validación de múltiples etapas. En la mayoría de los casos, el primer enfoque es un diseño y validación de un modelo conceptual y a un alto nivel de abstracción. A continuación, el modelo se implementa en software y, finalmente, el programa se ejecuta en el hardware final, que está conectado en un simulador que simula las señales de entrada y de salida para el HiL

Diagrama en V

El diagrama V mostrado en la figura es un modelo de ingeniería de software utilizado para demostrar el paralelismo entre el desarrollo del producto y el desarrollo de pruebas de validación para ese producto.

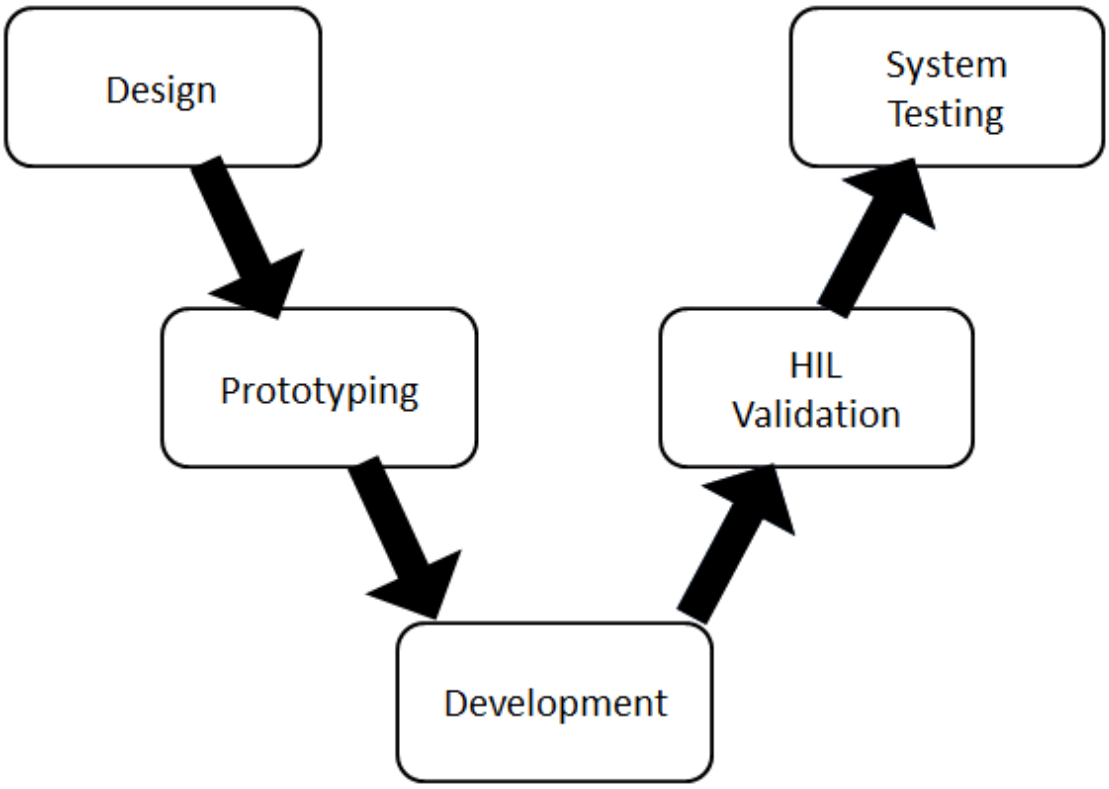


Ilustración 1. Modelo en V

El lado izquierdo de la V representa actividades de desarrollo y el lado derecho representa actividades de verificación y validación.

El diagrama V acorta el tiempo hasta la finalización al reducir la repetición de tareas durante el desarrollo y la verificación.

Analizando más en detalle cada fase del diagrama se tendría:

- Fase de diseño en la cual partiendo de unos requisitos de entrada se establece cómo se realizará la implementación del proyecto que se desea desarrollar.
- Fase de prototipado, en esta fase, partiendo de los requerimientos establecidos de entrada, se desarrolla un modelo prototipo del proyecto a un alto nivel, un modelo conceptual del proyecto donde se comprueba que la lógica de éste funciona correctamente.

- Fase de desarrollo, es la fase donde, una vez ya se tiene cierta seguridad de la funcionalidad del modelo prototipo del proyecto, este se implementa para la fase de producción.
- Validación HiL, es la fase donde se somete el sistema desarrollado a validación, esta validación HiL implica que la planta del sistema que se desea validar no es la original, sino una simulada.
- Fase de test, donde el sistema se somete a un conjunto de test automatizados.

Closed Loop Control

Hay dos componentes principales de un sistema de control de lazo cerrado: el **Controlador** y la **Planta**. El controlador recibe la respuesta deseada como una entrada, conocida como **Setpoint**. El controlador utiliza este valor para generar un estímulo para la planta. La planta recibe ese estímulo e intenta lograr la respuesta deseada. La planta puede experimentar una o más perturbaciones (factores externos que afectan el comportamiento de la planta), lo que dificulta su capacidad para alcanzar el punto de ajuste. Basado en el comando del controlador y cualquier perturbación, la planta genera su respuesta real como salida. El controlador recibe la salida de la planta y la compara con la respuesta deseada. El controlador utiliza esta retroalimentación para ajustar los comandos que envía a la planta de nuevo, cerrando el bucle de control.

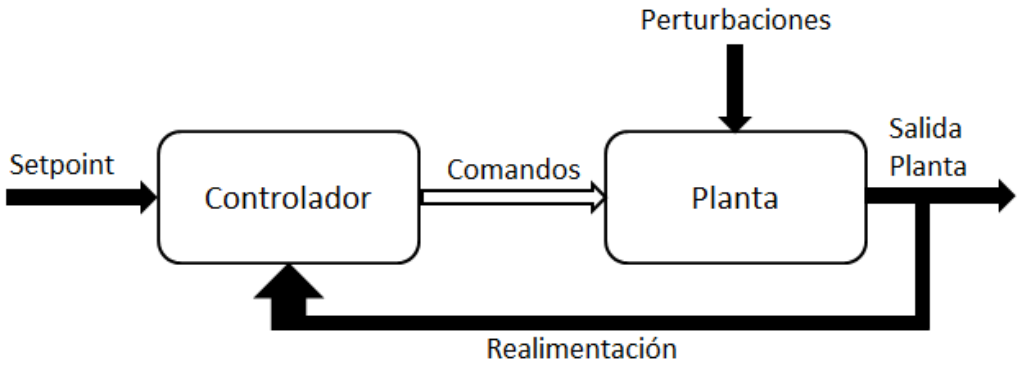


Ilustración 2. Bucle cerrado

Model-in-the-Loop (MiL)

Un modelo es una función matemática que representa cómo las salidas de un sistema son afectadas por las entradas del mismo.

El Model in the Loop **MiL** es el uso de un **entorno de prueba** para ejecutar los modelos de planta y controlador conjuntamente. La prueba MiL aplica conceptos de una prueba de sistema de control de lazo cerrado (estímulo, respuesta, planta, controlador, etc.) y los aplica a un entorno de software que ejecuta sus modelos de simulación, como se muestra en la figura:

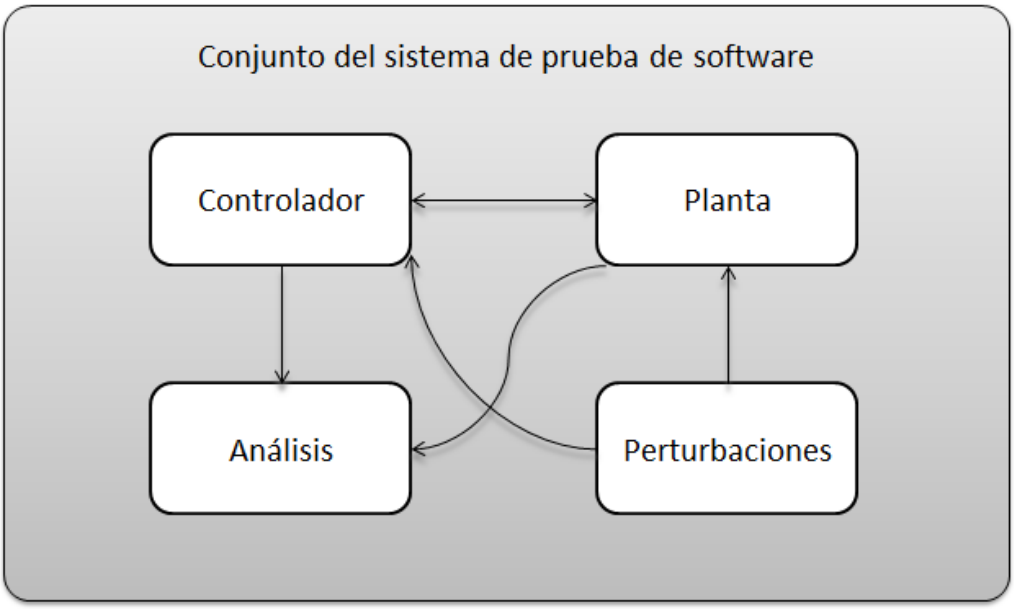


Ilustración 3. Model in the Loop

Después de haber desarrollado un modelo para su controlador, la prueba MiL le ayuda a validarlo sin requerir que tenga una planta física a mano.

Este método de prueba le permite probar y alterar rápidamente el modelo del controlador. También permite identificar los problemas del modelo de controlador sin el riesgo de dañar el hardware de la planta.

Software-in-the-Loop (SiL)

La prueba del SiL se utiliza comúnmente para probar la funcionalidad de una parte específica de código antes de integrarlo en un modelo.

La prueba SiL utiliza el mismo marco que las pruebas MiL.

Rapid Prototyping (RP)

Rapid Prototyping o prototipado rápido, es la ejecución de un sistema en bucle cerrado usando un hardware real para la planta y un modelo para el controlador. La finalidad del rapid prototyping es caracterizar cómo el controlador debería comportarse.

Hardware-in-the-Loop (HiL)

En la simulación HiL, se utiliza una computadora en tiempo real como representación virtual del modelo de planta y una versión real del controlador. De manera que, el modelo de controlador que usábamos antes para la prueba de MiL se reemplaza por el controlador real. La figura muestra una configuración típica de simulación HiL.

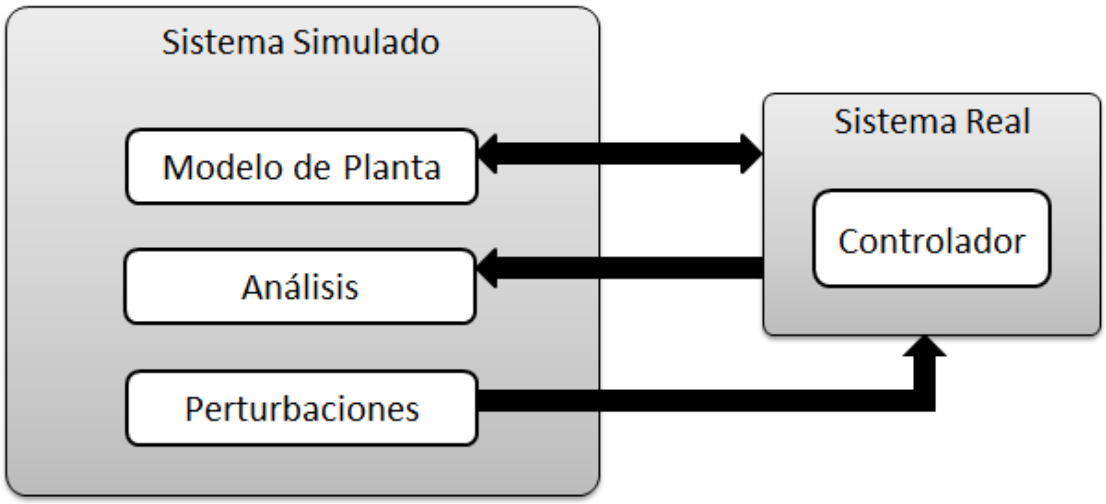


Ilustración 4. Hardware in the Loop

El objetivo de la prueba HiL es validar el controlador utilizando una planta simulada. También puede utilizar HiL para determinar si su modelo del sistema físico (planta) es válido.

Para entender el funcionamiento de la simulación HiL, hay que considerar lo que su dispositivo bajo validación "sabe acerca del mundo que lo rodea" Es decir, ¿cómo interactúa con el mundo real? Todo lo que la UUT "conoce" del mundo que la rodea es a través de sensores y señales eléctricas. Si podemos conservar las

características de voltaje, corriente, carga y tiempo del resto del sistema, el dispositivo bajo validación no "conocerá" la diferencia entre estar conectado al sistema real o una versión simulada del mismo.

2.2 Terminología

Advanced Driver Assistance Systems (ADAS)

Advanced Driver Assistance Systems son sistemas incluidos en un vehículo que tienen como objetivo mejorar la movilidad y seguridad del vehículo. Con el bajo coste actual de los sistemas de sensores y la electrónica, múltiples funciones ADAS han sido desarrolladas recientemente.

Los sistemas ADAS difieren de los sistemas de seguridad clásicos en el sentido de que predicen y evitan accidente de antemano, lo que se conoce como seguridad pasiva, es decir, que pueden tener un conocimiento básico de los alrededores del vehículo y tomar una decisión.

Actualmente, muchos fabricantes de automóviles ofrecen ADAS como complementos a los vehículos especialmente de alta gama y vehículos de lujo. Aquellos tipos de vehículos por lo general, un enorme abanico de posibilidades ADAS (Lane Keeping Assistance, Adaptative Cruise Control, Advance Emergency Brake, Fordware Collision Warning, etc). El precio de estas funciones se ha reducido considerablemente, incrementando el mercado de estos sistemas, de tal manera que en un pequeño periodo de tiempo también estará disponible para los vehículos de gama baja.

El sistema clásico ADAS utiliza los datos obtenidos únicamente del propio vehículo, esto quiere decir, sin necesidad de una infraestructura, como satélite para la localización GPS u otras. Sin embargo, la tendencia que se está adaptando, es que los coches usen la infraestructura de interconexión utilizando V2V (Vehículo a Vehículo) y V2I (Vehículo a Infraestructura) como fuente alternativa de información. Esta información puede ser utilizada para desarrollar nuevos sistemas ADAS, que también podrán tener una capacidad de decisión más compleja y desarrollada, ya que no sólo tienen la información del propio vehículo, sino también la de su entorno.

Algunas de las funciones ADAS más comunes en el mercado actualmente son:

- **AEB → Automated Emergency Brake** (Frenada de emergencia)

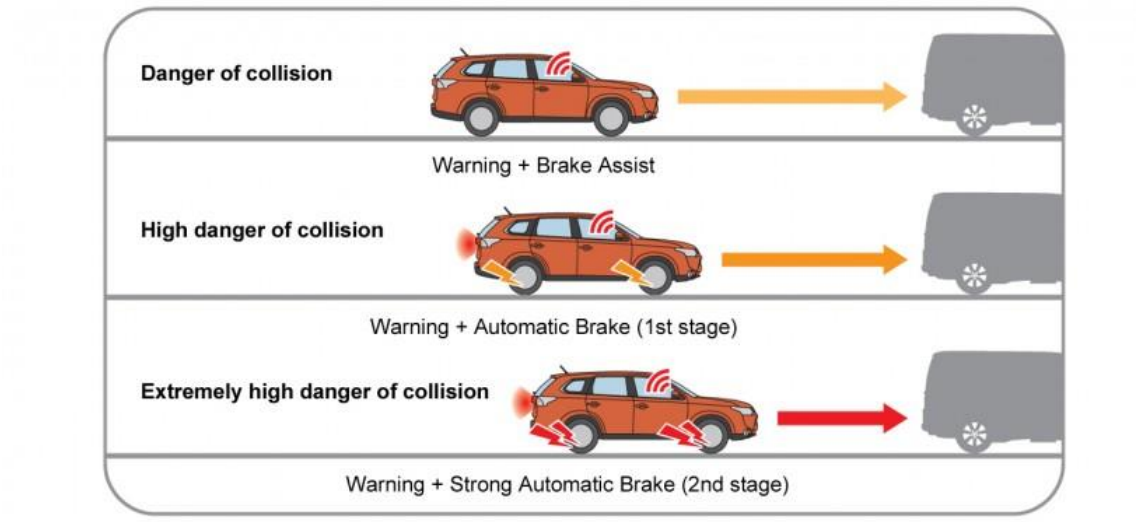


Ilustración 5. AEB

AEB es un sistema de prevención de colisiones es un sistema de seguridad del automóvil está diseñado a reducir la gravedad de una colisión o evitarla. También se conoce como un sistema precrash, sistema de alerta de colisión frontal, o sistema para mitigar la colisión. Se utiliza el radar (en todo tipo de climas) y, a veces láser (LIDAR) y cámara (para el empleo de reconocimiento de imágenes) que detectan una colisión inminente.

- **FCW → Forward Collision Warning** (Aviso de colisión frontal)

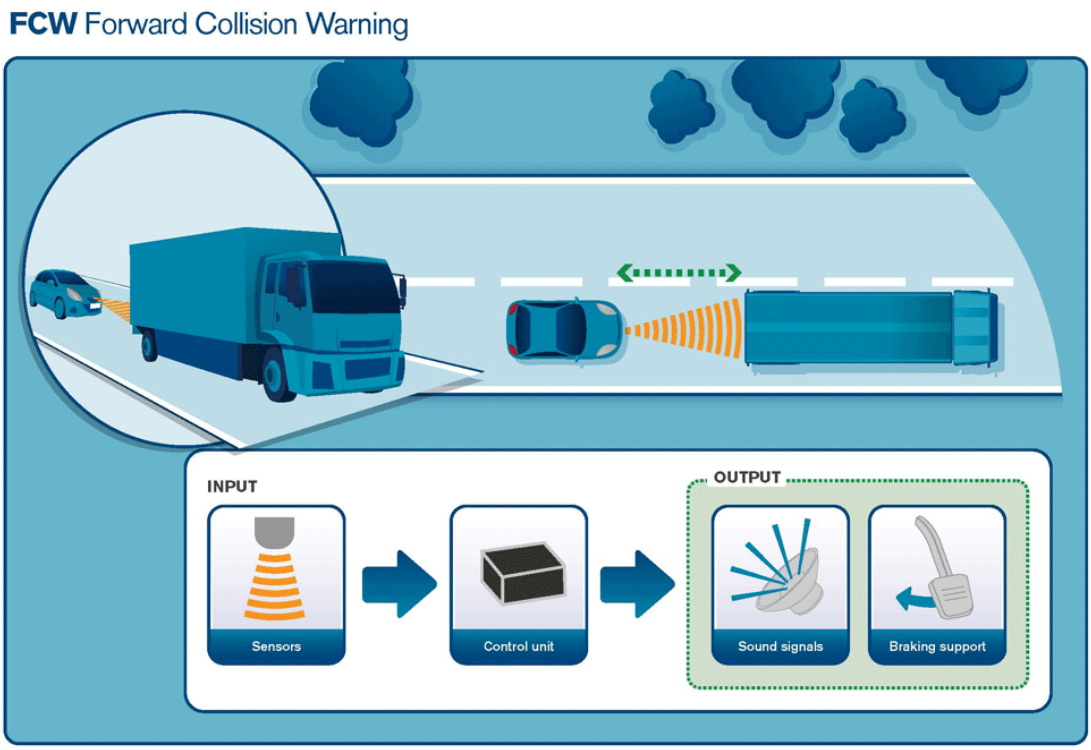


Ilustración 6. FCW

El sistema de aviso de colisión frontal (FCW) utiliza un radar que detecta los vehículos u obstáculos en la parte delantera del coche. El sistema calcula la distancia al objeto de delante y, si el coche se encuentra lo suficiente cerca como para que haya un riesgo de colisión, suena una alarma y muestra una alerta visual, impulsando al conductor a aplicar una frenada.

El aviso de colisión frontal le avisa de si un objeto en su trayectoria se ha detenido o ralentizado, para que pueda reaccionar más rápido. FCW reducirá significativamente la posibilidad de un choque o un accidente mortal.

- **ACC → Automated Cruise Control** (Velocidad de crucero adaptativa)

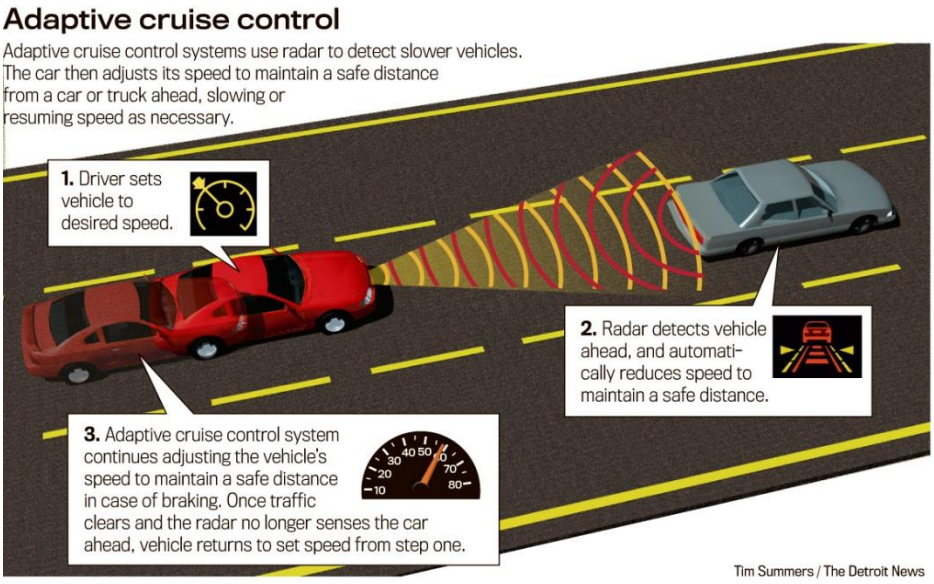


Ilustración 7. ACC

ACC, (también llamado control de crucero adaptativo, control radar de crucero, control de crucero o tráfico conscientes) es un sistema de control de crucero opcional para vehículos de carretera que ajusta automáticamente la velocidad del vehículo manteniendo una distancia segura de los vehículos que se encuentran por delante.

El control se basa en la información de los sensores de a bordo. (Ningún sistema hace uso de las infraestructuras como satélites o de carretera ni de soporte cooperativo de otros vehículos.)

CAN. (Controller Area Network)

Es un protocolo de comunicaciones serie que soporta eficientemente el control distribuido en tiempo real con un nivel de seguridad muy alto.

CAN es el protocolo de comunicaciones estándar en la automoción y proporciona los siguientes beneficios:

- Ofrece alta inmunidad a las interferencias, habilidad para el autodiagnóstico y la reparación de errores de datos.
- Es un protocolo de comunicaciones normalizado, con lo que se simplifica y economiza la tarea de comunicar subsistemas de diferentes fabricantes sobre una red común o bus.

- El procesador anfitrión (*host*) delega la carga de comunicaciones a un periférico inteligente, por lo tanto, el procesador anfitrión dispone de mayor tiempo para ejecutar sus propias tareas.
- Al ser una red multiplexada, reduce considerablemente el cableado y elimina las conexiones punto a punto, excepto en los enganches.

CAN dbc

Una dbc, es una base de datos que contiene información sobre los mensajes CAN.

Al desarrollar una red ECU distribuida basada en CAN, un componente clave es la descripción de la comunicación en forma de archivos DBC.

En las bases de datos DBC se describen las propiedades de la red CAN, las ECUs conectadas al bus y los mensajes y señales CAN.

ECU (Electronic Control Unit)

La ECU, o centralita electrónica es un dispositivo electrónico normalmente conectado a una serie de sensores que le proporcionan información y actuadores que ejecutan sus comandos. Una centralita electrónica cuenta con software cuya lógica le permite tomar decisiones (operar los actuadores) según la información del entorno proporcionada por los sensores.

Sistema en tiempo real (STR)

Un STR es aquel sistema digital que interactúa activamente con un entorno con dinámica conocida en relación con sus entradas, salidas y restricciones temporales, para darle un correcto funcionamiento de acuerdo con los conceptos de predictibilidad, estabilidad, controlabilidad y alcanzabilidad.

2.3 Descripción Herramientas

2.3.1 Hardware

MicroAutoBox II

MicroAutoBox II es un sistema en tiempo real para realizar prototipos de funciones rápidas. Puede funcionar sin la intervención del usuario, al igual que un ECU.

En este sistema correrá el modelo Simulink desarrollado.

Ilustración 8. MicroAutoBox II

Real Time PC-PXI

PXI es una plataforma robusta basada en PC para sistemas de medición y automatización. PXI es una plataforma de despliegue de alto rendimiento y bajo costo para aplicaciones tales como prueba de fabricación, militar y aeroespacial, monitoreo de máquinas, automoción y pruebas industriales.

En este caso será el ordenador en tiempo real donde será simulada la planta del vehículo a validar.

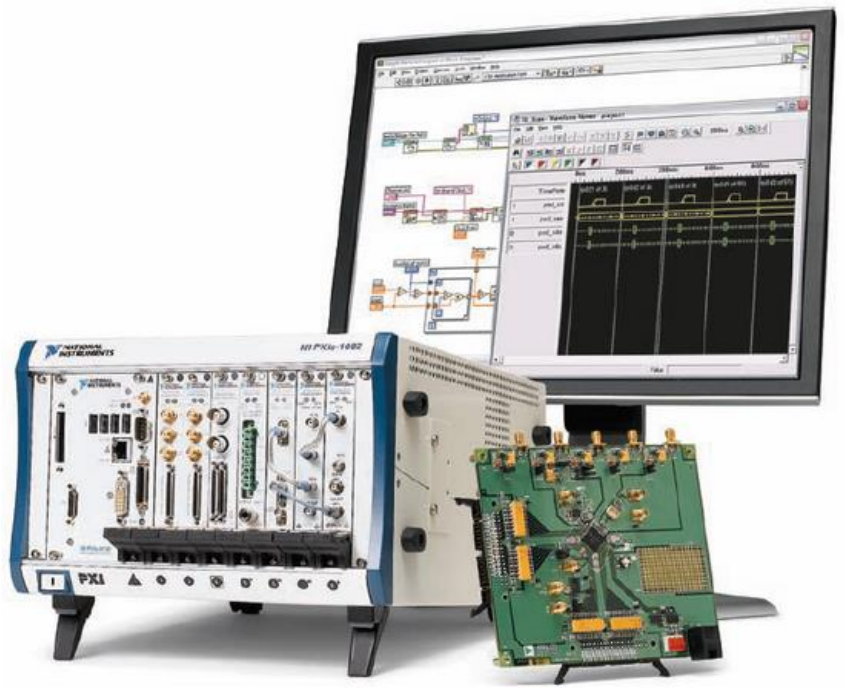


Ilustración 9. Real Time PC-PXI

PC Simulación

Será el ordenador donde se ejecuta la simulación. Es un PC de escritorio y tiene:

- CPU: Core i7-6700
- GPU: GeForce GTX TITAN x2
- RAM: 32 Gb
- SSD con suficiente espacio para SO, PreScan, modelos Simulink y cualquier otro programa necesario. No es un requisito indispensable el uso de este PC, pero mejora significativamente el rendimiento.

CANCaseXL

El CANcaseXL es una interfaz USB con dos controladores CAN que pueden enviar y recibir mensajes CAN con identificadores de 11 bits y 29 bits sin restricciones. Además, el CANcaseXL es capaz de detectar y generar tramas de error en el bus.

La función que desempeñará el CANCaseXL en este proyecto será, por una parte, grabar las tramas CAN, para después reproducirlas, y por otra parte será la conexión física entre el PC de simulación y el PC de Baselabs.

Ilustración 10. CanCaseXL

Radar

Información

El radar se ha usado debido a su alta precisión para la localización de objetos, sin embargo, la utilización únicamente de un radar no sería suficiente.

El radar genera unos mensajes vía CAN, como los que se adjuntan en la siguiente tabla.

CAN	Frame Format	Message ID[1]	Message Name	Content	Section
1	CAN 2.0A (11 Bit)	0x200	RadarConfiguration	Sensor configuration	0
1	CAN 2.0A (11 Bit)	0x201	RadarState	State output	0
1	CAN 2.0A (11 Bit)	0x300	SpeedInformation	Vehicle speed	8.1
1	CAN 2.0A (11 Bit)	0x301	YawRateInformation	Vehicle yaw rate	8.2
1	CAN 2.0A (11 Bit)	0x600	CAN1_Target_Status	Target status	9.1
1	CAN 2.0A (11 Bit)	0x60A	CAN1_Obj_Status	Object status	9.2
1	CAN 2.0A (11 Bit)	0x60B	CAN1_Obj_1	Object information 1	9.2
1	CAN 2.0A (11 Bit)	0x60C	CAN1_Obj_2	Object information 2	9.2
1	CAN 2.0A (11 Bit)	0x700	CAN1_VersionID	Object List Interface Version	9.2
1	CAN 2.0A (11 Bit)	0x701	CAN1_Target_1	Target information 1	9.1
1	CAN 2.0A (11 Bit)	0x702	CAN1_Target_2	Target information 2	9.1
1	CAN 2.0A (11 Bit)	0x400	CollDetWarnConfig	Collision detection warning configuration	10.1
1	CAN 2.0A (11 Bit)	0x401	CollDetRegionConfig	Collision detection region configuration	10.2
1	CAN 2.0A (11 Bit)	0x408	CollDetState	Collision detection warning state	10.3
1	CAN 2.0A (11 Bit)	0x409	CollDetWarn	Collision detection region warning	10.4

Tabla 1

Interfaz

El radar tiene una interfaz CAN (llamada CAN1). La red de comunicación es un bus CAN como se especifica en ISO 11898-2 con una velocidad de transmisión de 500 KBits / s.

Los mensajes en el CAN-Bus están definidos por los archivos **DBC** (Comunication DataBase for CAN) correspondientes.

Como ejemplo, en la figura se muestra una posible red de bus CAN. La conexión CAN1 del radar se utiliza para la configuración, la salida del estado del sensor, la entrada de datos y la salida de datos.

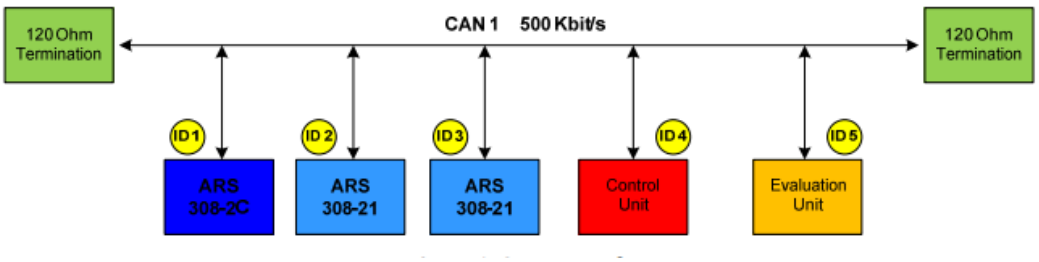


Ilustración 11. Esquema red bus CAN

Cámara

La cámara se utiliza para identificar los objetos, ya que tiene mejor precisión que el radar a corta distancia, y nos da una interpretación de los obstáculos encontrados en el entorno. Así, la función de la cámara es la de discriminar los objetos considerados por el radar como obstáculos que no supongan un peligro. Además, la cámara es útil determinando el tamaño de los obstáculos. Pudiendo determinar si el obstáculo es un peatón, un coche, una bicicleta u otro vehículo indeterminado.

Logitech

Se incorporará al entorno de simulación un volante y unos pedales Logitech G29, para realizar las simulaciones de conducción.

Ilustración 12. Logitech

2.3.2 Software

PreScan

PreScan es una plataforma de simulación basada en la física que se utiliza en la industria de automoción para el desarrollo de sistemas ADAS basados en tecnologías de sensores como radar, láser / lidar, cámara y GPS. PreScan también se utiliza para diseñar y evaluar aplicaciones de comunicación de vehículo a vehículo (V2V) y vehículo a infraestructura (V2I), así como aplicaciones de conducción autónoma. PreScan puede utilizarse desde el diseño de controlador basado en modelos (MiL) hasta pruebas en tiempo real con sistemas SiL y HiL.

En este proyecto se encargará de la simulación y representación de los escenarios virtuales.

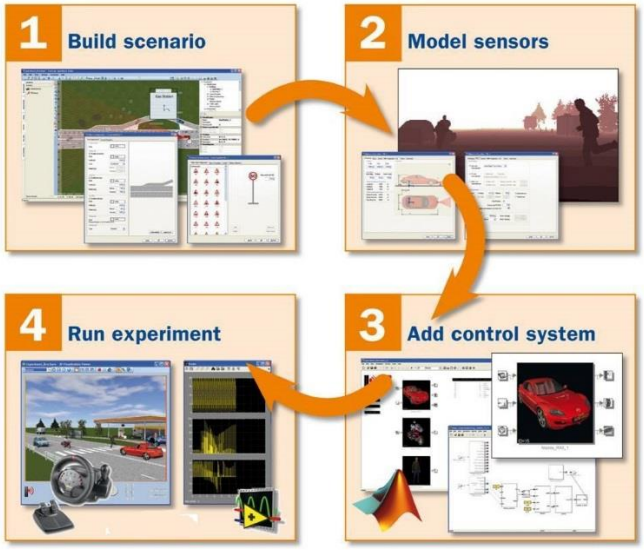


Ilustración 13. PreScan

Baselabs

BASELABS permite obtener resultados de fusión de datos en escenarios de múltiples sensores. Con la creciente complejidad de las aplicaciones, la fusión de sensores arbitrarios de diferentes tipos es cada vez más difícil.

BASELABS ofrece resultados de fusión de datos de sensores rápidos con el diseñador intuitivo de fusión de datos. Se trata de un marco de software prototipo que está diseñado para el rápido desarrollo de complejos algoritmos de fusión de

datos y modelos de entorno. El software es compatible con todos los productos middleware (lógica de intercambio de información entre aplicaciones) relevantes como Vector vADASdeveloper, EB Assist ADTF, Sistema Operativo de Robot (ROS), Simulink y otros. BASELABS Create proporciona numerosas funciones para el desarrollo de aplicaciones de fusión de datos.

Su función en este proyecto será la de fusionar los datos de los mensajes que recibe de un radar y de una cámara, la aplicación tendrá como inputs unos mensajes determinados de cada sensor, y como salida obtendremos región de incertidumbre sobre dónde se encuentran los obstáculos detectados por los sensores.

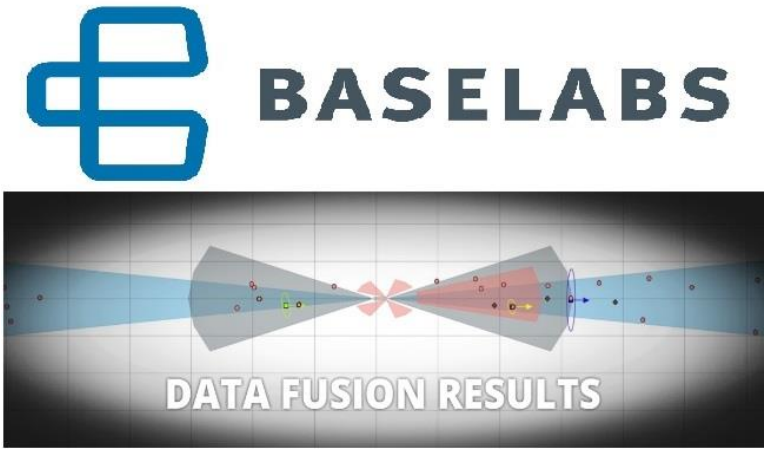


Ilustración 14. Baselabs

Simulink

Simulink es un entorno de programación visual, que funciona sobre el entorno de programación Matlab.

Es un entorno de programación de más alto nivel de abstracción que el lenguaje interpretado Matlab (archivos con extensión .m). Simulink genera archivos con extensión. mdl (de "model").

Simulink se comunica con PreScan, de manera que podremos interpretar los mensajes sobre la simulación creada previamente en PreScan para comunicarnos con éste.

En este proyecto será utilizado como lazo de conexión entre PreScan y Baselabs, ya que permitirá, a través de sus librerías interpretar los mensajes que le proporcione PreScan, adaptarlos, y enviarlos por CAN a Baselabs.

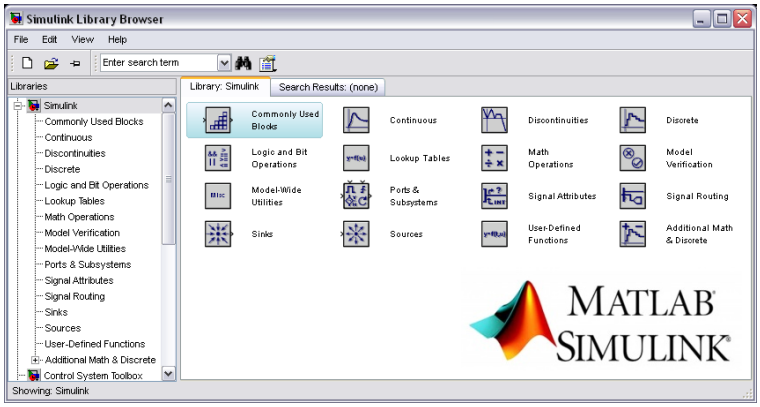


Ilustración 15. Simulink

CANalyzer

CANalyzer es la herramienta de software para el análisis y estimulación de la comunicación de bus, (en nuestro caso CAN). CANalyzer es la herramienta de software integral con operación intuitiva para el análisis y estimulación de la comunicación de bus.

Se utiliza CANalyzer para comprobar qué tipo de comunicación se está produciendo en el bus. También se puede utilizar para enviar, grabar, registrar y recibir datos.

En este proyecto será el software encargado de enviar, grabar, recibir y depurar los mensajes recibidos de Simulink a través del protocolo de comunicación CAN.

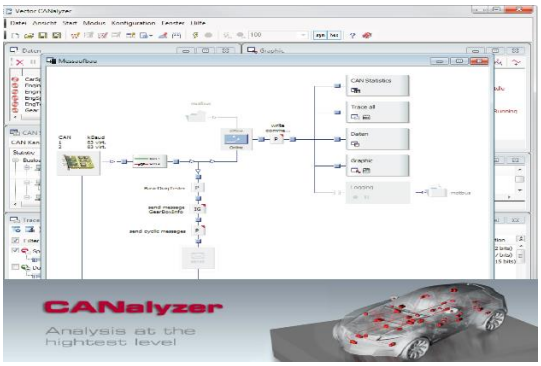


Ilustración 16. CANalyzer

2.4 DUT, Dispositivo Bajo Validación

Hay que tener en cuenta que esta es la primera fase de un proyecto de validación, pero para validar cualquier sistema, lo primero de todo es preguntarse; **¿Qué se desea validar?**, por lo tanto, en este apartado se explicará el proyecto que se desea validar.

¿Cuál es nuestro DUT (Device Under Test)?

El objetivo de esta sección estará enfocado al conocimiento del proyecto que se desea validar, para más adelante, poder establecer unos requisitos de validación acordes al proyecto.

El dispositivo bajo validación es una Unidad Electrónica de Control que incluye funciones ADAS. Concretamente tres funciones:

- ACC → Velocidad de crucero adaptativa
- FCW → Aviso de Colisión Frontal
- AEB → Frenada de emergencia autónoma

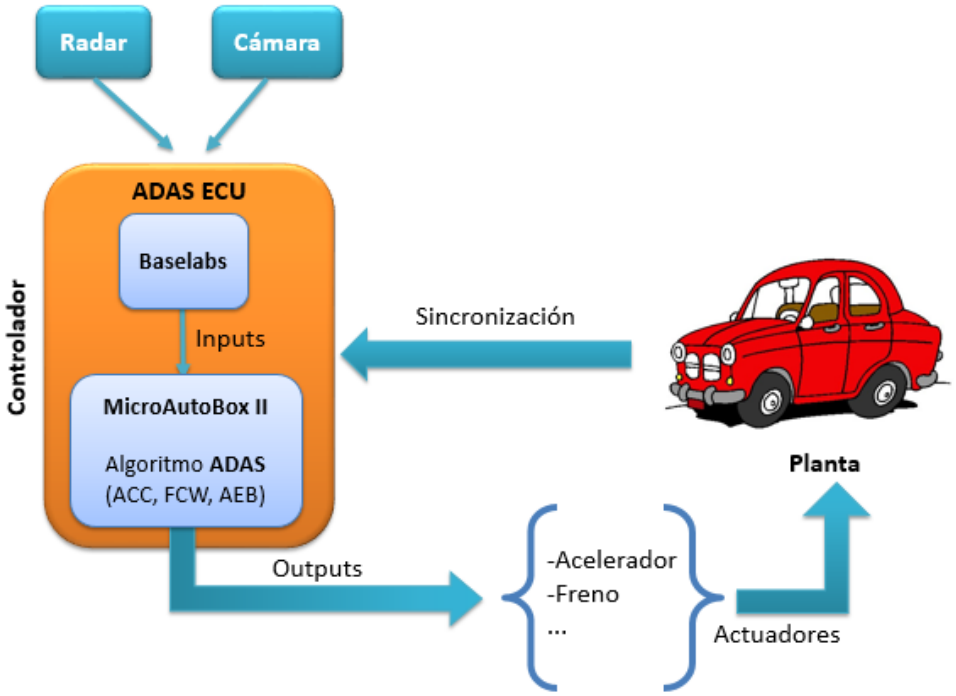


Ilustración 17. Esquema DUT

Este proyecto ha sido desarrollado en un vehículo de IDIADA, el cual ha sido utilizado como **planta real** del sistema.

El vehículo lleva incorporado una cámara y un radar, los cuales se encargan de recibir la información del entorno e interpretarla.

La información generada por estos sensores servirá de entrada al software de fusión de datos BaseLabs, el cual la tratará y generará a partir de ésta, una región de posible convergencia o colisión con un obstáculo, que se conoce como **región de incertidumbre**.

Una vez identificados los objetos, y su región de incertidumbre, la información es enviada a la **MicroAutobox II,** una Unidad Electrónica de Control de prototipo, donde se encuentra el modelo desarrollado por IDIADA con las funciones ADAS, además de toda la información de las diferentes ECUs del vehículo; como el ABS, BCM, HMI etc. Es el cerebro del sistema.

En dicha ECU se generan unos mensajes de salida que irán a los actuadores de la planta, concretamente al acelerador y freno.

Una vez, los actuadores intervienen sobre la planta, ésta envía la información de vuelta a la MicroAutoBox, realimentándola y cerrando el bucle. De esta manera se sincroniza la planta con el controlador.

Finalmente se repite una y otra vez el ciclo en bucle cerrado.

El método de desarrollo para la implementación de este proyecto, ha sido, por tanto, MiL (Model in the Loop), y Rapid prototyping, donde, como ya se vio con anterioridad, se desarrolla un modelo prototipo a un alto nivel de abstracción y se prueba su funcionamiento directamente en pista, sobre la planta real (vehículo).

Este es un método de desarrollo bastante efectivo, pero tiene ciertas carencias, ya que, cada vez que se desea validar la funcionalidad del modelo, es necesario disponer de la planta real, es decir, el vehículo, además de un escenario donde validarlo, (pista o circuito). Por lo tanto, este método supone, además de un coste elevado, una limitación a la hora de representar ciertos escenarios que impliquen un riesgo o situaciones extremas. Por otra parte, en un MiL, la comunicación entre los distintos sistemas es ideal, la respuesta generada por los sensores también es ideal… Lo único que se valida en un MiL es que la lógica del algoritmo implementado funciona, esto conlleva a que la respuesta al realizar las pruebas correspondientes simuladas, puede diferir considerablemente de las pruebas realizadas en el propio vehículo, suponiendo una gran pérdida de tiempo y dinero.

Es por ello que se desea validar este proyecto siguiendo la metodología HiL.

Para entender mejor la metodología HiL debemos preguntarnos, **¿qué conoce el controlador acerca de la planta?** Al fin y al cabo, de lo único que entiende el vehículo es de señales eléctricas, que se comunican a través de un bus CAN. Por lo tanto, si somos capaces de simular estas señales en tiempo real, el controlador no conocerá diferencia entre vehículo real o planta simulada en tiempo real. Esto permitirá desarrollar y amplificar el número de pruebas sobre el controlador sin necesidad de la planta real, y de un escenario real.

3. Objetivos

3.1 Requisitos

Requisitos del sistema de validación

- **Usar el protocolo de comunicación CAN**

  Es fundamental el uso del protocolo de comunicación CAN, ya que cumple al 100% con los 3 parámetros en los que se basan dichos requisitos. En el vehículo, toda la comunicación será a través de CAN, por lo que, si se desea validar por completo la ECU, los mensajes deberán ser enviados de la misma manera.

- **Cumplir los tiempos de ejecución del sistema**

  Este requisito hace referencia a la generación, transmisión y recepción de los mensajes a las frecuencias determinadas. Éste es el principal motivo por el que para una validación HiL se necesita un ordenador en tiempo real, que cumpla con los tiempos con total precisión.

- **Realizar la simulación para conducción autónoma y manual.**

  Este requisito se ha establecido para poder automatizar la validación, y no tener que depender de una persona física para cada ensayo de validación.

3.2 Alcance del proyecto HiL

Es importante delimitar una frontera de validación del sistema. Especificar cuál es el alcance de este proyecto, y establecer un grado de validación sobre el **DUT**.

Dado que el propósito general de este proyecto es la validación de un algoritmo con funciones ADAS, no se entrará en detalle acerca del funcionamiento interno de los sensores, cámara y radar, quedando éstos fuera del alcance de este proyecto.

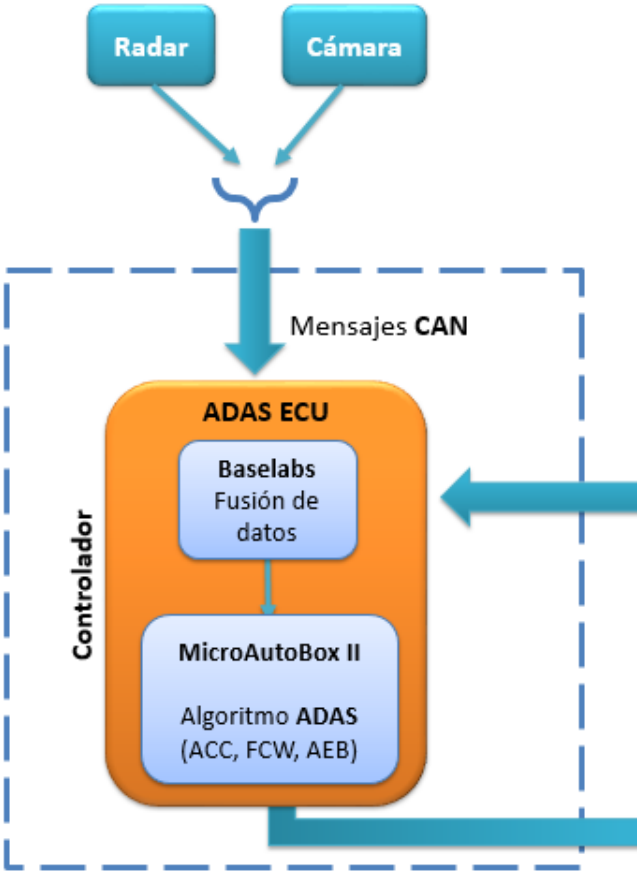


Ilustración 18. Esquema Alcance HiL

Por lo tanto, se considera un funcionamiento ideal de los sensores, partiendo nuestra validación desde los mensajes CAN generados por dichos sensores.

Alcance del TFG

Como ya se comentó con anterioridad, debido a la extensión de este proyecto, en este trabajo sólo se llevará a cabo la fase de simulación del escenario y los mensajes de los sensores.

En la siguiente imagen se puede apreciar cual será el alcance de este TFG.

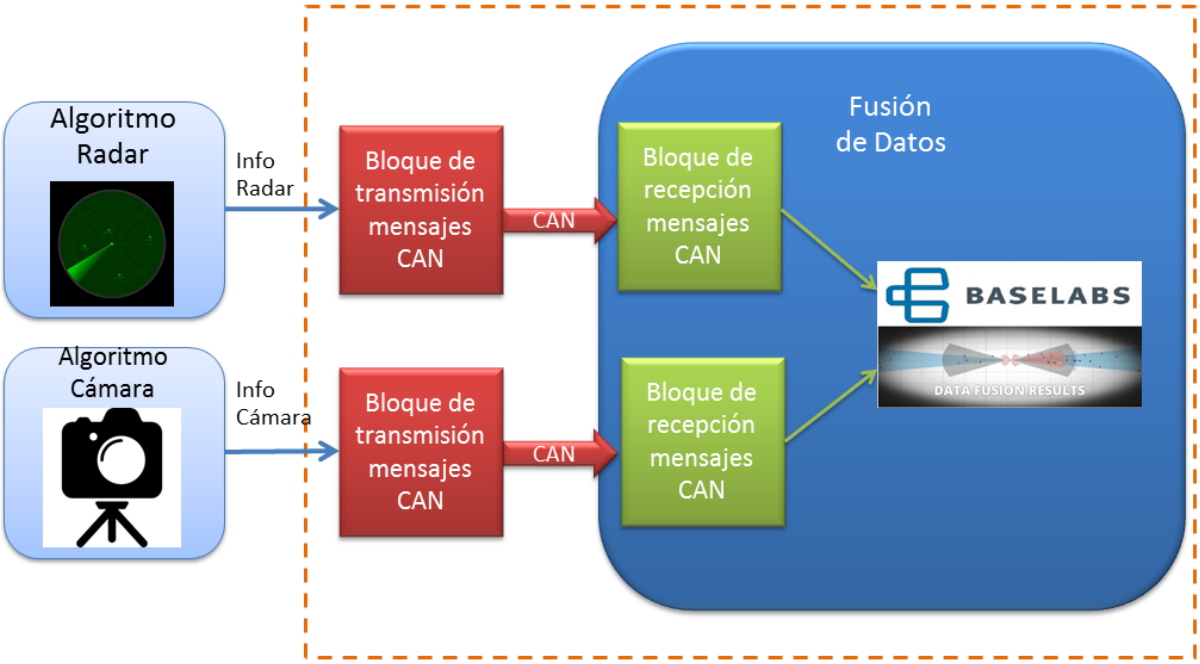


Ilustración 19. Alcance fase de simulación

3.3 Objetivos

El propósito de este proyecto es la validación de una **ECU** de un vehículo con funciones **ADAS** mediante la metodología **HiL.**

3.3.1 Objetivos Generales del HiL

1. **Instalación** e **integración** programación de los distintos equipos.
2. **Simulación** de diferentes escenarios en el entorno de simulación **PreScan**.
3. Establecer la **comunicación** mediante el protocolo de comunicaciones **CAN.**
4. **Configuración** y **simulación** de la planta del proyecto en el Real Time **PC-PXI**.
5. Pruebas en **Bucle Abierto**
6. Pruebas en **Bucle Cerrado** y **sincronización**.
7. **Automatización** del proceso mediante el uso de las herramientas de National Instruments como **VeriStand** y **TestStand**.

Dada la extensión de este proyecto, en este documento se llevará a cabo el desarrollo de la primera fase del proyecto, la cual implica los siguientes objetivos.

3.3.2 Objetivos Específicos del TFG

1. **Instalación** e **integración** de los distintos equipos Hardware y Software necesarios.
2. **Simulación** de los sensores en el entorno **PreScan.**
3. **Adaptación, transmisión** y **recepción** de las señales en el entorno de programación **Simulink** mediante el protocolo de comunicaciones **CAN**.
4. **Recepción**, **gestión** y **fusión** de datos de los mensajes recibidos desde Simulink en **Baselabs**.

3.4 Diagrama de Gantt

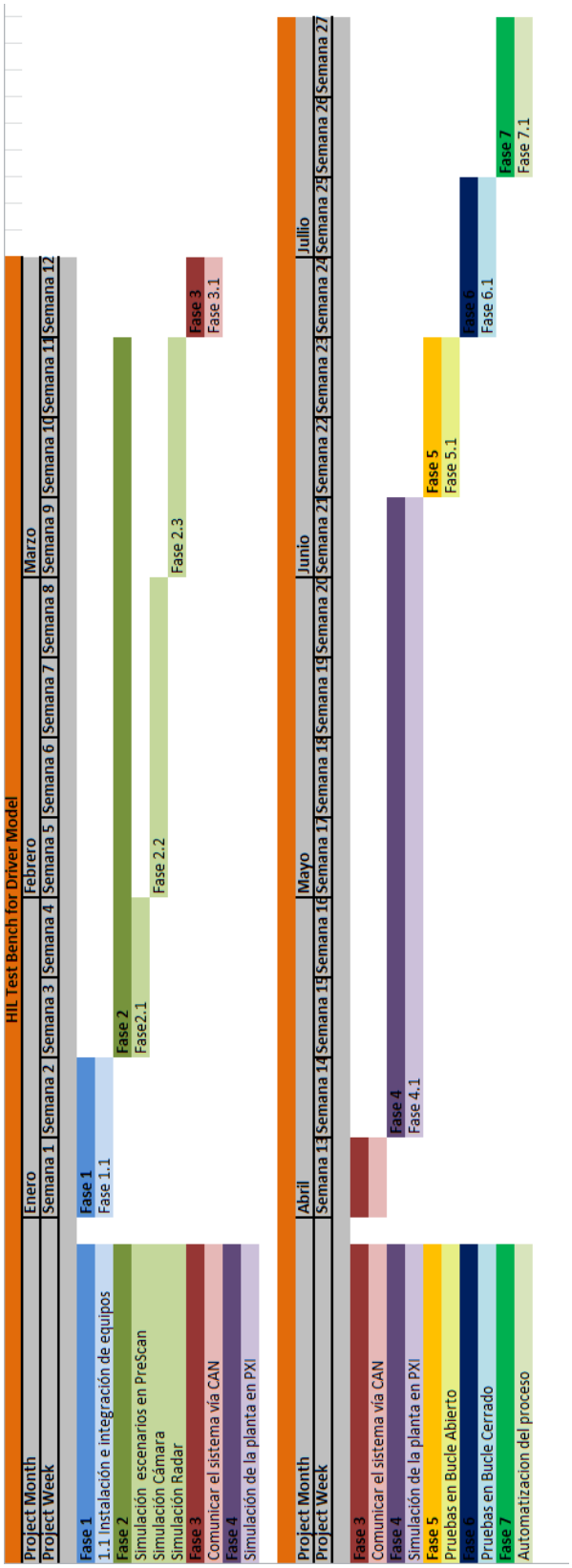


Ilustración 20. Diagrama de Gantt

4. Diseño del HiL

Se seguirá una estructura HiL como la explicada en el apartado 2.1

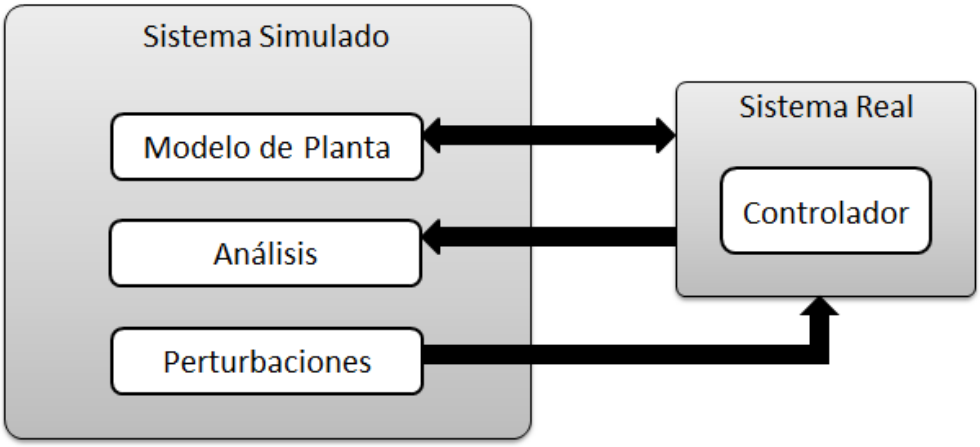


Ilustración 21. Estructura HiL

En esta estructura de simulación, la única parte real será el controlador, ya que es la parte que se desea validar.

- El modelo de planta se refiere al modelo dinámico del vehículo, si estuviéramos realizando una validación de Rapid Prototyping, la planta sería el propio vehículo. Al llevar a cabo una simulación HiL, el modelo de planta es simulado en el ordenador en tiempo real, PXI.
- Las perturbaciones para este sistema, pueden ser comprendidas como los obstáculos encontrados por los sensores, los cuales serán simulados en el entorno de simulación PreScan.
- El análisis de los mensajes generados en el escenario se realizará con la herramienta Simulink.

El sistema tendrá la siguiente estructura:

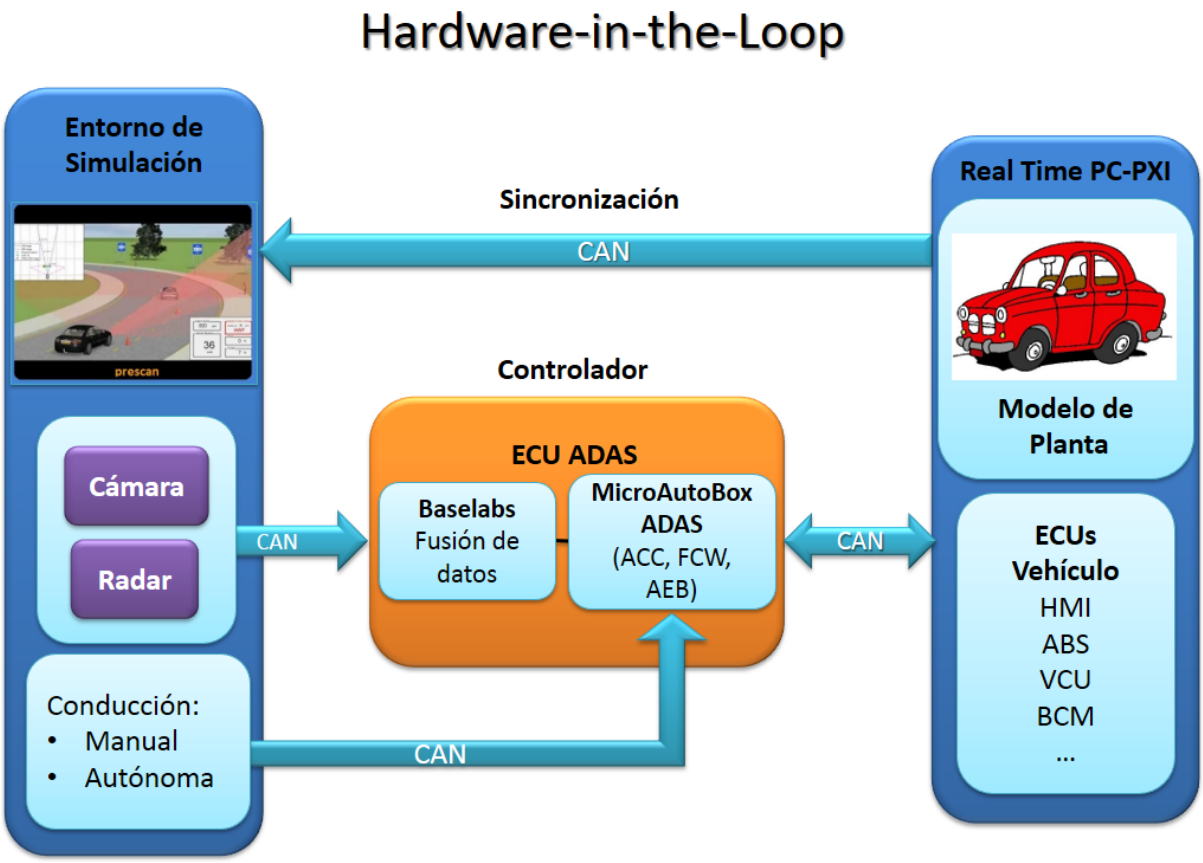


Ilustración 22. Esquema HiL

En esta otra imagen se muestra el esquema con las herramientas hardware y software que serán usadas.

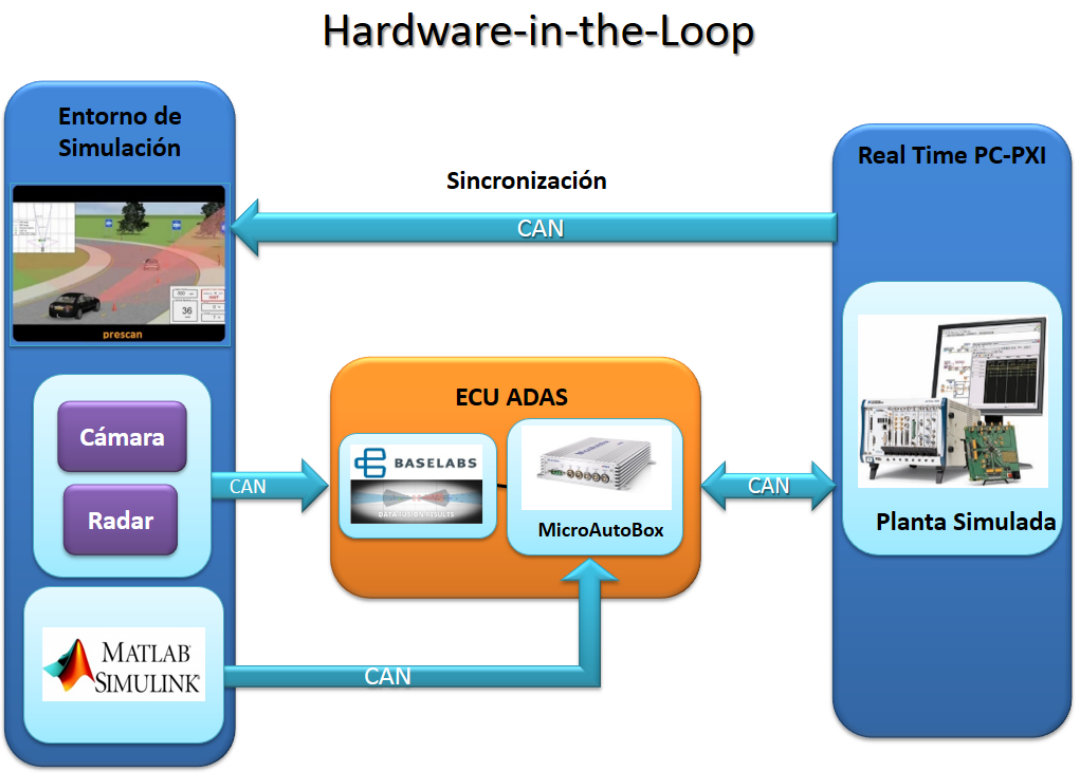


Ilustración 23. Esquema HiL con herramientas utilizadas

A continuación, se procederá a una explicación más detallada de cada parte.

Entorno de simulación:

Ilustración 24. Entorno de simulación

El ordenador de simulación es el PC de escritorio descrito en el apartado 2.3.1. En el correrán los siguientes softwares:

- PreScan
- Simulink

En este ordenador se realizará, el diseño del escenario de simulación, la interpretación de los mensajes en Simulink, y la elección del tipo de conducción, donde se podrá elegir entre:

- Conducción manual
- Conducción autónoma

ECU Prototipo

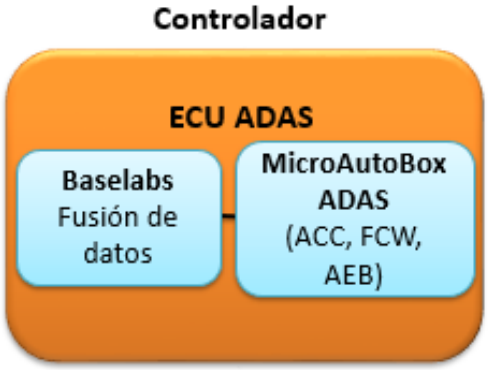


Ilustración 25. ECU Prototipo ADAS

Esta Unidad Electrónica de Control comprende:

- **Car-PC**

En el Car-PC correrá el software de **Baselabs**, donde se realizará la fusión de datos a partir de la información de los mensajes generada en PreScan. Este PC cuenta con un sistema operativo Windows.

- **MicroAutoBox II**

La MicroAutoBox II, como se explicó con anterioridad, es una Unidad Electrónica de Control Prototipo, en ella correrá el modelo Simulink con el algoritmo ADAS.

Actualmente, el ordenador donde corre Baselabs cuenta con un sistema operativo Windows, además la comunicación entre el Car-PC y la MB II se realiza vía Ethernet. Esto lleva asociado un retraso en el procesamiento y envío de la información Esto se debe a que nos encontramos en una fase de prototipo. Sin embargo, en un futuro, la intención sería que la aplicación de fusión de datos esté integrada en un mismo dispositivo junto con el algoritmo de control ADAS que corre en la MBII.

RT-PXI

En el Ordenador en tiempo Real – PXI correrá:

- La planta del vehículo simulada

Ilustración 26. RT PC-PXI Esquema

Al crear un escenario en PreScan, éste genera automáticamente un modelo simulink con la información necesaria acerca de cada vehículo, y del escenario en conjunto.

Cada vehículo disponible en PreScan para simulación cuenta con una planta con los datos reales de éste. A partir de estos datos reales, PreScan genera un modelo de planta con la dinámica del vehículo, la cual cuenta con unos parámetros de entrada, (Steer, Throttle, Brake, Shift), (Volante, acelerador, freno, cambio manual/automático…), y unas salidas, (State, Vx, Parameters), (Posición, Velocidad, Parámetros de salida).

Este modelo de planta será incorporado en el PC-PXI, donde correrá en tiempo real.

- ECUs del Vehículo

En el PC-PXI irán simuladas las diferentes ECUs del vehículo, entre ellas una interfaz HMI desde la cual se podrá activar o desactivar funciones ADAS, así como mostrar la información de interés para el propio conductor, (esta Interfaz HMI no será desarrollada para este TFG).

5. Desarrollo

En este apartado se explicará cómo se ha llevado a cabo la primera fase de este proyecto, la cual abarca las tareas definidas en el apartado anterior.

5.1 Instalación e Integración de equipos

Instalación e **integración** de los distintos equipos hardware y software necesarios.

En esta fase se describe cómo se han configurado e integrado los distintos equipos, para un correcto funcionamiento.

Antes de realizar la instalación e integración del software correspondiente se han tenido en cuenta los siguientes factores:

- Compatibilidad entre las diferentes versiones software.
- Precio
- Reciclaje de las herramientas disponibles en IDIADA.

Se han instalado las siguientes herramientas:

- Matlab y Simulink 2013, de Mathworks, se ha usado esta versión además de por disponer de la licencia, porque incluye las librerías CAN necesarias para enviar y recibir los mensajes CAN.
- PreScan de TASS, el cual se integra con Simulink, de tal manera que cuando se genera un escenario en PreScan, automáticamente, éste crea un modelo en un fichero de Simulink.
- CANalyzer, el software a través del cual serán analizadas las tramas CAN enviadas por Simulink.
- Baselabs, software donde, una vez recibidos los mensajes CAN correspondientes, se realizará la fusión de datos que nos generará la región de incertidumbre, de posible colisión con los obstáculos.

  El sistema de simulación estará integrado de la siguiente manera:

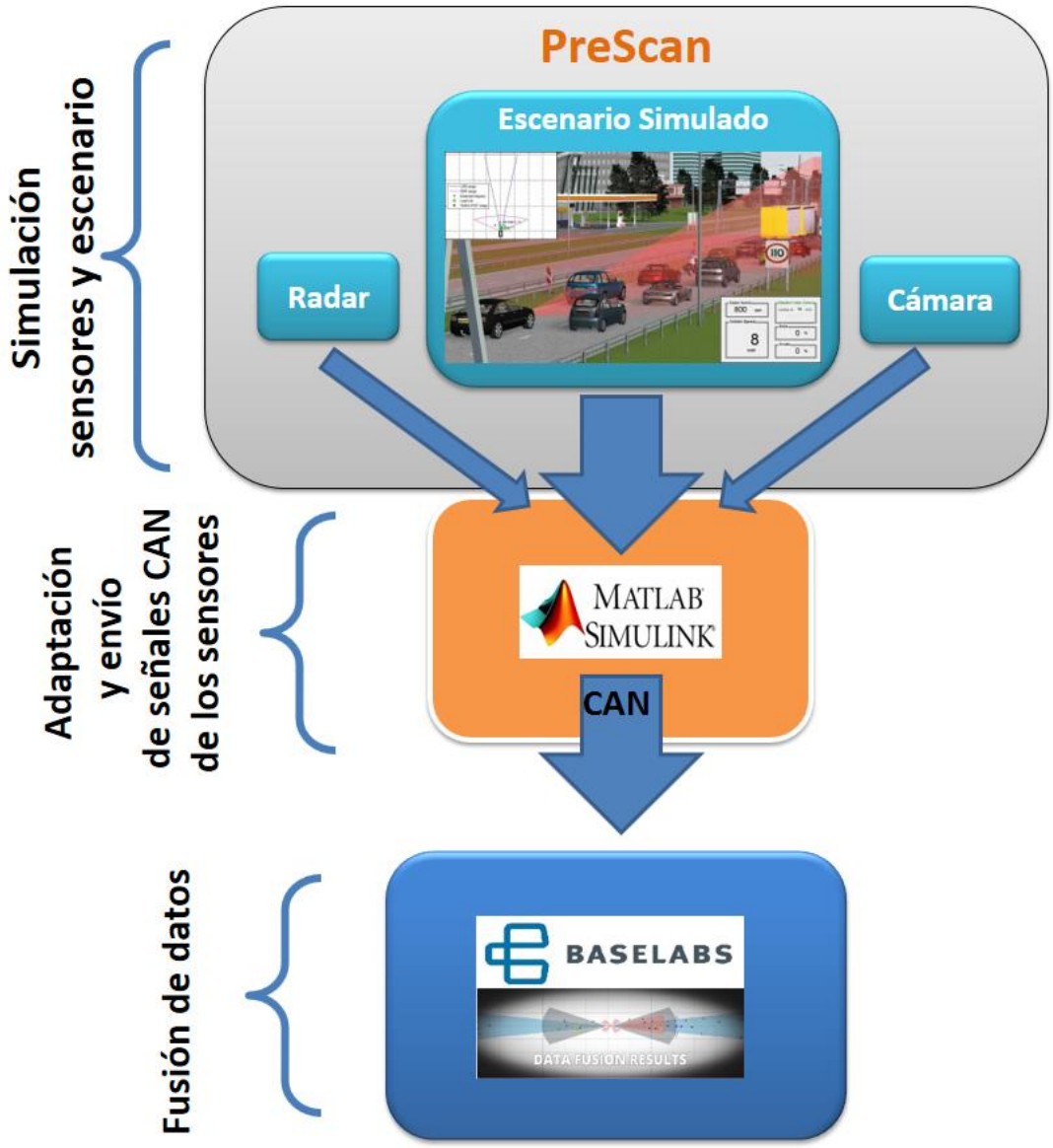


Ilustración 27. Integración Sistema de Simulación

5.2 Simulación de los sensores en el entorno PreScan.

En esta fase se explicará cómo se ha realizado la simulación de los distintos escenarios en el entorno de simulación PreScan.

El software de PreScan tiene dos herramientas:

- El software de desarrollo, **PreScan Editor.**
- El software de visualización, **PreScan VisViewer.**

Primero, abriremos el software de desarrollo, para proceder a la simulación del escenario.

Cuando ejecutamos el PreScan, se muestra la siguiente pantalla:

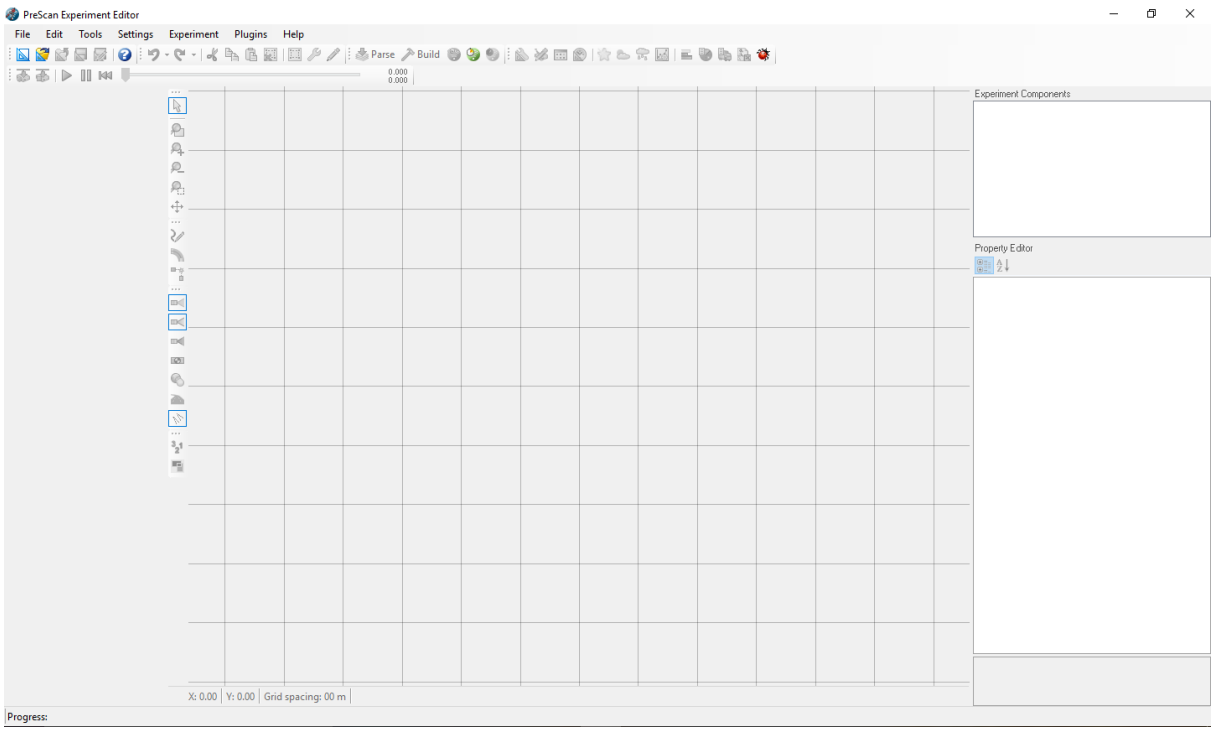


Ilustración 28. Página en blanco PreScan

En esta pantalla podemos observar la siguiente barra de herramientas:

Ilustración 29. Barra de herramientas PreScan

Con la ayuda de ésta, podemos crear un nuevo experimento, donde simularemos el escenario.

PreScan cuenta con una gran variedad de elementos para la simulación, como se puede observar en las siguientes imágenes:

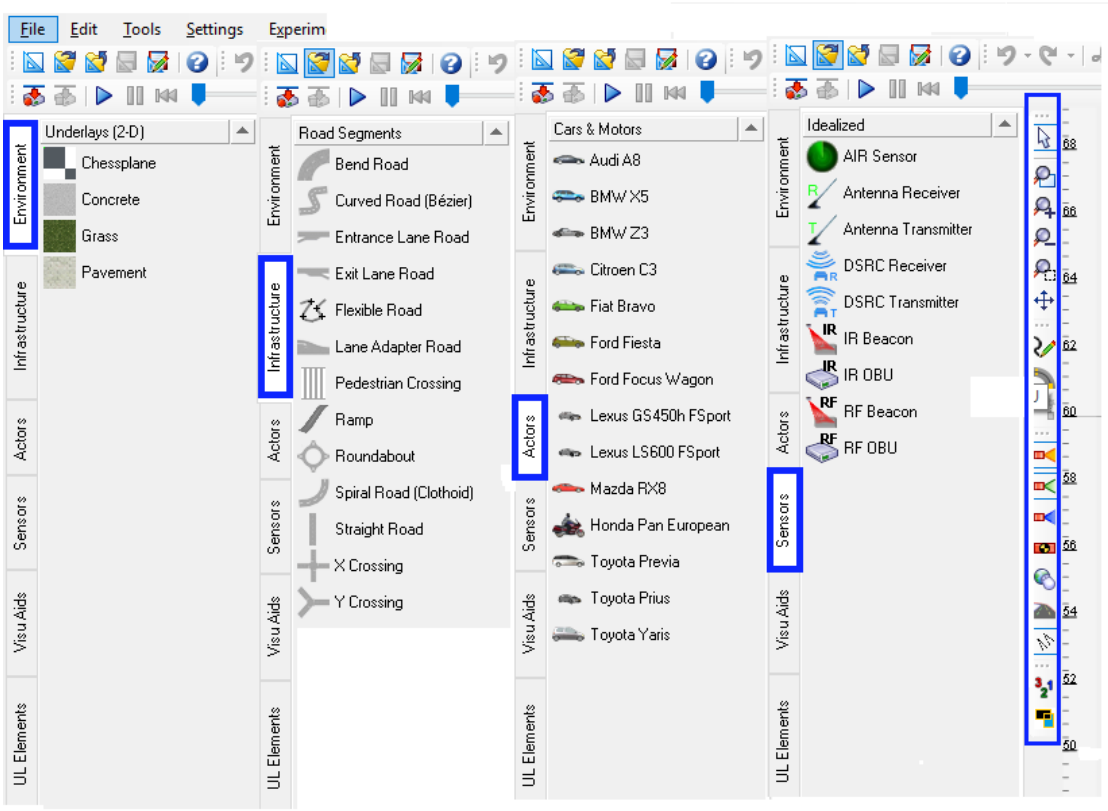


Ilustración 30. Elementos PreScan

Los elementos más importantes que se han utilizado para el desarrollo de este trabajo son:

- Road segments
- Cars & Motors
- Sensors

Ahora se procederá al **diseño del escenario**.

En primer lugar, se crea un nuevo proyecto experimento en blanco, como la de la figura 28, dicho experimento se guardará en la carpeta que se le especifique, es muy importante tener en cuenta el **Path,** ya que PreScan está integrado con Matlab, y para que se comuniquen entre ellos, deben tener el mismo Path.

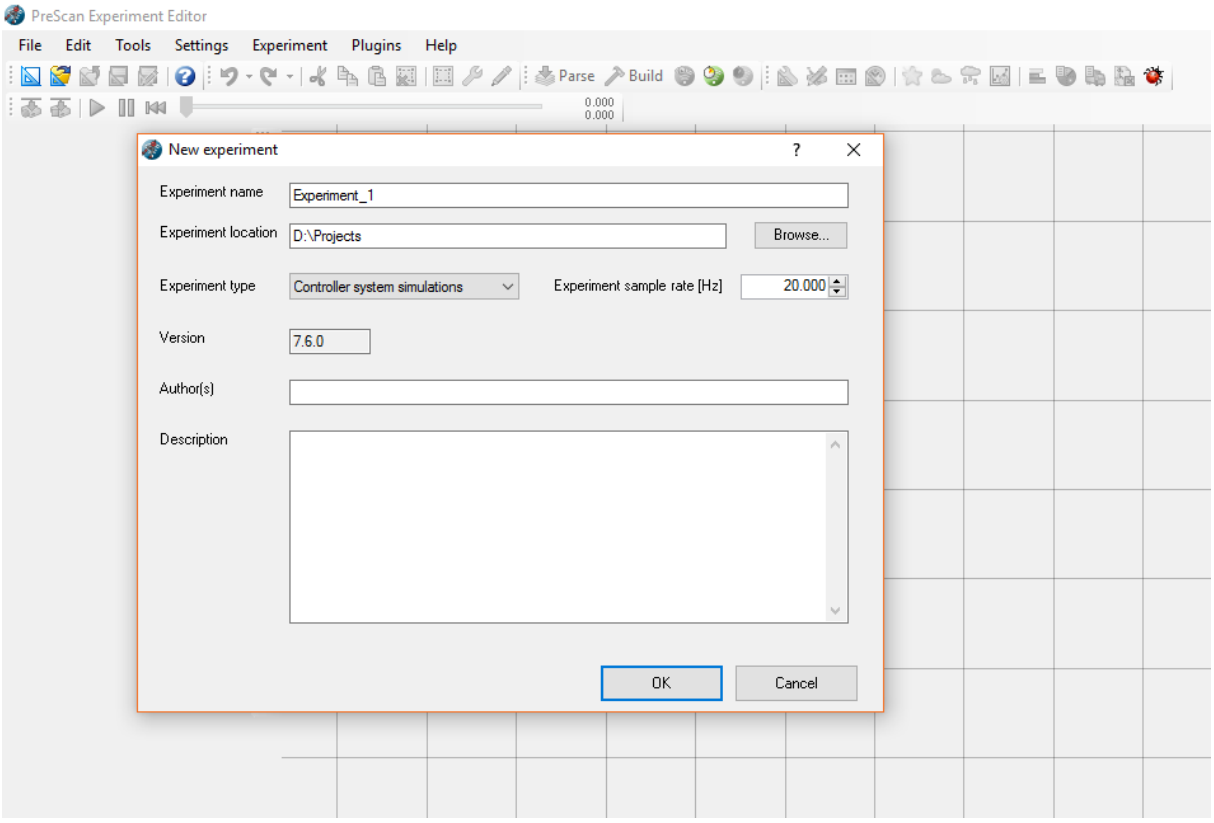


Ilustración 31. Nuevo Experimento PreScan

5.2.1 Infraestructura

El siguiente paso será añadir una infraestructura donde circularán los vehículos, para ello, se selecciona una de la pestaña Road Segments, y se arrastra al mapa.

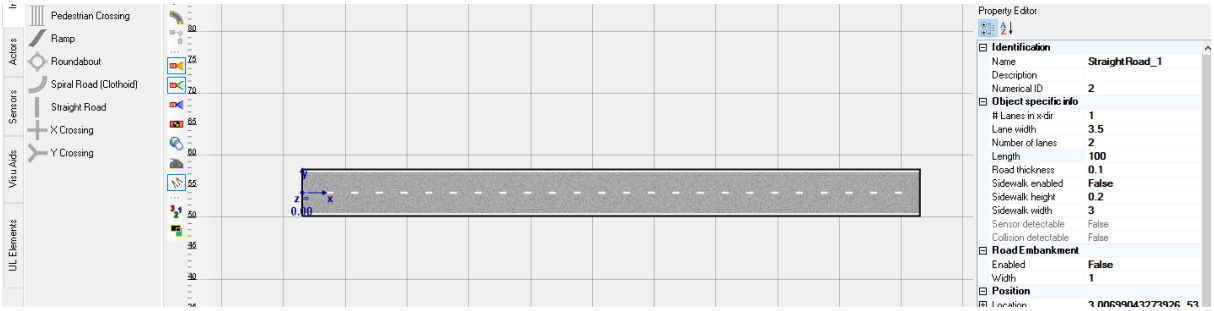


Ilustración 32. Insertar PreScan

Una vez en el mapa, seleccionando la infraestructura podremos modificar sus características a nuestra preferencia.

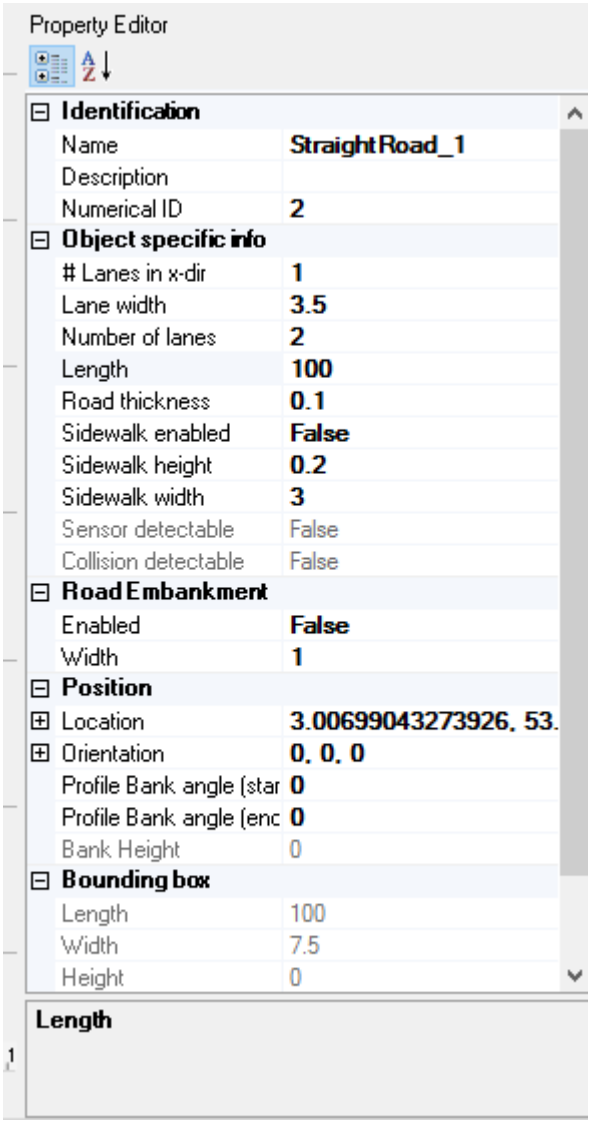


Ilustración 33. Características Infraestructura PreScan

Podremos modificar características como:

- Nombre
- Número de carriles
- Longitud
- Ancho
- Posición
- Orientación

Para que los vehículos puedan circular por ella, y entiendan que se trata de un segmento de carretera, es necesario añadir una trayectoria a dicho segmento de carretera, se puede crear una trayectoria de dos formas:

- Manualmente
- Por tramos

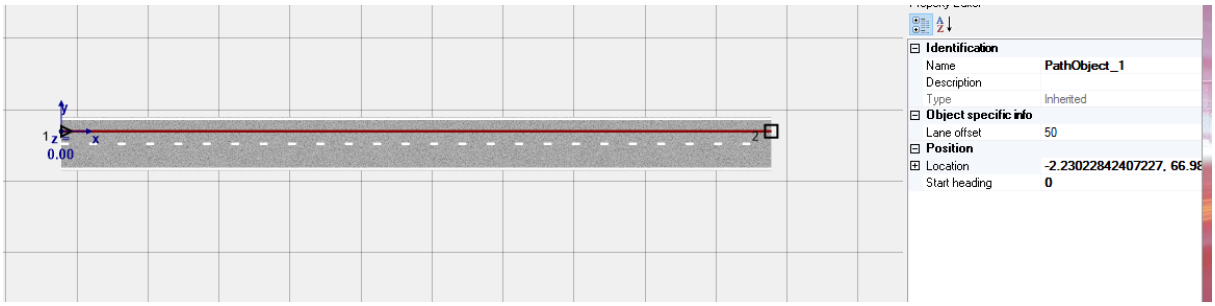


Ilustración 34. Nueva Trayectoria PreScan

Una vez se tiene la carretera, el siguiente paso será añadir a los actores, por actores se entiende cualquier objeto que tenga movilidad:

- Vehículos
- Humanos
- Camiones y autobuses
- Trailers

5.2.2 Vehículo

Para este escenario en concreto, se ha seleccionado el modelo de vehículo **BMW Z3** como el "vehículo anfitrión" (**Host Vehicle**), es decir, el vehículo de referencia para la simulación, el que incluirá los sensores que le proporcionarán la información del entorno. Éste será, por tanto, el punto de partida del proyecto.

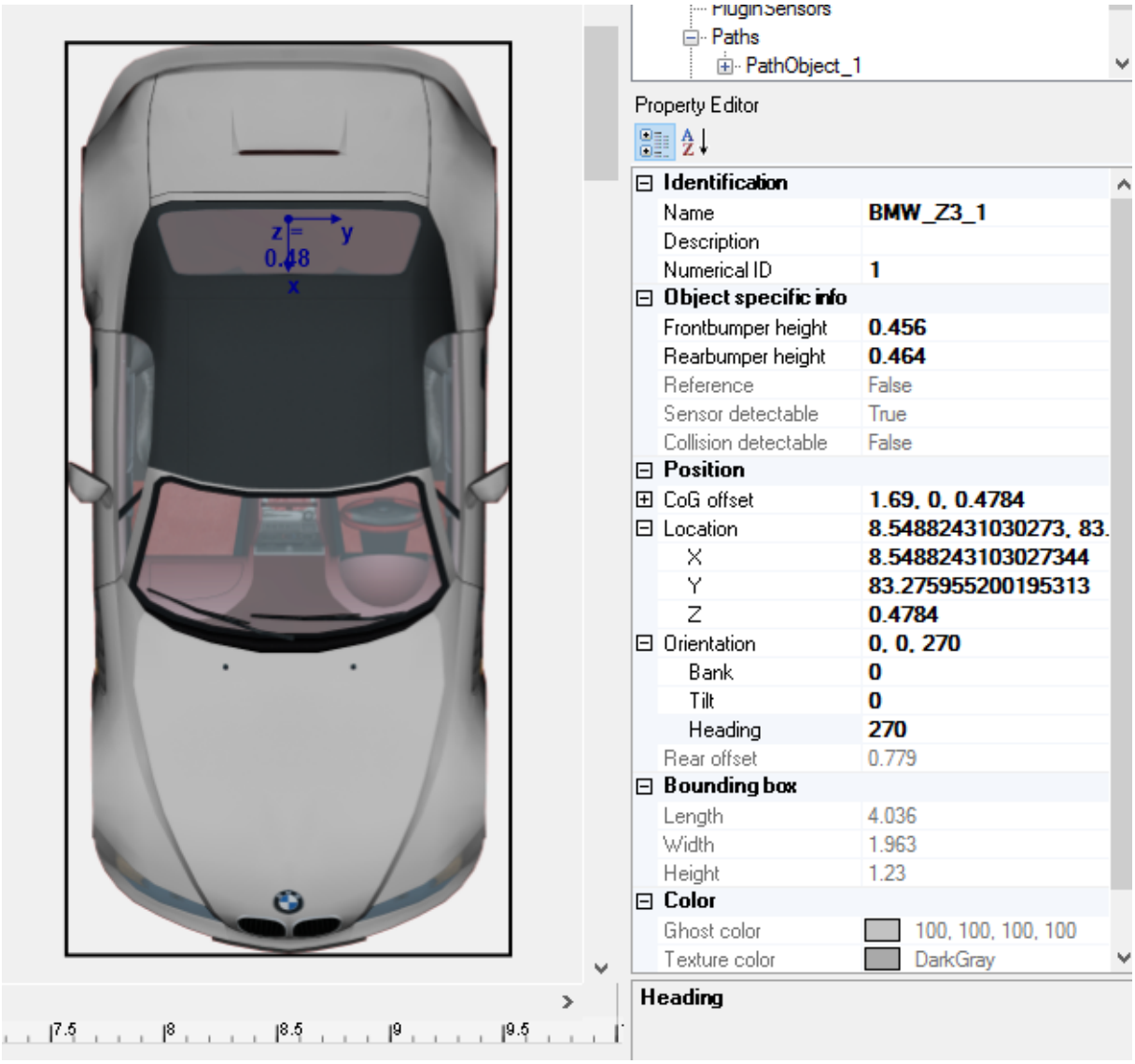


Ilustración 35. Vehículo BMW Z3 PreScan

Para añadir el vehículo al escenario, lo arrastramos sobre la trayectoria previamente diseñada, y el sistema quedará como se muestra:

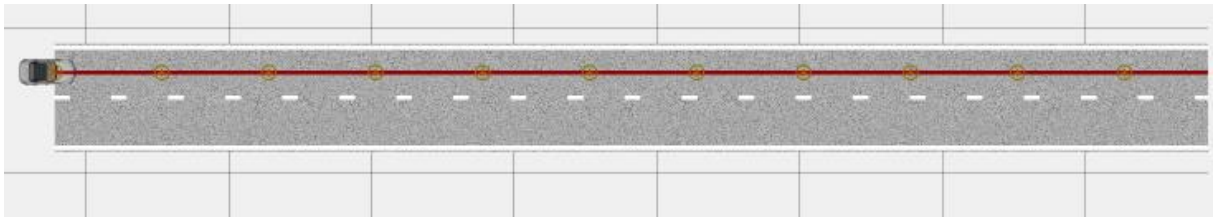


Ilustración 36. Vehículo con trayectoria y perfil de velocidades PreScan

Ahora, una vez el vehículo esté sobre la trayectoria, PreScan generará automáticamente un **perfil de velocidades** (speed profile) por defecto.

En él, se especificará la trayectoria y velocidades que seguirá el **Host Vehicle.**

Para modificar la configuración del vehículo, pinchar botón derecho sobre el vehículo→**Object Configuration.**

Se abrirá una ventana con diferentes opciones acerca del vehículo, entre ellas, el perfil de velocidades. En este perfil se pueden modificar las preferencias del vehículo.

No se entrará en detalle a la explicación detallada de la configuración del vehículo debido a su gran extensión, y a que no son objeto de estudio de este proyecto. Se explicarán exclusivamente las que se han considerado de interés para el lector.

Configuración Trayectoria Vehículo

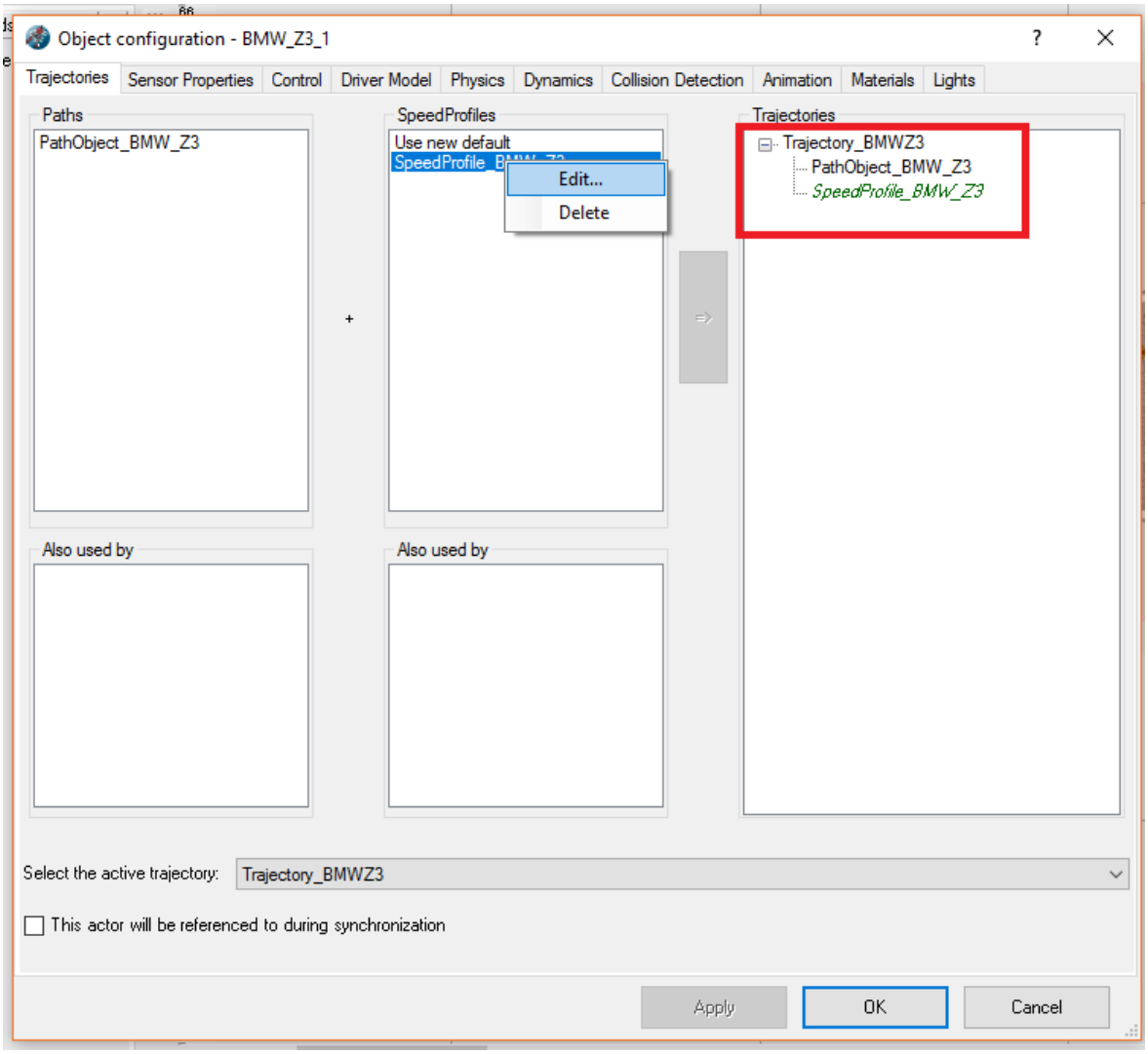


Ilustración 37. Configuración Trayectoria PreScan

Aquí se puede modificar el perfil de velocidades creado por PreScan, como se aprecia en la imagen 38.

Configuración Perfil de Velocidades

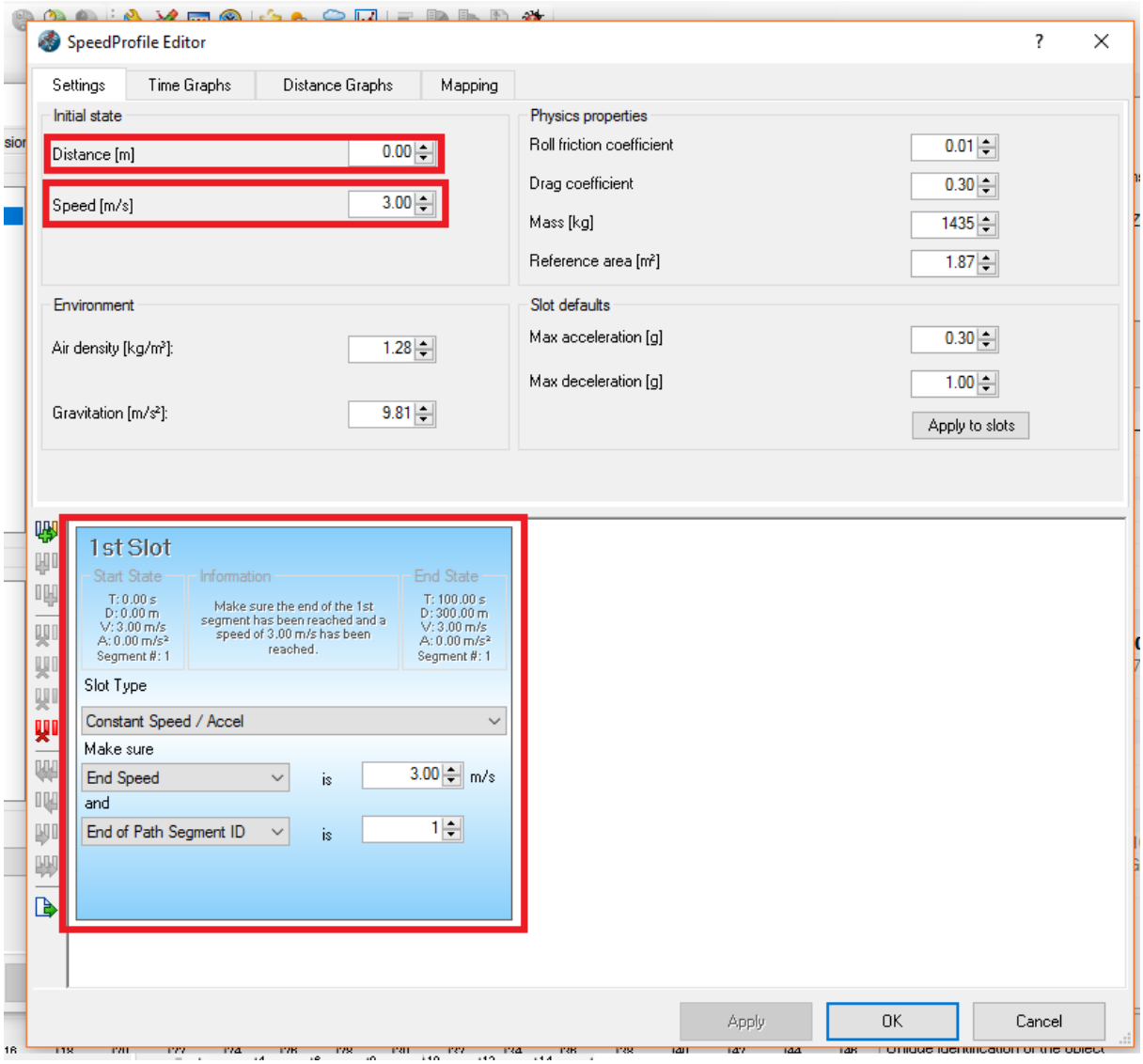


Ilustración 38. Configuración Perfil de velocidades PreScan

Para este ejemplo, se ha establecido una velocidad inicial del vehículo de 3 m/s, y una velocidad final de 3m/s, por lo tanto, se tendrá una aceleración nula, se ha realizado de este modo para comprobar más adelante la precisión de los sensores.

Configuración Modelo de conductor

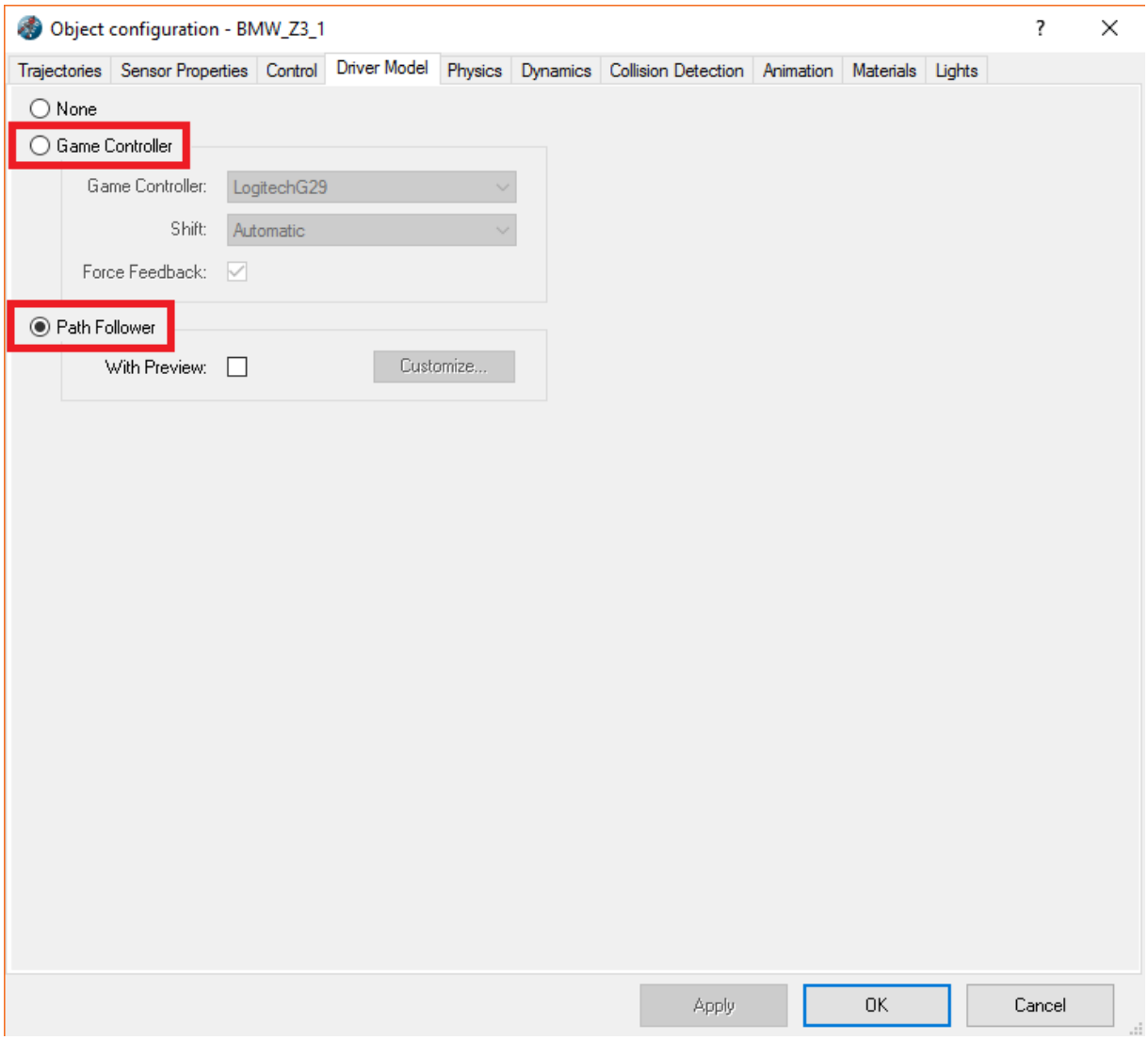


Ilustración 39. Configuración Tipo de conducción

En esta pestaña se puede seleccionar el modo de conducción, nos encontramos tres opciones:

- Ninguno (None)
- Modo conductor (Game Controller)

  En esta opción, el vehículo será controlado desde un simulador hardware externo, que permitirá controlar el volante (steer) y los pedales de acelerador y freno (throttle and brake).

- Seguidor de camino (Path Follower)

En este modo de conducción, el vehículo será conducido de manera autónoma, esta opción será útil para la fase de automatización de las pruebas de validación, al final del proyecto.

Configuración Modelo Dinámico del Vehículo

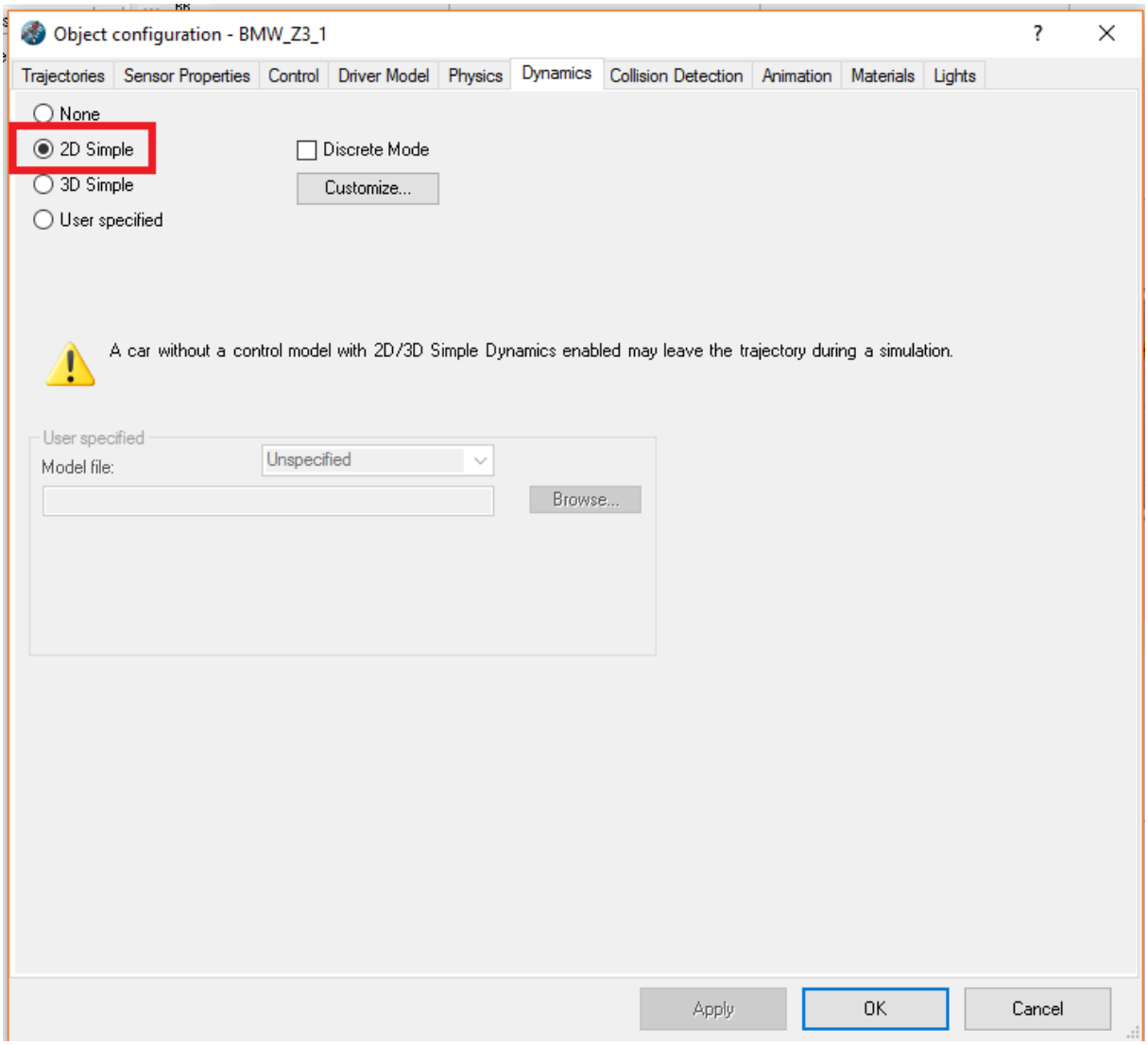


Ilustración 40. Configuración modelo dinámico del vehículo

Esta opción generará un modelo dinámico del vehículo, dependiendo de la opción seleccionada:

- Ninguno (None)
- 2D →Se ha seleccionado éste, ya que para esta simulación no se considerará el **eje Z**.
- 3D

Este modelo dinámico simula el comportamiento del vehículo.

A continuación, se añadirán los sensores. Para este proyecto se utilizarán una cámara y un radar, al igual que los que lleva incorporados el vehículo real a validar.

5.2.3 Sensores

Hay una gran cantidad de sensores, cada uno ofrece un tipo información acerca del entorno. El objetivo de este TFG es simular las señales y mensajes que envían el radar y la cámara incorporados en el vehículo real, por lo tanto, en este trabajo, se han elegido una cámara y un radar acordes a los sensores que se desea simular.

Se procederá primero a la descripción del radar simulado incorporado en el vehículo.

Radar TIS

Ilustración 41. Radar PreScan

Para incorporar el radar de PreScan en el modelo vehículo BMW Z3, se selecciona y se arrastra sobre el coche. Para configurarlo, se accede presionando botón derecho y accediendo al menú object configuration, en el, se puede modificar las características del radar.

Configuración de posición y orientación (Radar)

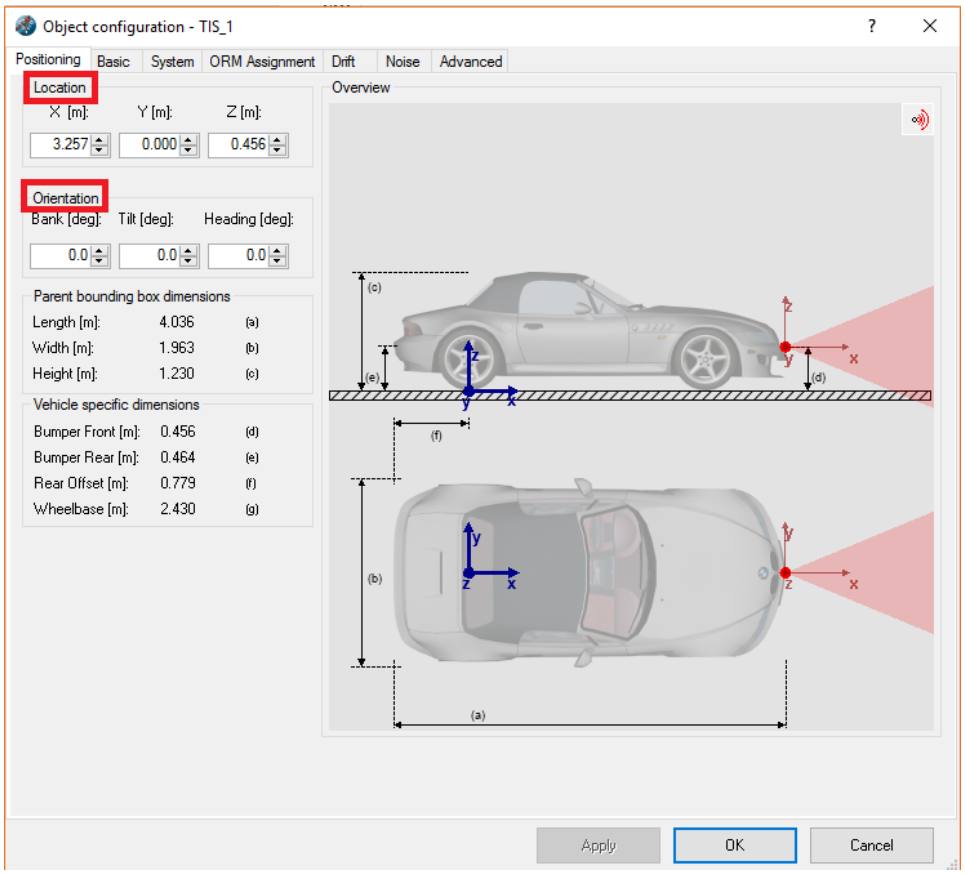


Ilustración 42. Configuración de posición y orientación Radar

Posición y Orientación

En esta pestaña se establece la posición y orientación del radar, acorde a cómo está configurado en el vehículo.

Posición:

- X(m): 3,257
- Y(m): 0,000
- Z(m): 0,456

Orientación:

- Cabeceo (grados): 0,0
- Alabeo (grados): 0,0
- Guiñada (grados): 0,0

Configuración Básica (Radar)

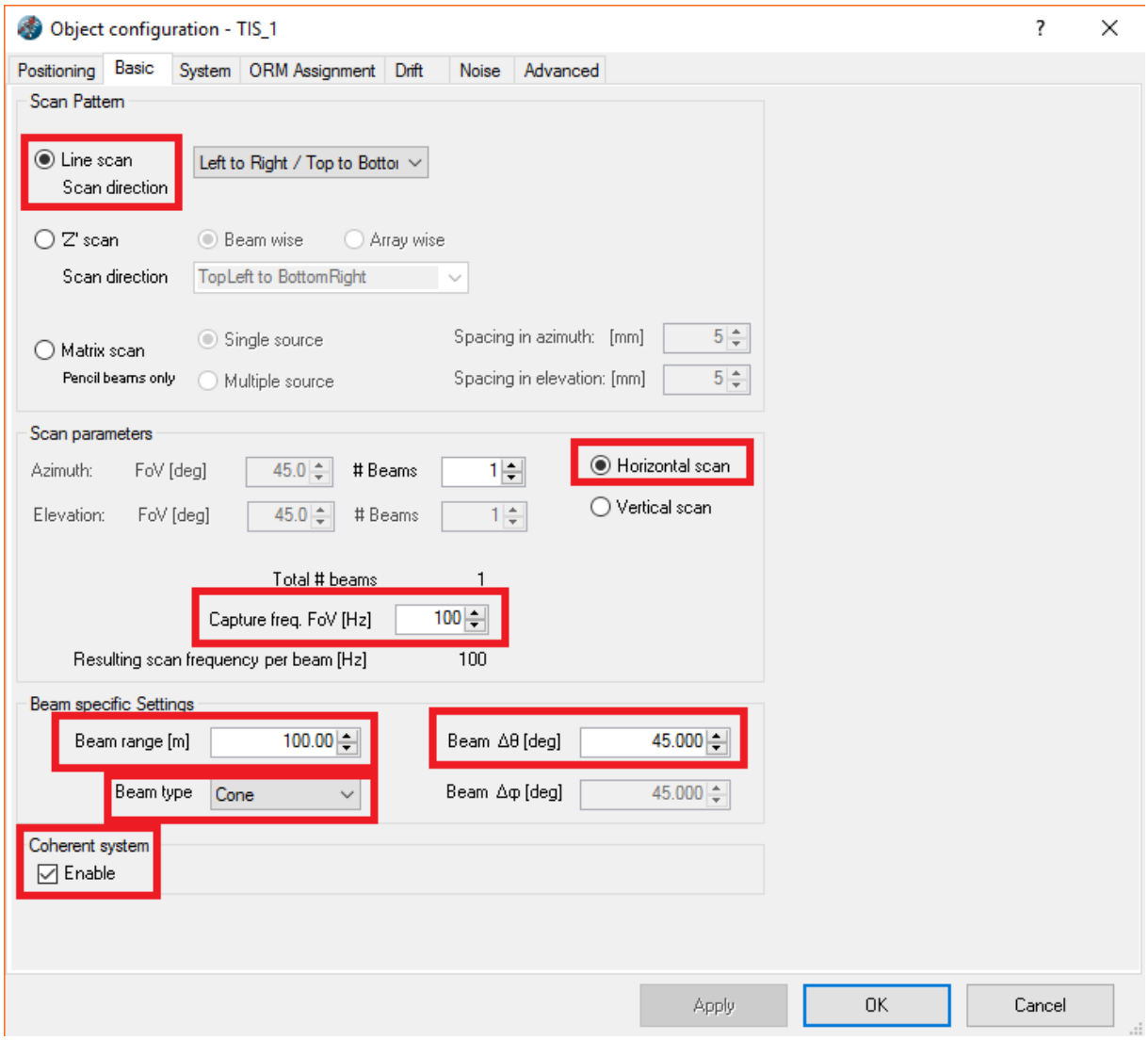


Ilustración 43. Configuración Básica (Radar)

- Parámetros

- Tipo de escáner: Escáner de línea de izquierda a derecha
- Escáner Horizontal
- Frecuencia de la captura del campo de visión (Field of Vision): 100 Hz
- Alcance (m): 100
- Grado de visión (grados): 45
- Tipo de haz: Cónico
- Sistema coherente: On

Configuración Sistema (Radar)

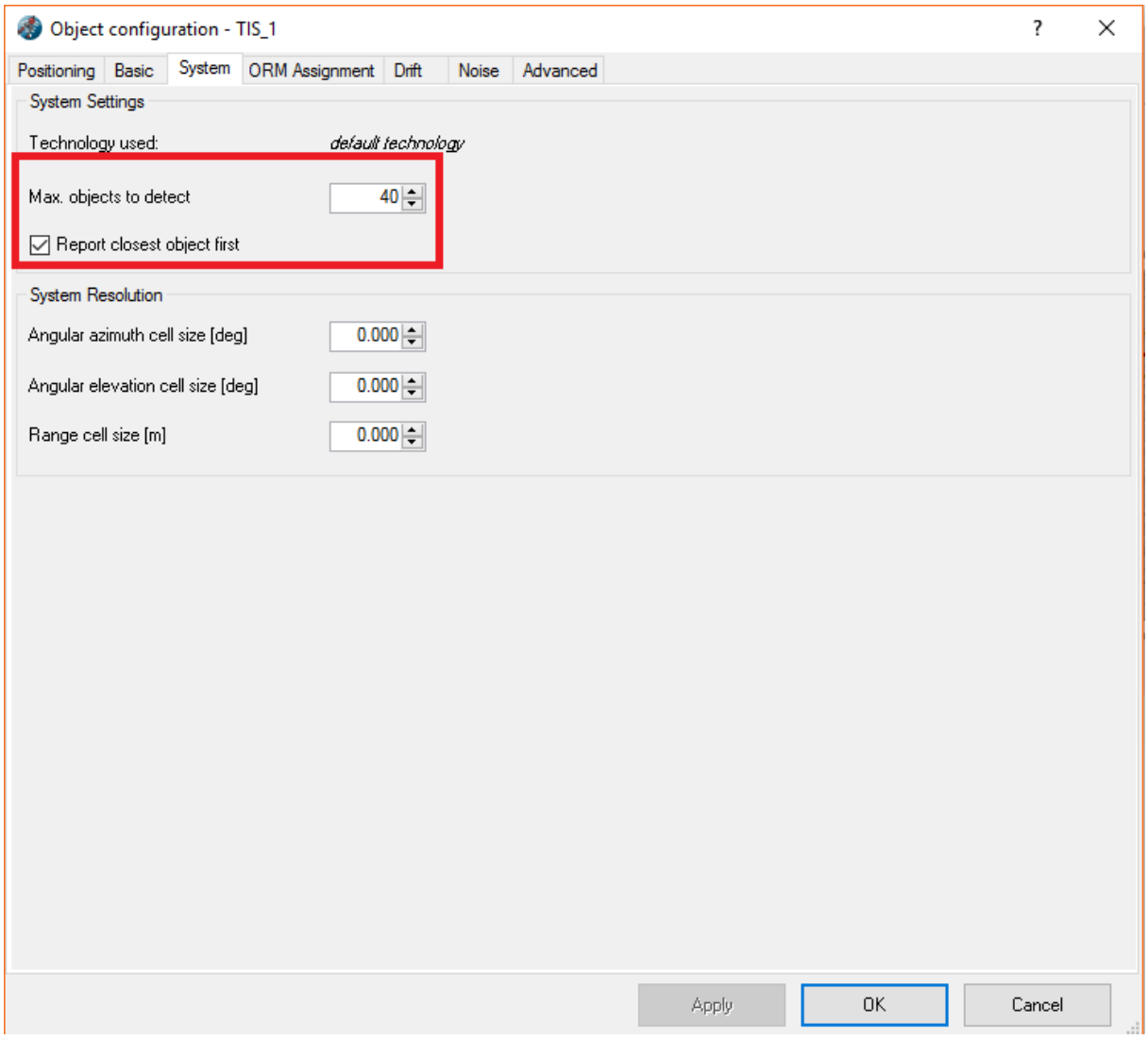


Ilustración 44. Configuración Sistema (Radar)

Tecnología empleada:

- Número máximo de objetos a detectar: **40**
- Reportar primero los obstáculos más cercanos: **On**

Estas dos preferencias con importantes a considerar, ya que el radar real que se desea simular reporta **40 objetos por cada barrido** (frame).

La opción de reportar los objetos cercanos primero nos facilitará calcular cual es el **obstáculo con mayor prioridad** para nuestro sistema ADAS.

Las demás pestañas de configuración se dejarán por defecto y no serán descritas en este TFG, por no ser objeto de estudio para este proyecto.

Una vez configurado el radar acorde a nuestros objetivos, PreScan generará un bloque en Simulink, este bloque será a través del cual el sensor nos facilitará la información que adquiere del entorno.

Procederíamos ahora a la configuración de la cámara.

Cámara

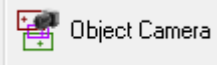


Ilustración 45. Object Camera Sensor PreScan

La cámara se llama **ObjectCameraSensor.**

Al igual que se hizo con el radar y los demás elementos, será necesario modificar su configuración acorde a nuestros requisitos.

Configuración del sistema

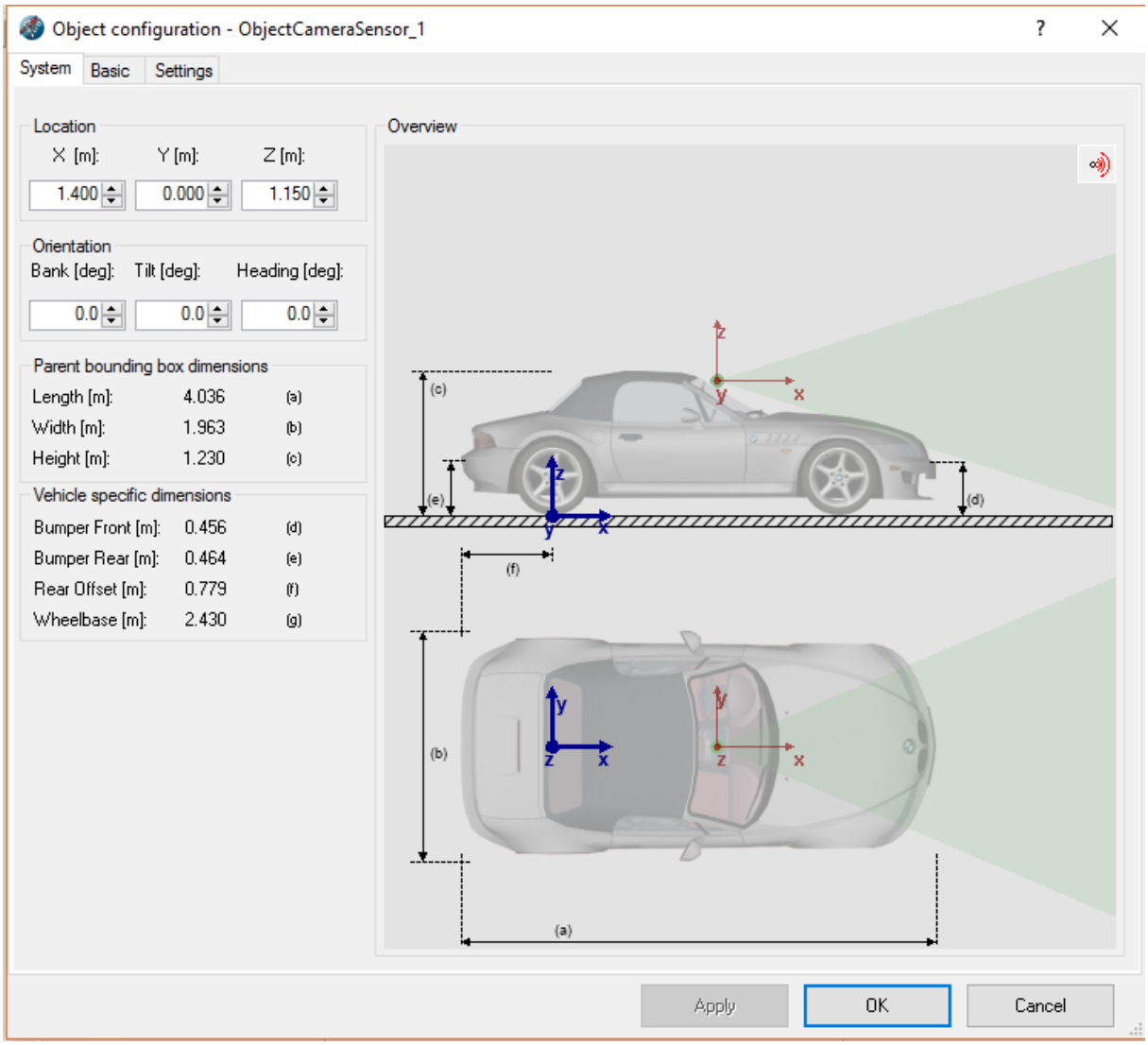


Ilustración 46. Configuración del sistema

En esta pestaña se establece la posición y orientación de la cámara, acorde a cómo está configurado en el vehículo.

Posición:

- X(m): 1,400
- Y(m): 0,000
- Z(m): 1,150

Orientación:

- Cabeceo (grados): 0,0
- Alabeo (grados): 0,0

- Guiñada (grados): 0,0

Configuración básica

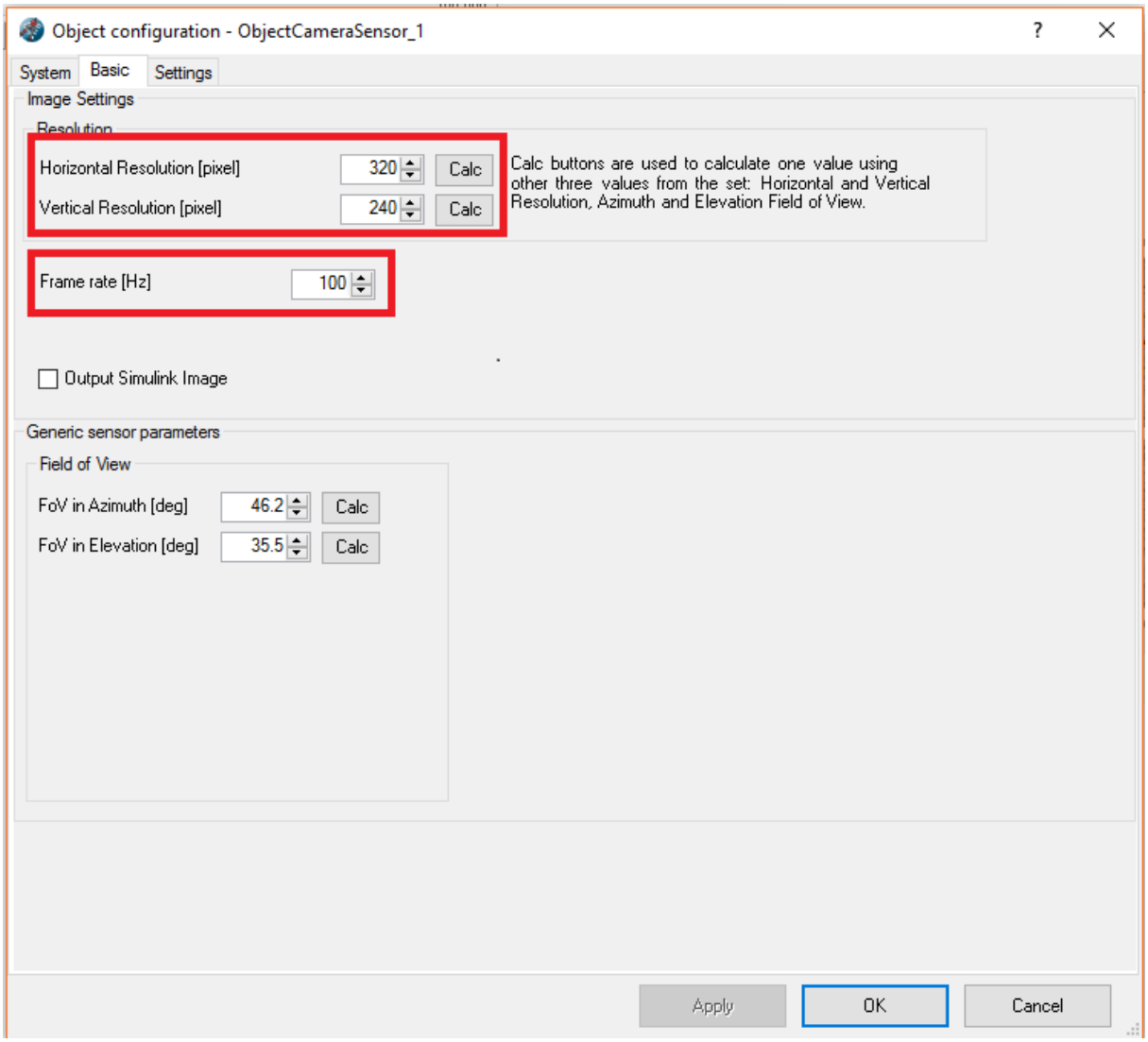


Ilustración 47. Configuración básica

Resolución de la cámara:

- Horizontal (pixel): 320
- Vertical (pixel): 240
- Frecuencia (Hz): 100

La frecuencia será la misma que la del radar, una frecuencia de 100 Hz, que equivale a 10 milisegundos, frecuencia superior a la del barrido de los sensores reales, para evitar la pérdida de mensajes.

Configuración de ajustes

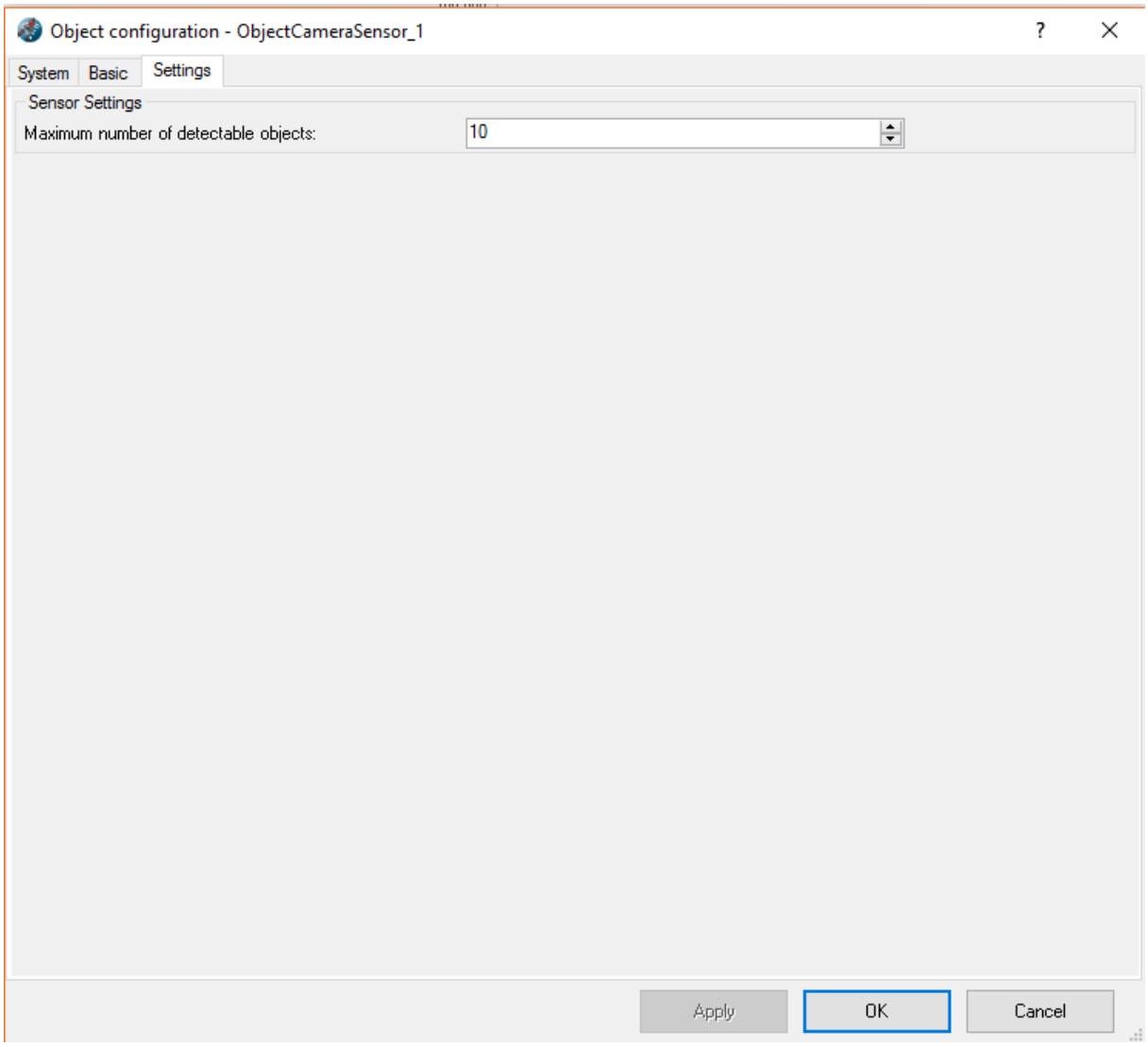


Ilustración 48. Configuración de ajustes

En esta ventana se selecciona el número de obstáculos que se desea detectar, 10 según las especificaciones de nuestra cámara.

Al igual que con el radar, PreScan, una vez el modelo sea construido, generará un bloque en Simulink con la información que recibe la cámara.

Finalmente, el vehículo con los sensores quedaría como tal:

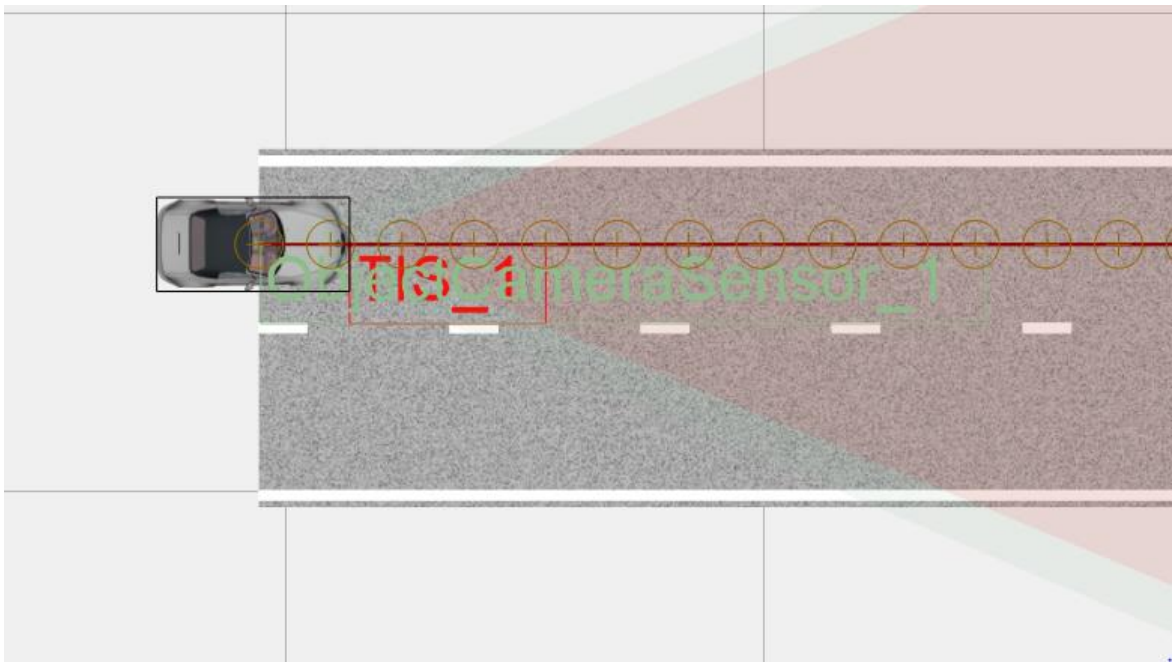


Ilustración 49. Vehículo con sensores incorporados

Se añadirán algunos vehículos adicionales que se encuentren en el campo de visión de los sensores, que nos servirán como obstáculos para nuestro Host vehículo.

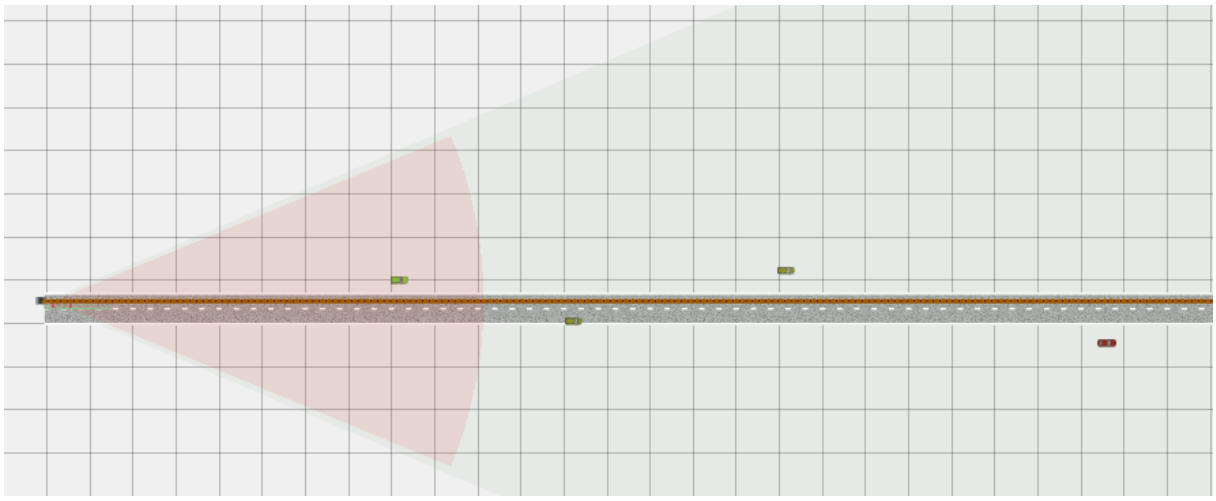


Ilustración 50. Escenario de simulación

Por último, se establecerá un reloj que programe, por una parte, la frecuencia a la que se comunicará con Simulink, y, por otra parte, la frecuencia a la que se actualizará la interfaz donde se reproducirá el escenario.

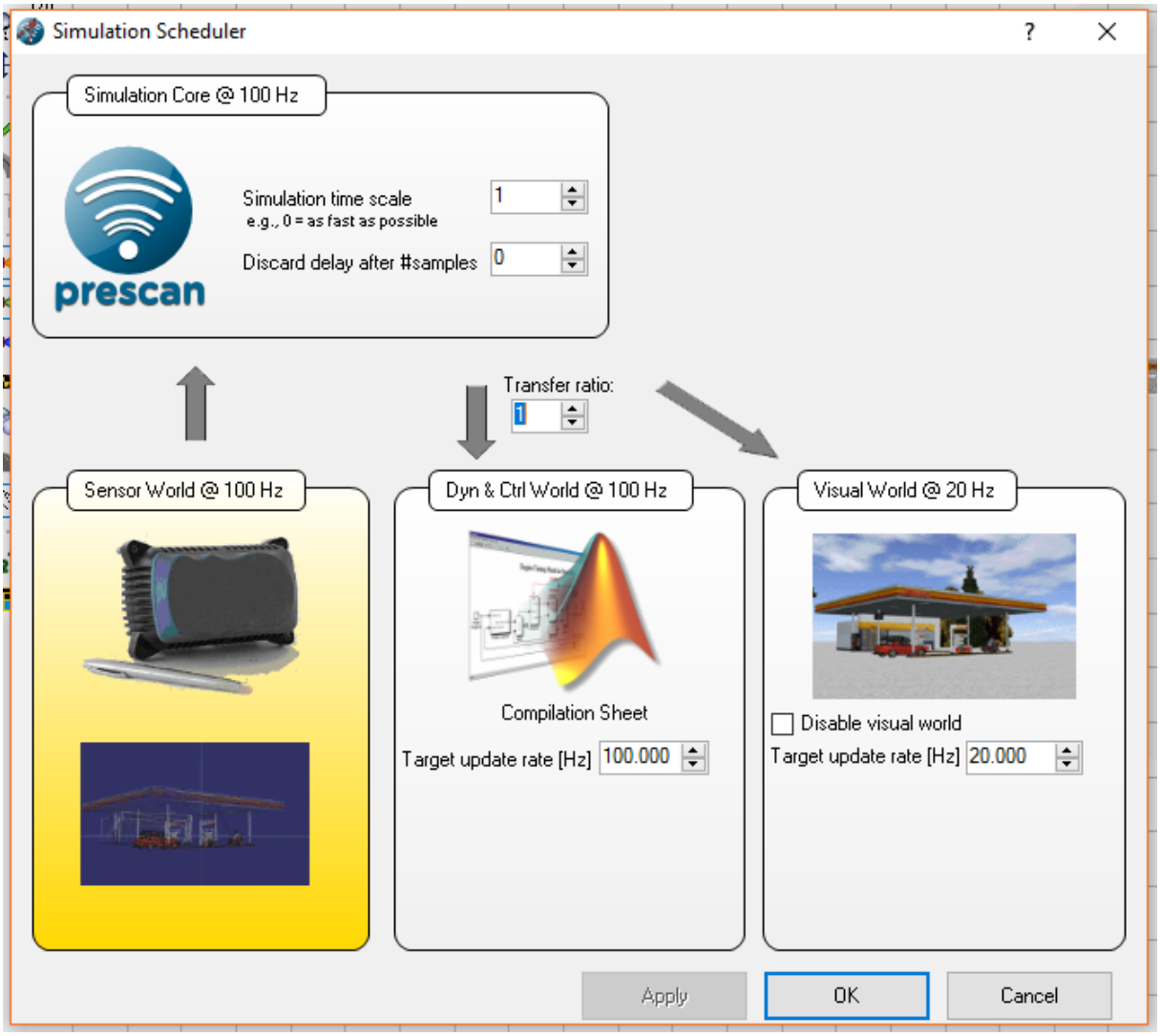


Ilustración 51. Simulation Scheduler PreScan

Un aspecto a tener en cuenta es, que la frecuencia del modelo simulink, siempre deberá ser mayor que la del entorno visual, ya que si fuera al contrario, el entorno podría estar perdiendo información de las funciones que se realizan en el modelo y podría haber conflictos.

Una vez se haya simulado el escenario y los sensores, y se hayan establecido las frecuencias convenientes, el siguiente paso será construir el modelo. Esta función generará programáticamente un modelo simulink con la información sobre el escenario diseñado, y al mismo tiempo, abrirá una interfaz donde se podrá visualizar el comportamiento que tendrá el escenario creado cuando se proceda a su reproducción.

Para llevar esto a cabo, se pincha sobre el icono que pone Parse sobre la barra de herramientas.

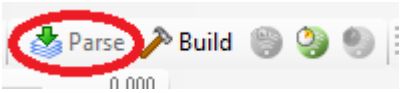


Ilustración 52. Build PreScan

Esta opción realizará un análisis sobre el experimento y nos informará de si hay algún error o aviso importante.

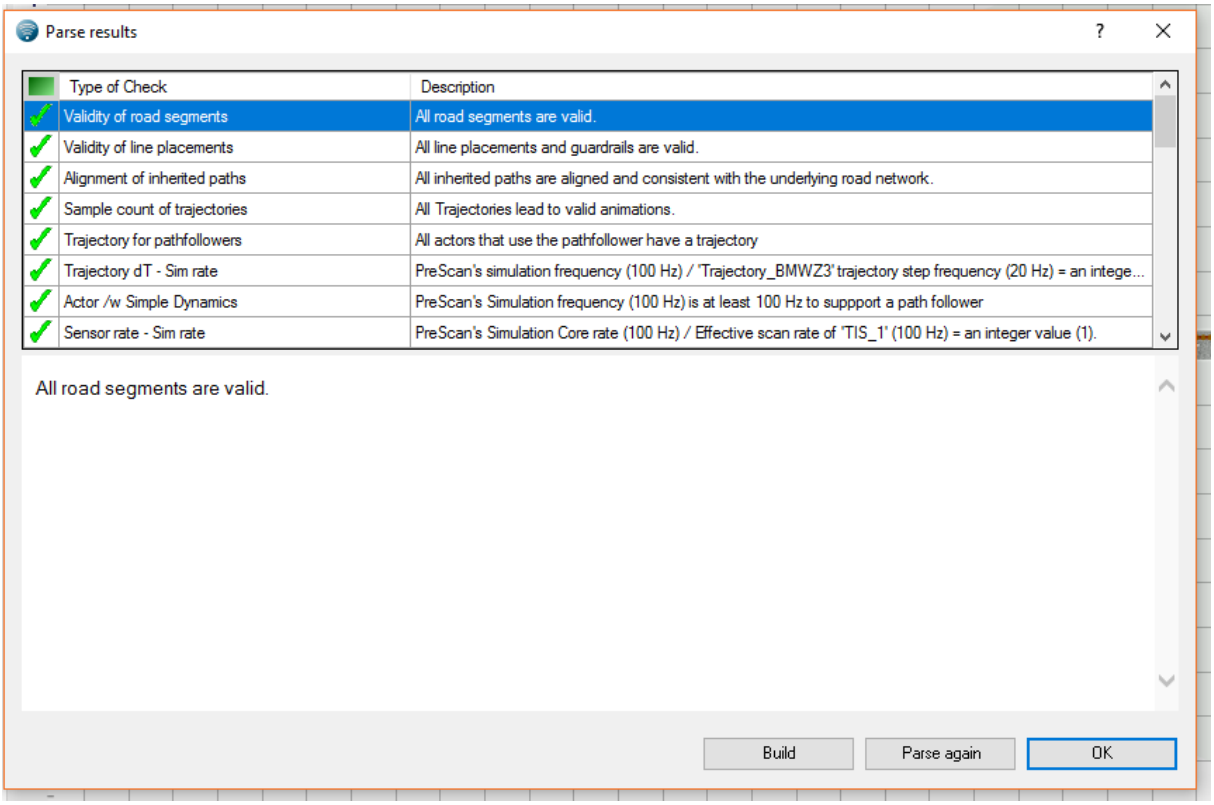


Ilustración 53. Resultados Análisis

Como podemos ver, todos los elementos han superado el análisis y podemos proceder a la construcción.

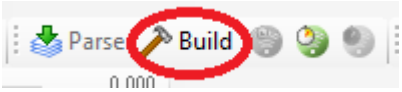


Ilustración 54. Build PreScan

Como hemos dicho anteriormente, ahora PreScan genera:

- Un modelo Simulink
- Una interfaz visual

Aquí podemos ver el modelo simulink generado en Matlab.

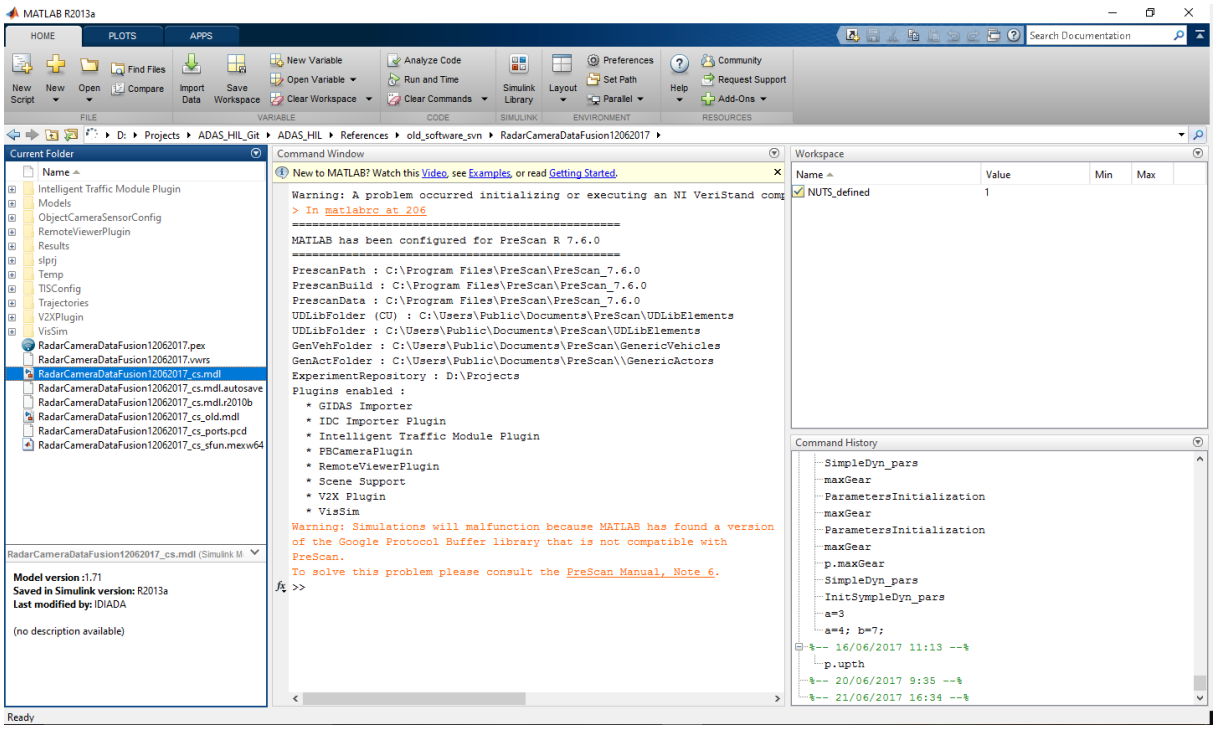


Ilustración 55. Matlab

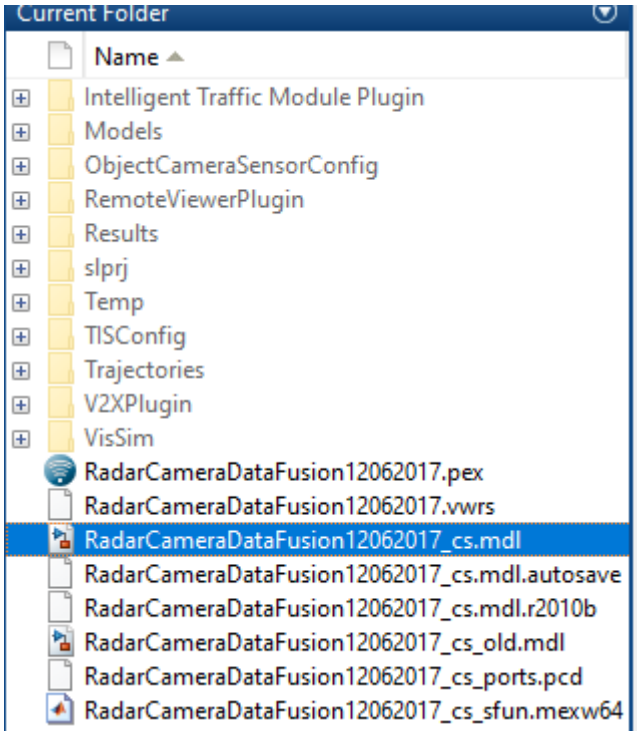


Ilustración 56. Modelo Simulink generado por PreScan

Y la interfaz visual donde se reproducirá el escenario:

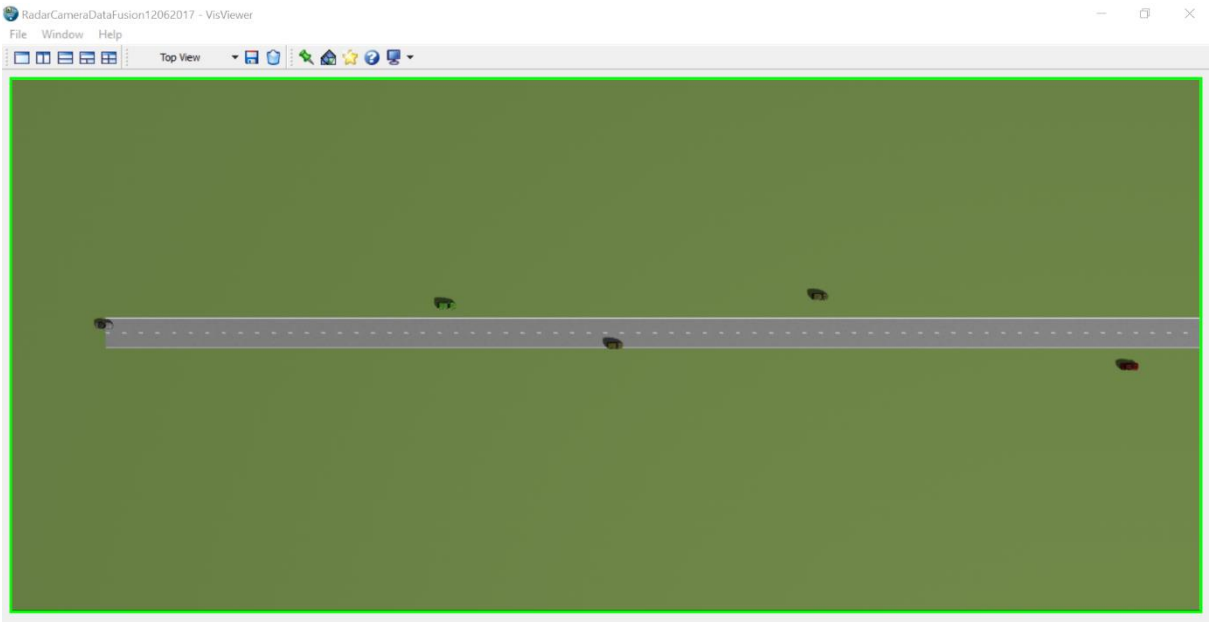


Ilustración 57. Interfaz VisViewer PreScan

A continuación, ejecutaremos el modelo simulink, y se nos abrirá una ventana como la siguiente:

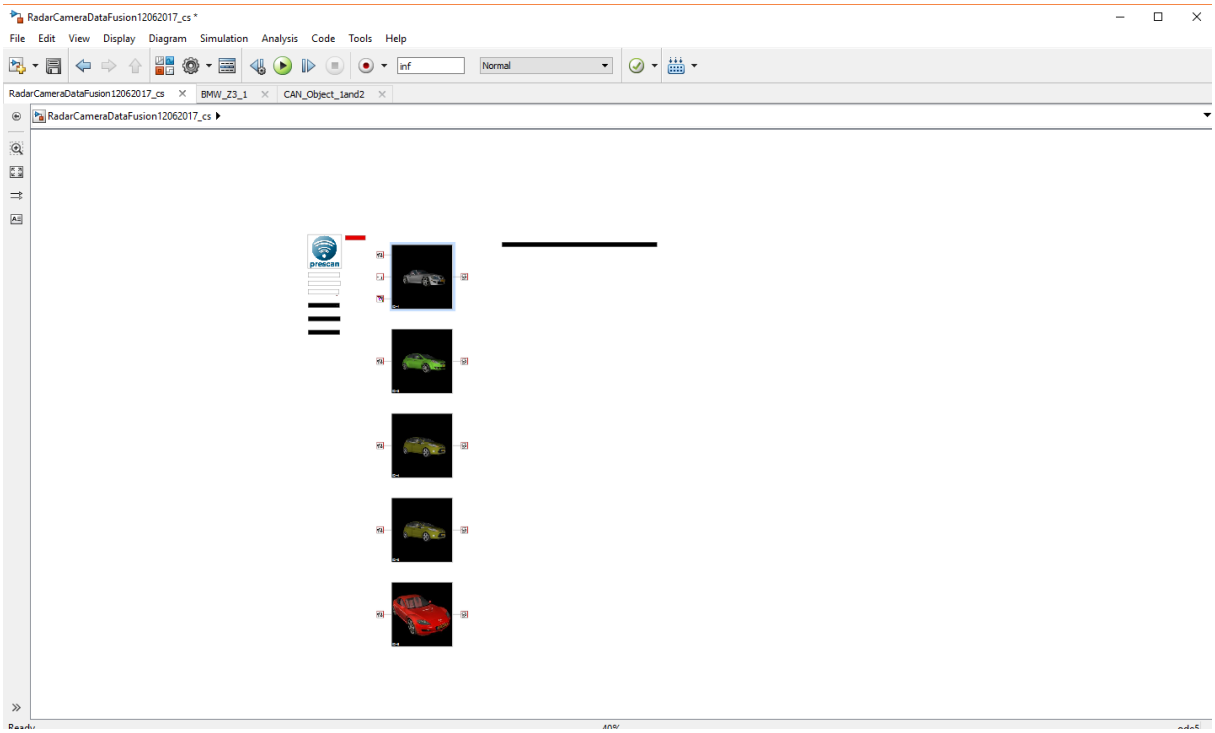


Ilustración 58.Modelo Simulink abierto

En ella, podemos ver, cómo se ha generado un bloque específico para cada vehículo incluido en el escenario.

Nos centraremos en el BMW_Z3, ya que será el vehículo bajo prueba simulado.

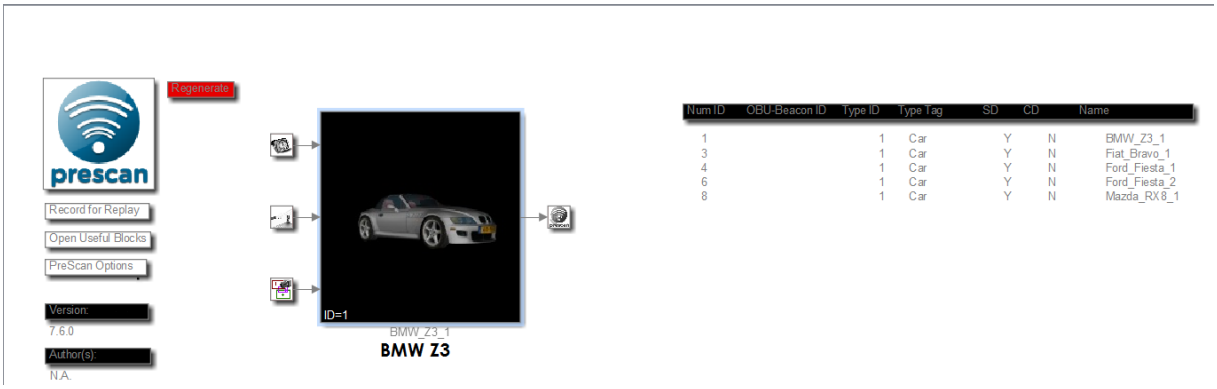


Ilustración 59. Bloque BMW Z3 en Simulink

Si entramos dentro de la configuración del BMW Z3, nos encontraremos con lo siguiente.

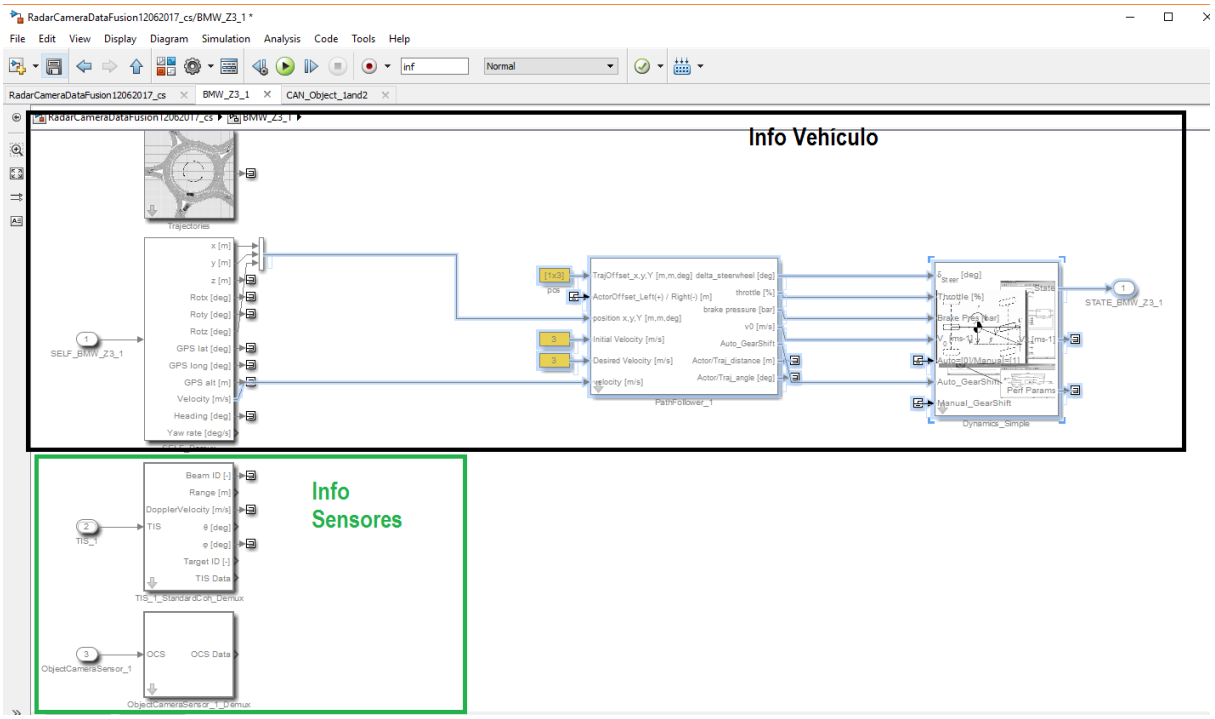


Ilustración 60. Modelo BMW Z3 en Simulink generado por PreScan

Este es el modelo creado por PreScan, a continuación, se analizará cada bloque de manera independiente.

Información vehículo

Ilustración 61. Trayectoria

Este bloque contiene la información de la trayectoria del vehículo.

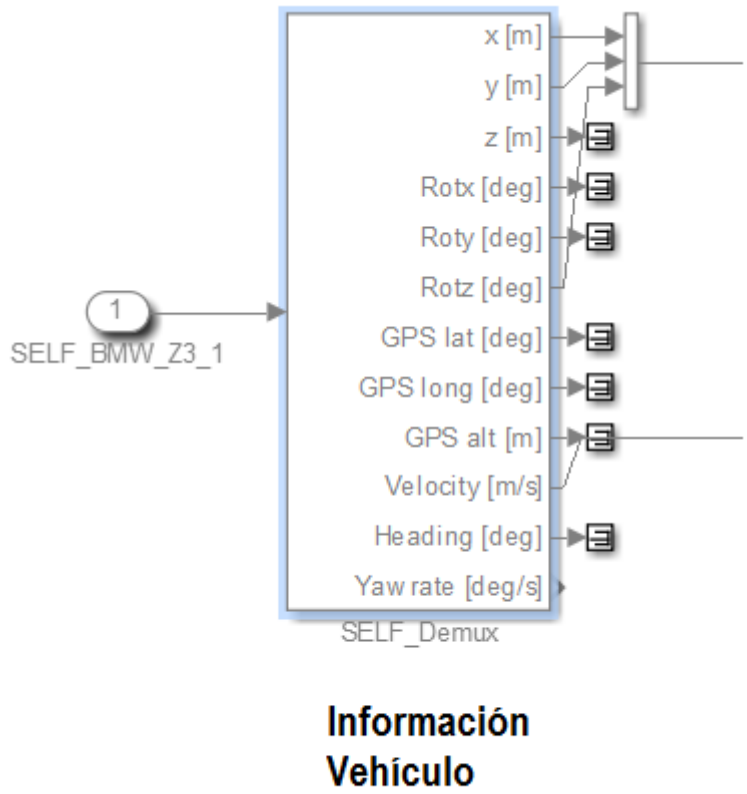


Ilustración 62. Información Vehículo

Este bloque ofrece información sobre la situación actual del vehículo. La función de este bloque, además de informar al usuario, es enviar ciertos mensajes como entrada al bloque del modelo dinámico del vehículo.

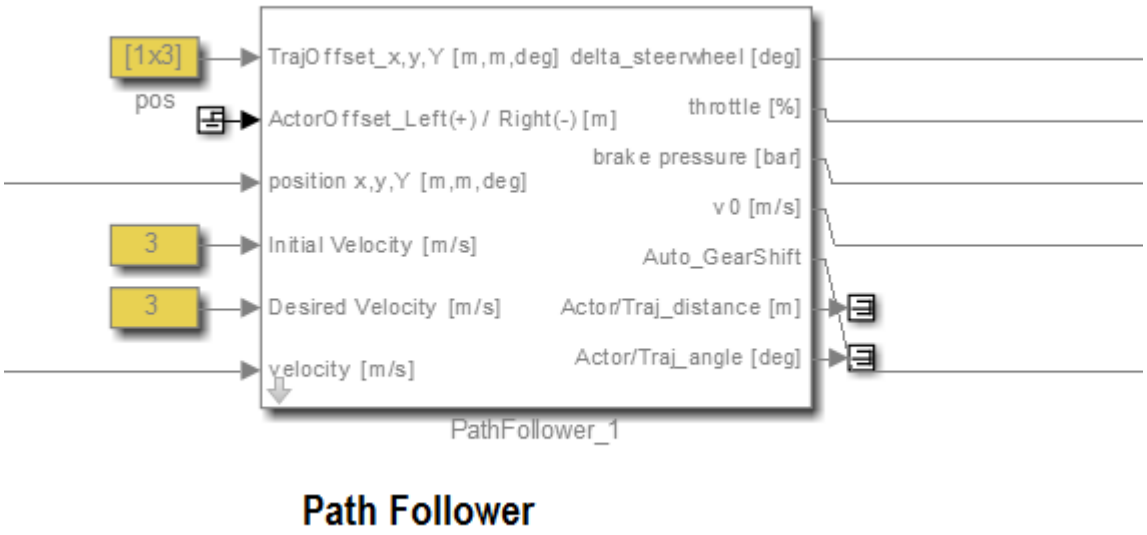


Ilustración 63. Modo seguidor de trayectoria

Este bloque es generado al seleccionar el modo de conducción automático en PreScan, como se mostró en la imagen 39.

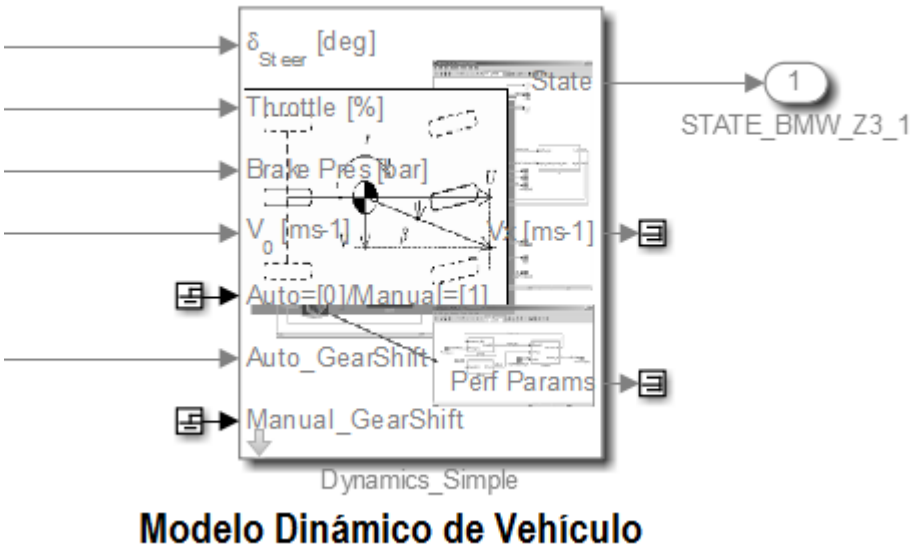


Ilustración 64. Modelo dinámico del vehículo

Este es el bloque simula el comportamiento del vehículo, es la **planta simulada** de nuestro sistema. Tiene unos parámetros de entrada, a partir de los cuales genera unas salidas, sobre el estado del vehículo.

Entradas:

Steer [deg	Posición del volante en grados
Throttle [%]	Acelerador
Brake [bar]	Freno
V0 [m/s]	Velocidad inicial
Auto/Manual [0/1]	Cambio automático o manual
Auto Gear [0/1]	Cambio de automático
Manual Gear [0/1]	Cambio manual

Tabla 2

Entrando en el modelo dinámico, se puede observar el algoritmo de ´la planta del vehículo.

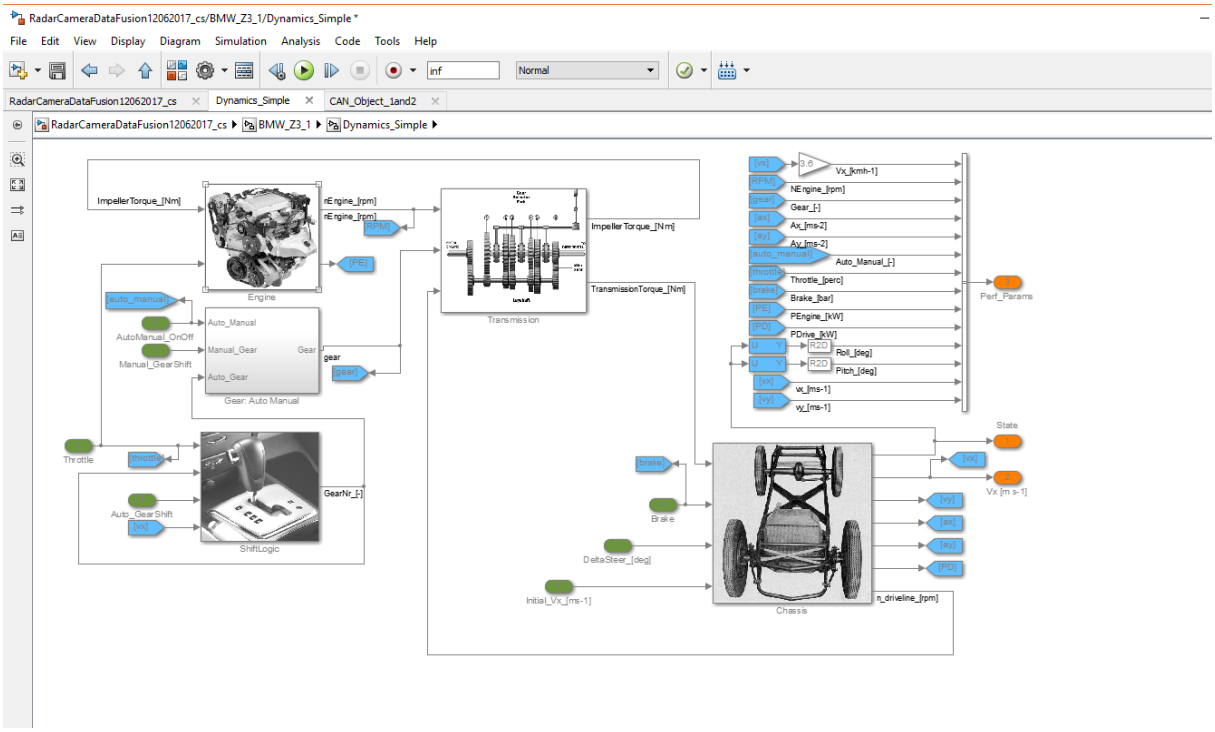


Ilustración 65. Modelo dinámico del vehículo (Nivel Inferior)

Información Sensores

Radar

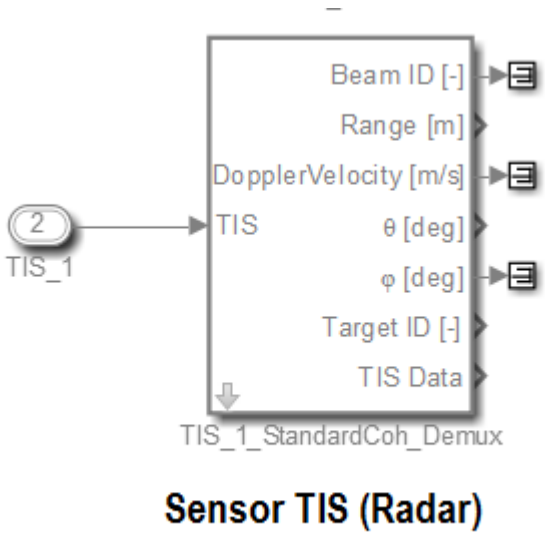


Ilustración 66. Bloque Información radar generada por PreScan

El sensor TIS, ofrece información acerca del entorno y de los obstáculos que encuentra. Las salidas que ofrece son:

Beam ID	ID del foco radar
Range [m]	Distancia relativa del sensor al obstáculo
Doppler Velocity [m]	Velocidad relativa del obstáculo respecto al sensor
θ [deg]	Ángulo azimut.
φ [deg]	Orientación del obstáculo
Target ID	ID del obstáculo, (un número entero para cada obstáculo)
TIS Data	Incluye más información acerca de los obstáculos, la cual se puede ver accediendo dentro del bloque.

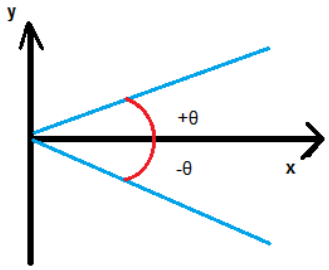


Ilustración 67. Criterio de signos sensores PreScan

Cámara

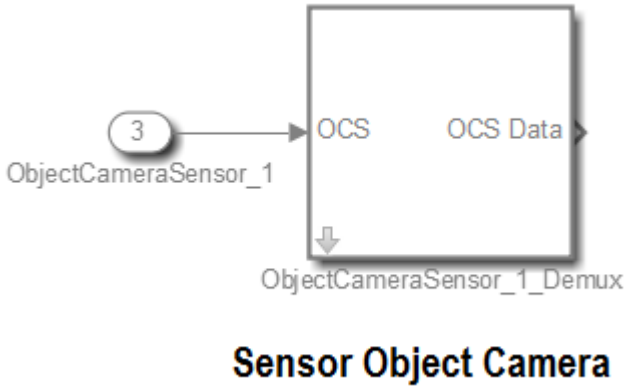


Ilustración 68. Bloque Información Cámara generada por PreScan

El bloque de la cámara también ofrece información acerca de los obstáculos, pero no se aprecia, en este nivel, la información. Es necesario acceder a un nivel inferior para ver qué información ofrece.

Aquí se describen las salidas que ofrece la cámara:

Object ID	Es un identificador exclusivo para cada obstáculo.
Object Type	Dependiendo el tipo de obstáculo (coche, humano, camión, etc), lo clasificará.
Left [m]	Informa, en metros, sobre la distancia que existe desde el sensor hasta el límite izquierdo del obstáculo.

Right [m	Informa, en metros, sobre la distancia que existe desde el sensor hasta el límite derecho del obstáculo.
Bottom [m]	Informa, en metros, sobre la distancia que existe desde el sensor hasta el límite inferior del obstáculo.
Top [m]	Informa, en metros, sobre la distancia que existe desde el sensor hasta el límite superior del obstáculo.
Range [m]	Distancia en metros desde el sensor al obstáculo.
Doppler Velocity [m/s]	Velocidad relativa del obstáculo respecto al sensor.
Theta [deg]	Ángulo azimut.
Phi [deg]	Orientación del vehículo.

Tabla 3

Con esta parte se puede dar por concluida la fase de simulación.

5.3 Adaptación, transmisión y recepción de las señales en el entorno de programación Simulink mediante el protocolo de comunicaciones CAN.

En este apartado se pretende, partiendo de la información de los sensores simulados de PreScan, acondicionar los mensajes que Baselabs necesita de entrada para realizar la fusión de datos.

Para entender un poco mejor lo que se desea conseguir, en la siguiente imagen se muestra, una comparación entre el sistema existente en el vehículo actualmente, y el sistema simulado que se desea conseguir.

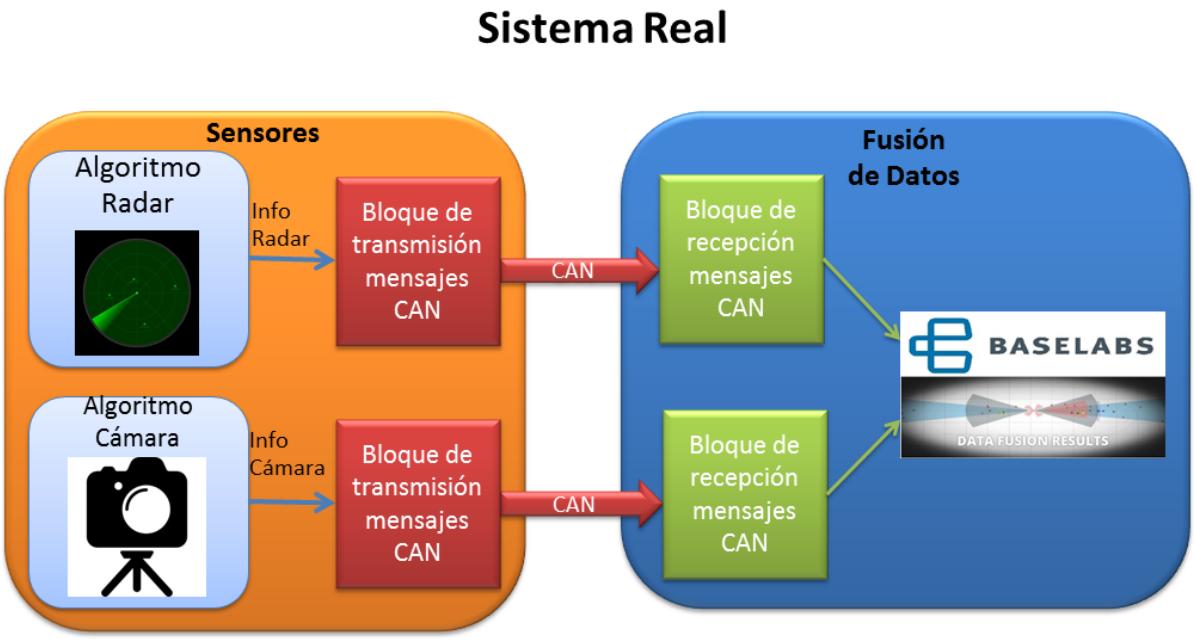


Ilustración 69. Ilustración 69. Esquema Simulación Sistema Real

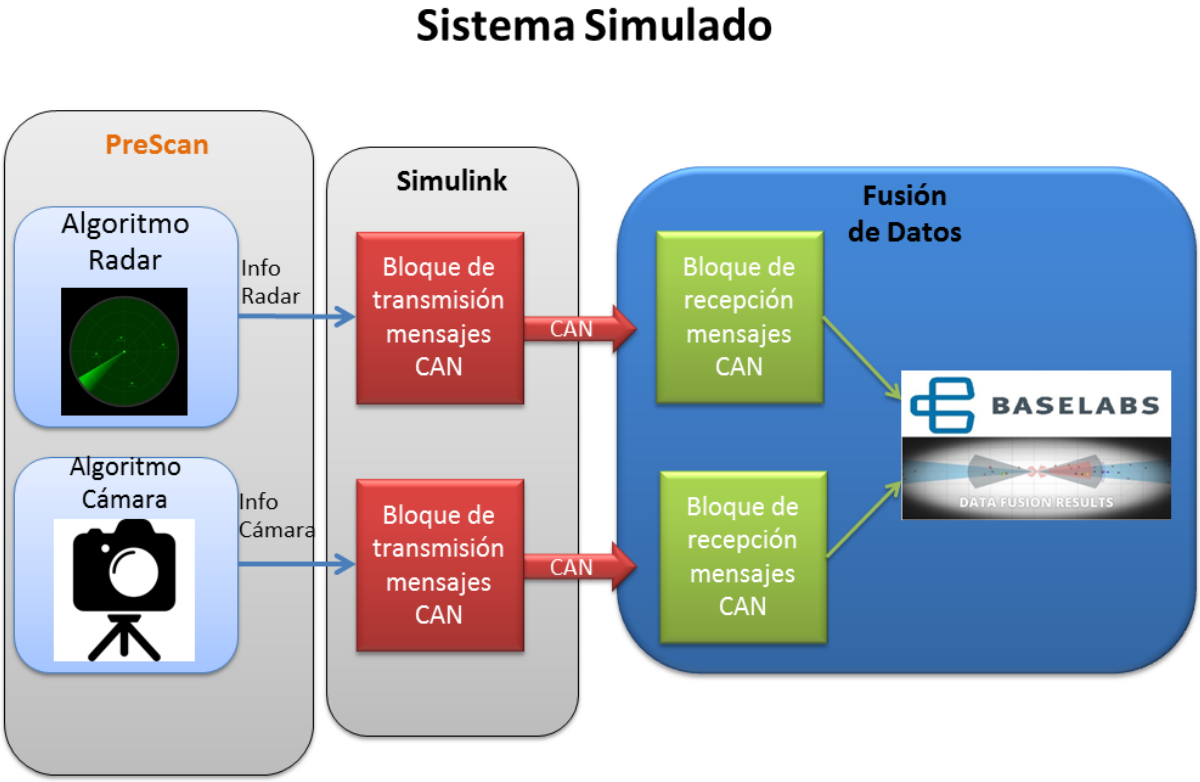


Ilustración 70. Esquema Simulación Sistema Simulado

Para reproducir estos mensajes, Simulink cuenta con unas librerías de comunicación CAN, a través de las cuales se puede tanto transmitir como recibir mensajes CAN.

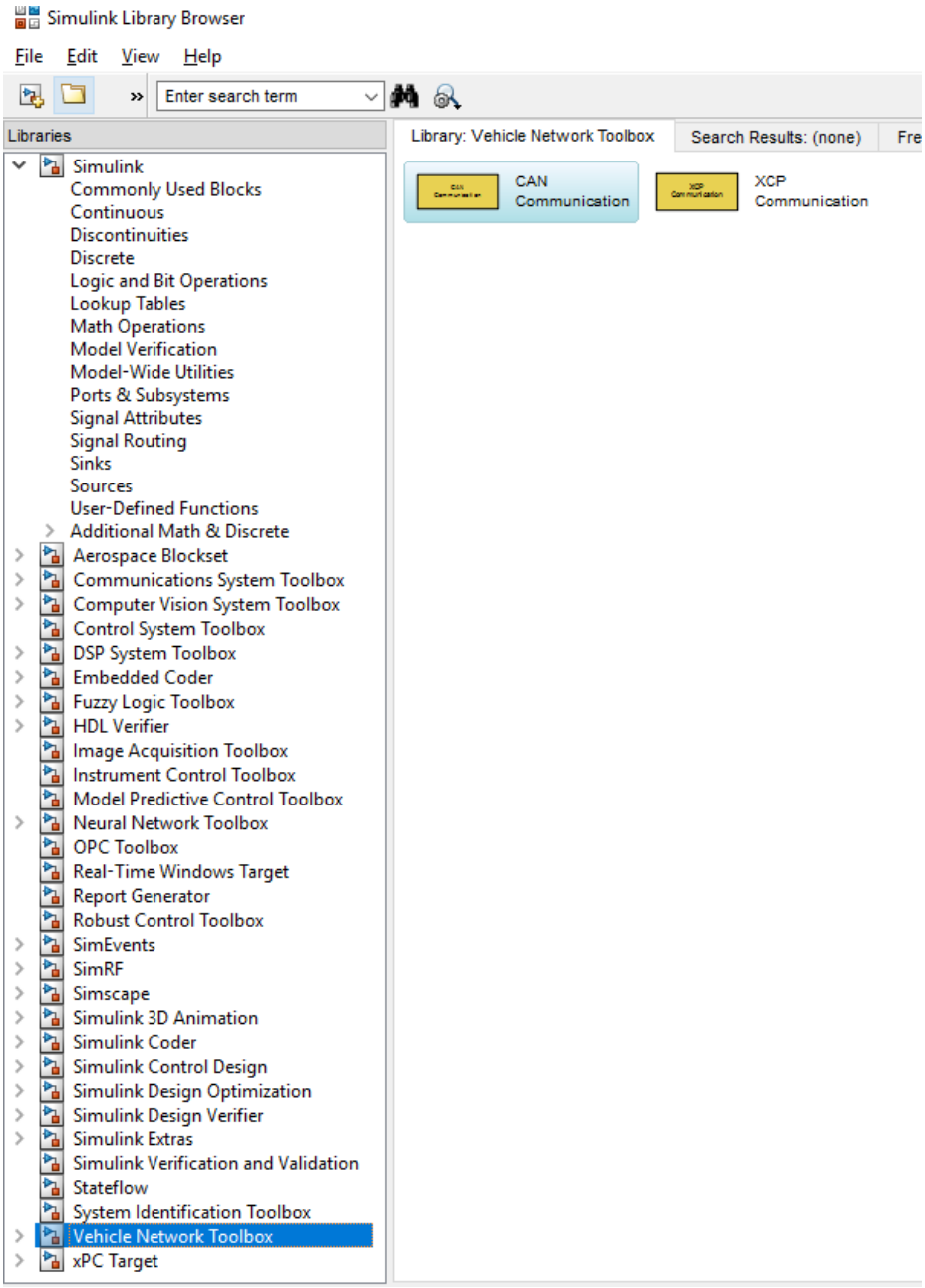


Ilustración 71 . Librería CAN en Simulink

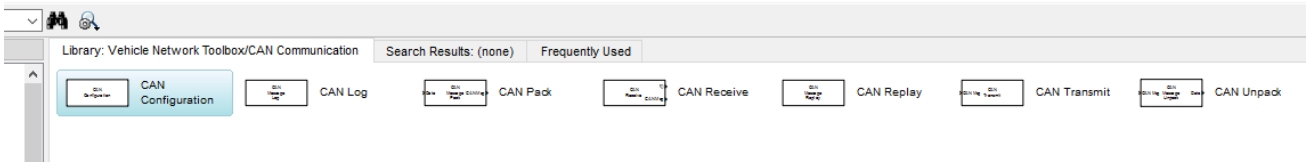


Ilustración 72 . Bloques librería CAN en Simulink

De todos estos bloques, a continuación, serán descritos los que han sido utilizados para este TFG.

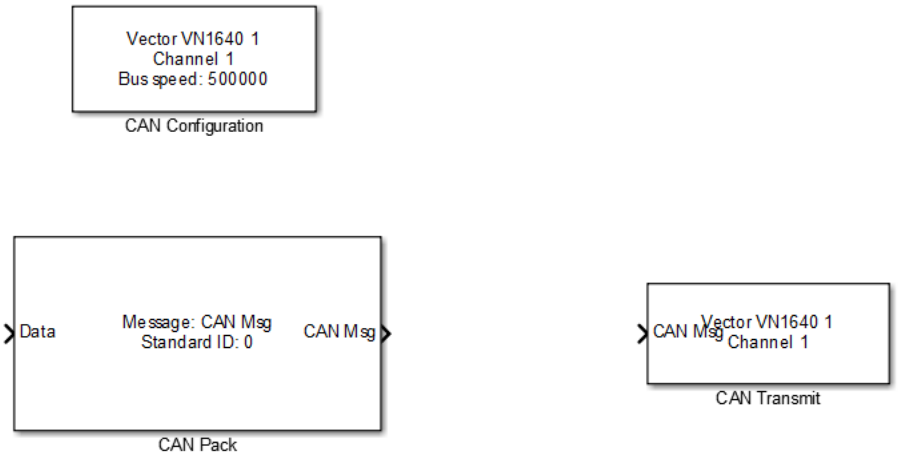


Ilustración 73. Bloques CAN empleados

- **CAN Configuration**

  Este bloque especifica cual será la velocidad a la que trabaja el bus y el canal de transmisión. En nuestro caso, los mensajes se transmitirán a través del canal 1 y a una velocidad de 500 KBauds.

- **CAN Transmit**

  Este bloque será el encargado de transmitir los mensajes, solamente tiene como parámetros de entrada el canal de transmisión y la frecuencia de transmisión de los mensajes, que para nuestro caso será aproximadamente de unos 100 Hz, es decir, un mensaje cada 10 milisegundos. La precisión en la transmisión está limitada al sistema operativo donde corre la simulación, Windows 7.

- **CAN Pack**

Este bloque es el encargado de decir cuáles son los mensajes se desean enviar, para que este bloque entienda de mensajes, es necesario asignarle un **fichero.dbc,** este fichero es una base de datos con la información sobre los mensajes CAN.

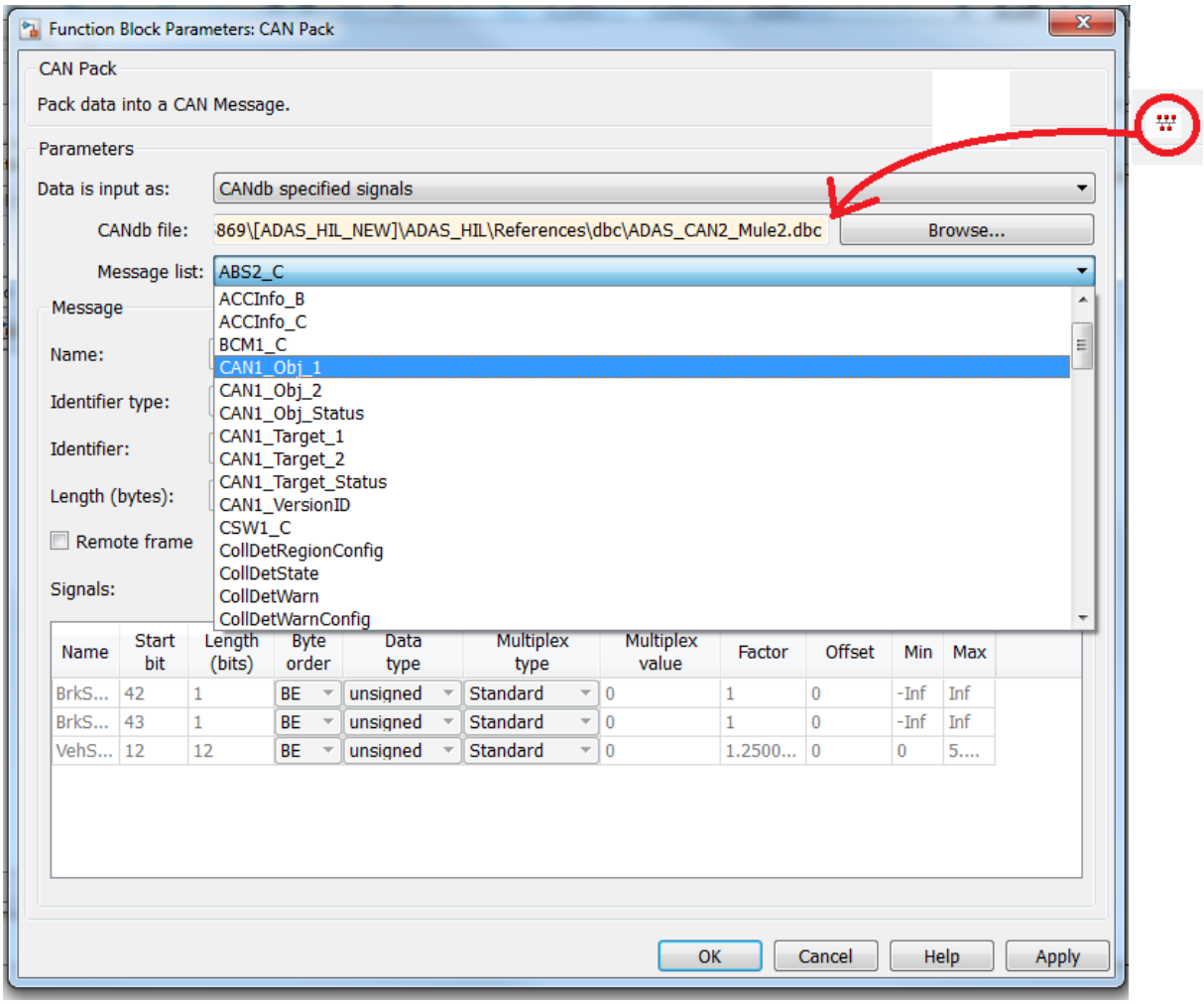


Ilustración 74. Bloque CAN Pack con dbc insertada

Y de esta manera, permitirá seleccionar el tipo de mensaje que se desea enviar.

Ahora se especificarán cuáles son los mensajes de radar y cámara que Baselabs necesita para la fusión de datos. A continuación, se procederá a su adaptación para su posterior envío.

Se realizará en primer lugar la adaptación de la cámara por simplicidad.

5.3.1 Cámara

La cámara en el sistema ADAS tiene el objetivo de detectar objetos y además calcular datos que son requeridos por el algoritmo ADAS.

Parámetros de comunicación

- Los mensajes se envían en un formato de encabezado CAN de **11 bits**.
- El baud rate (velocidad de baudios) debe ser de unos **500 kbps**.
- El barrido (frame) CAN con los mensajes de los objetos es transmitido aproximadamente cada **66 ms- 100 ms**.

Señales requeridas

La cámara envía todos los mensajes vía CAN. Primero, ésta envía tres mensajes **Status** donde está definida información como el número de objetos detectados y otras características comunes sobre el barrido actual. (Véase apéndice I).

- **ObstacleStatusA**
- **ObstacleStatusB**
- **ObstacleStatusC**

Después de estos mensajes, la cámara envía mensajes específicos de cada obstáculo detectado, hasta un máximo de 10.

Para cada objeto detectado envía un conjunto de 3 mensajes, por ejemplo, para el obstáculo 1 sería.

- **ObstacleDataA1**
- **ObstacleDataB1**
- **ObstacleDataC1**

Una vez conocidos los mensajes que se necesita enviar, se procederá a su envío.

En primer lugar, se ha extraído la información relevante que aporta la cámara.

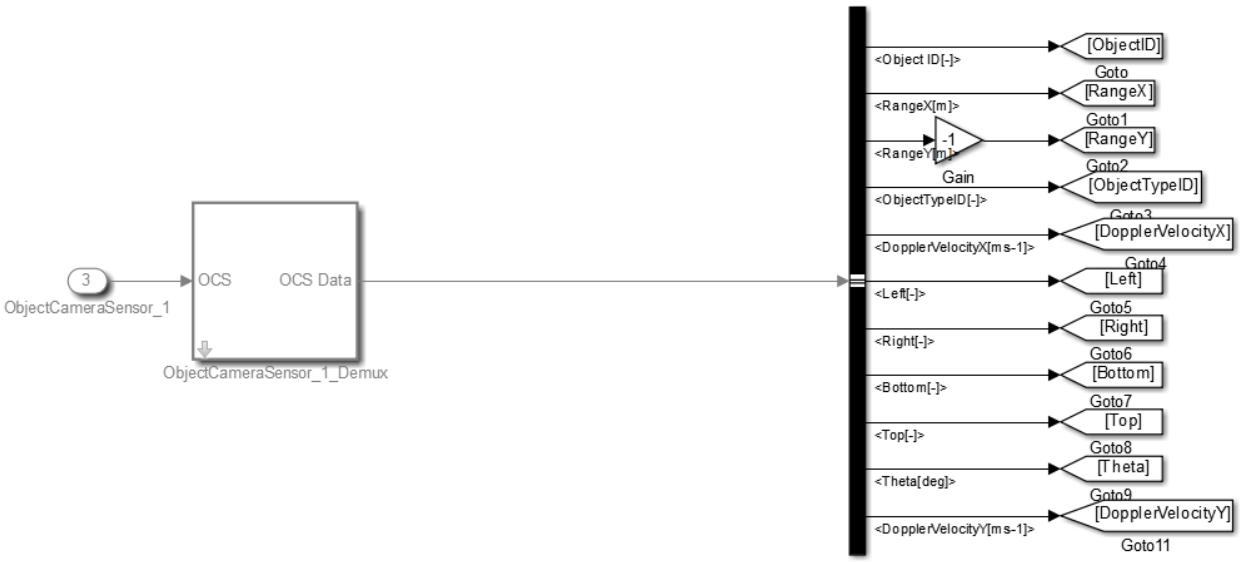


Ilustración 72. Información extraída de la cámara

A continuación, se han creado algunos bloques de adaptación customizados, como los que se observan más abajo. Los cuales permiten traducir los mensajes provenientes de PreScan para que Baselabs pueda entenderlos. (Véase descripción de bloques de adaptación en Apéndice II)

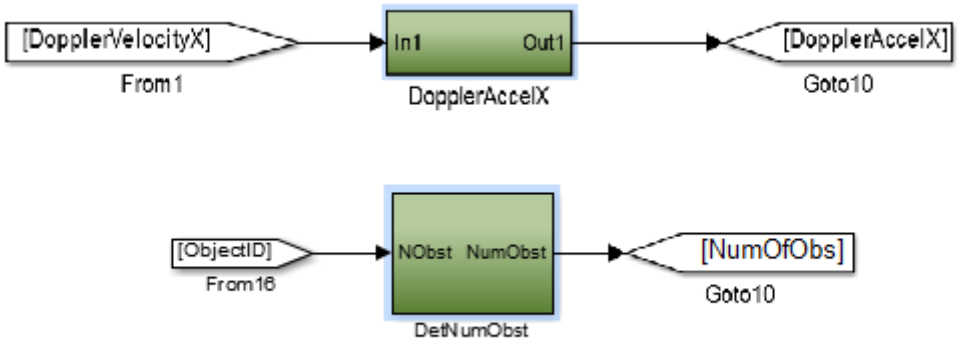


Ilustración 73. Bloques de adaptación Cámara

Una vez se han realizado las adaptaciones correspondientes, se envían los mensajes a través de los mensajes de transmisión CAN.

No se entrará muy en detalle en los valores asignados para cada entada, se adjuntan las dbc con la información acerca de las señales de cada mensaje y sus distintos valores.

ObstacleStatusA

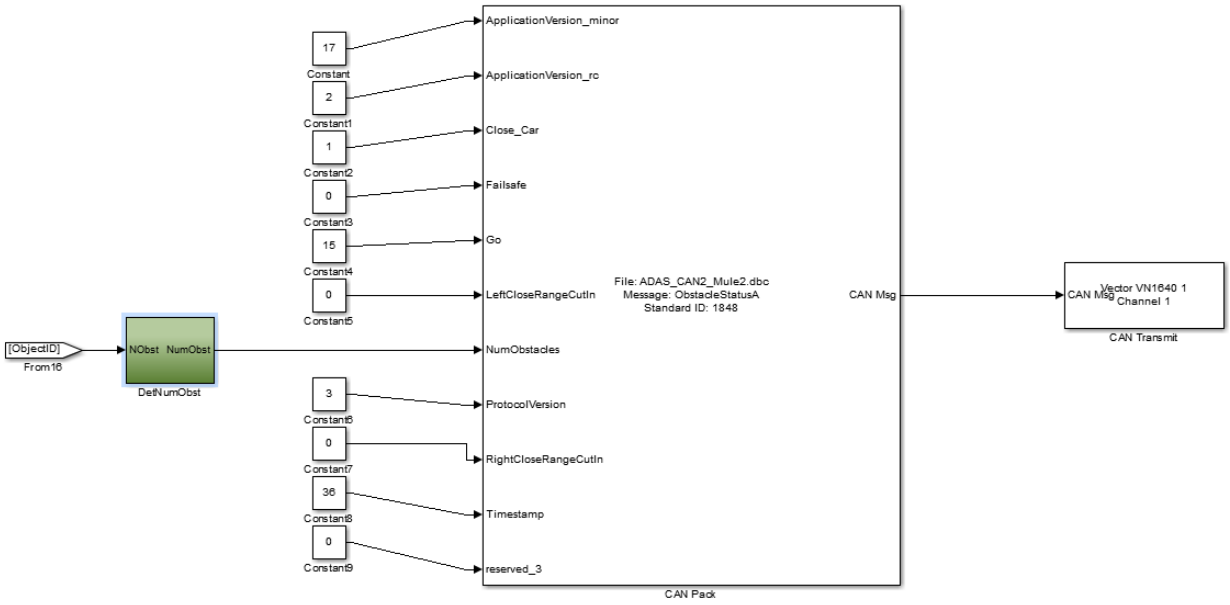


Ilustración 74. ObstacleStatusA configurado

ObstacleStatusB

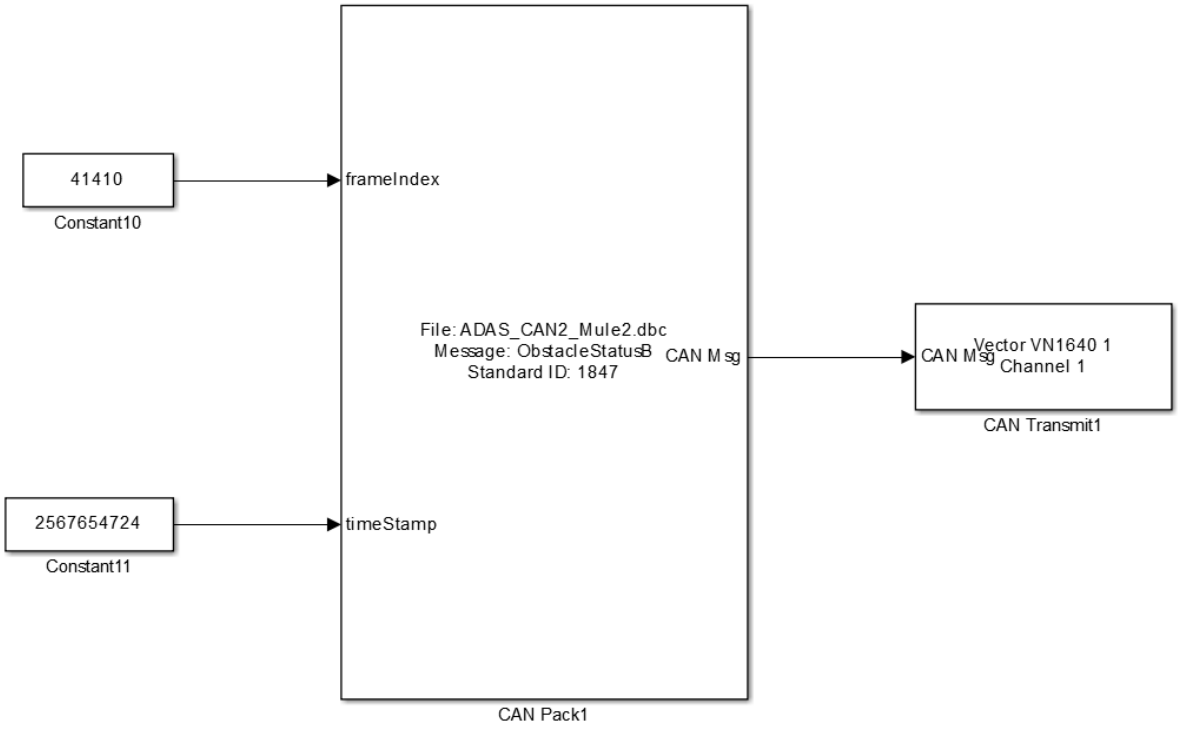


Ilustración 75. ObstacleStatusB configurado

ObstacleStatusC

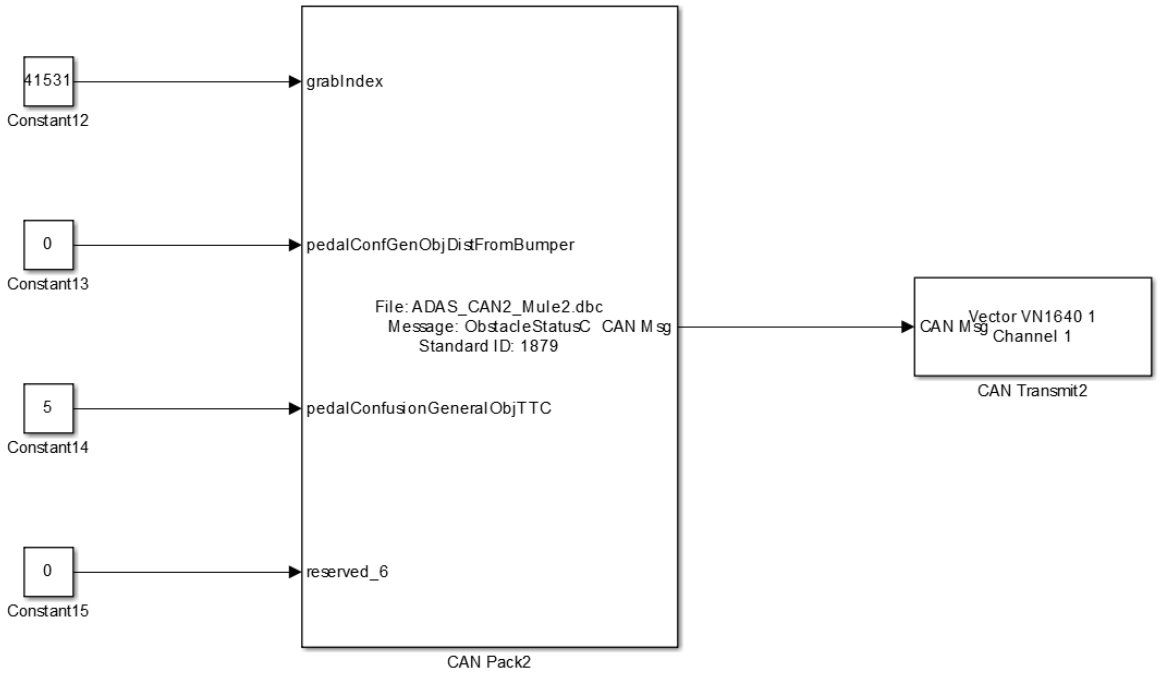


Ilustración 76. ObstacleStatusC configurado

A continuación, se procederá con los mensajes específicos de cada obstáculo, se realizará el ejemplo sólo para un obstáculo, ya que el proceso será idéntico para los 10 obstáculos. (Véase Apéndice III)

ObstacleDataA1

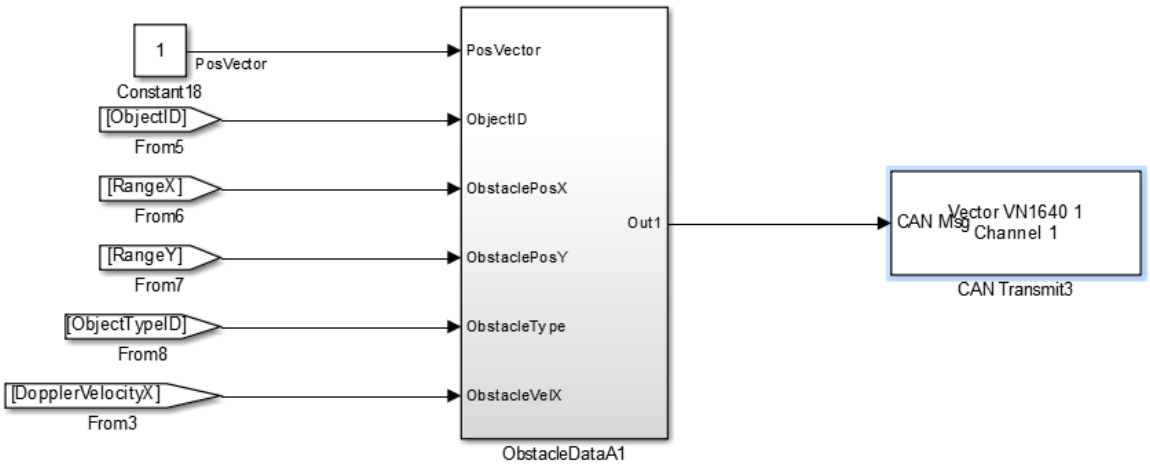


Ilustración 77. ObstacleDataA1 Configurado

ObstacleDataB1

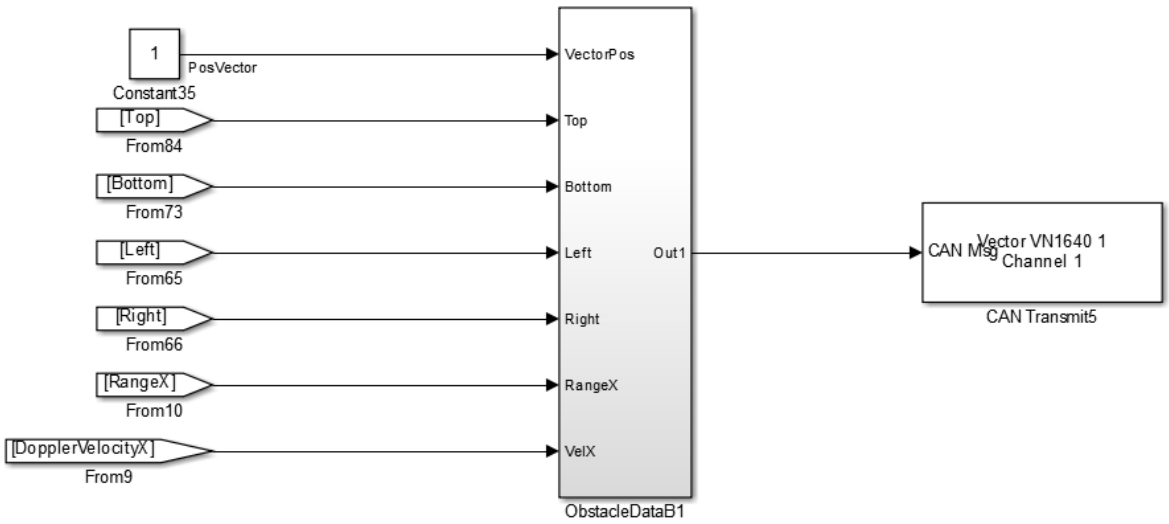


Ilustración 78. ObstacleDataB1 Configurado

ObstacleDataC1

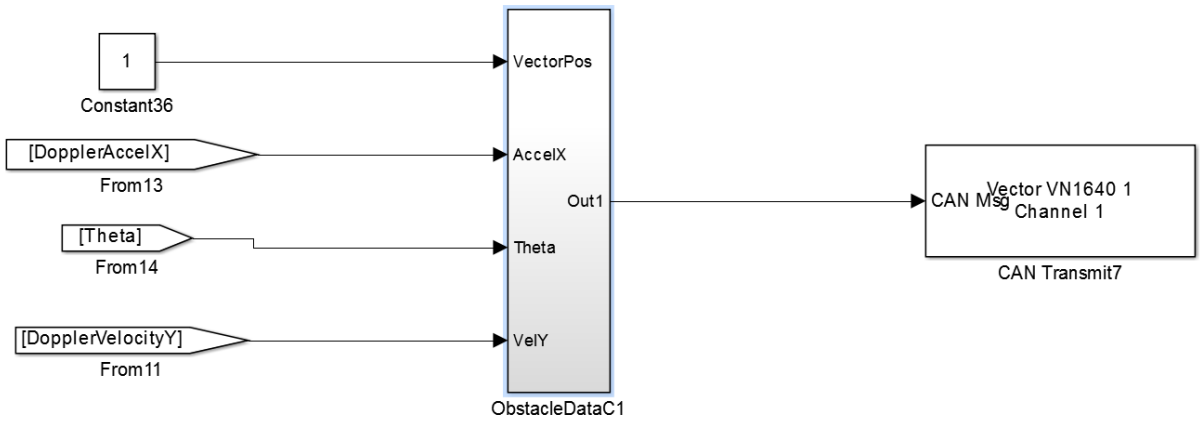


Ilustración 79. ObstacleDataC1 Configurado

5.3.2 Radar

El radar tiene el objetivo de detectar los objetos y, además, calcular algunos valores que son requeridas por el algoritmo ADAS.

Parámetros de comunicación

- Los mensajes se envían en un formato de encabezado CAN de **11 bits**.
- El baud rate (velocidad de baudios) debe ser de unos **500 kbps**.
- El barrido (frame) CAN con los mensajes de los objetos es transmitido aproximadamente cada **66 ms**.

El radar no envía un mensaje para cada objeto como la cámara. Para enviar la información solamente usa tres mensajes. Uno para la información del estado del frame actual de los obstáculos detectados, y dos más correspondientes a la información asociada para cada obstáculo detectado.

En la siguiente imagen se muestra cómo los mensajes son enviados vía CAN.

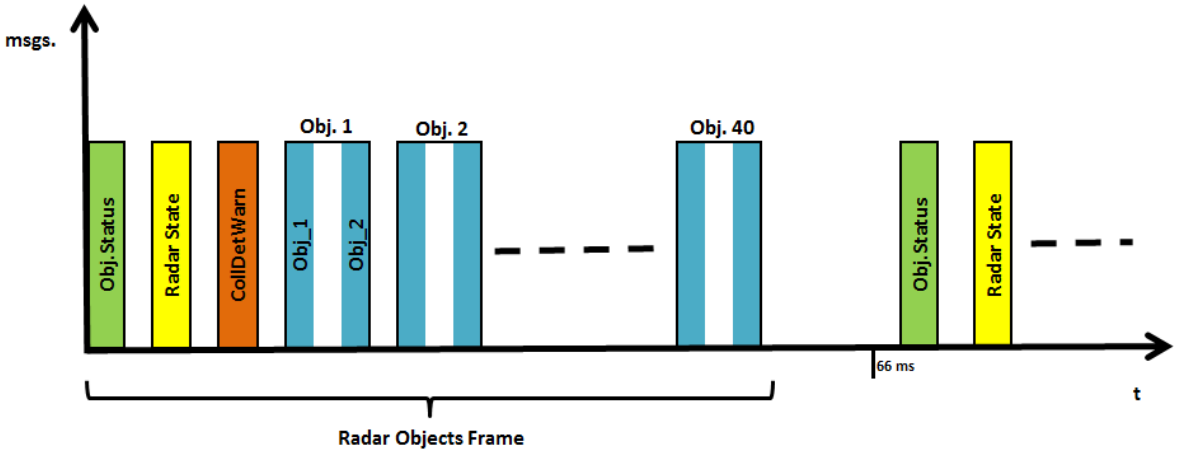


Ilustración 80. Esquema de envío mensajes CAN Radar

Los mensajes que el radar debe enviar a Baselabs, están incluidos en 3 bloques concretamente. (Véase Apéndice I)

- **CAN1_Obj_Status**
- **CAN1_Obj_1**
- **CAN1_Obj_2**

Una vez conocidos los mensajes, se procederá, en primer lugar, extrayendo la información referente al radar, al igual que se hizo con la cámara.

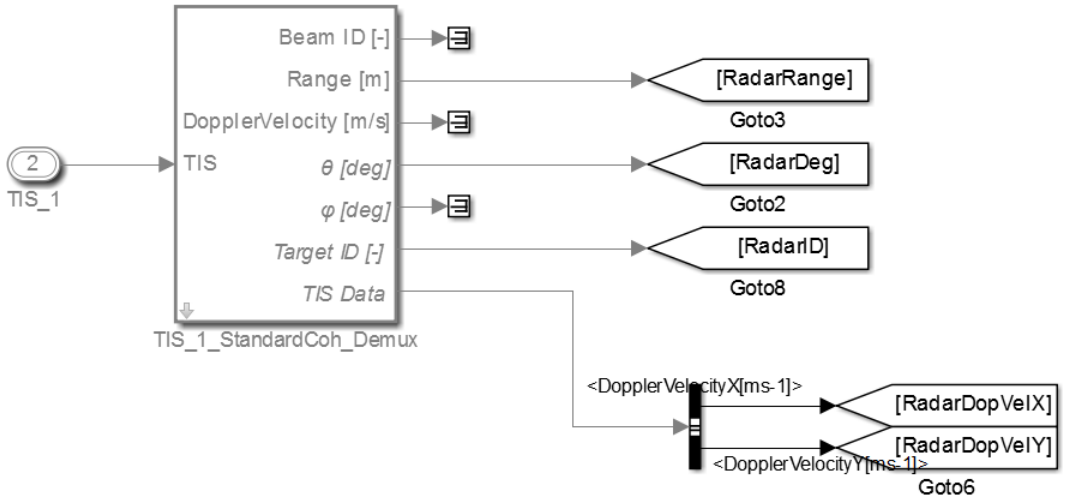


Ilustración 81. Información extraída del radar

Y a continuación se detallan los bloques de adaptación de la información. Estos bloques permiten traducir la información proveniente de PreScan, y hacerla

legible para Baselabs. (Véase en apéndice II la descripción de los bloques de adaptación).

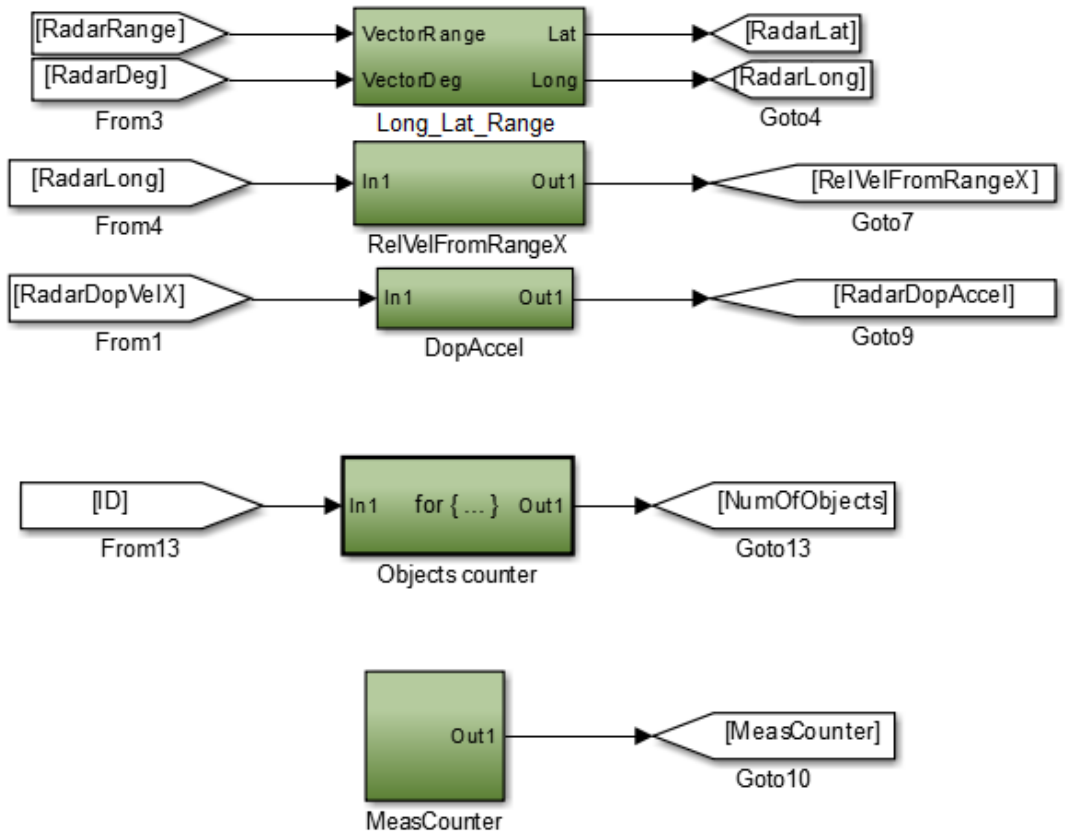


Ilustración 82. Bloques de adaptación Radar

Finalmente, quedará enviar la información de manera adecuada. (Véase Apéndice III). El radar se comporta de manera diferente a la cámara, ya que en el mismo barrido y con sólo 2 mensajes CAN, el sensor envía la información referente a los 40 obstáculos que es capaz de detectar.

Para simular el envío de estos mensajes, se ha utilizado un bucle For, que realiza una iteración para cada obstáculo y envía la información vía CAN, a través de los 2 mensajes correspondientes, como se puede observar más abajo.

También se ha incluido este módulo en un subsistema para facilitar su comprensión.

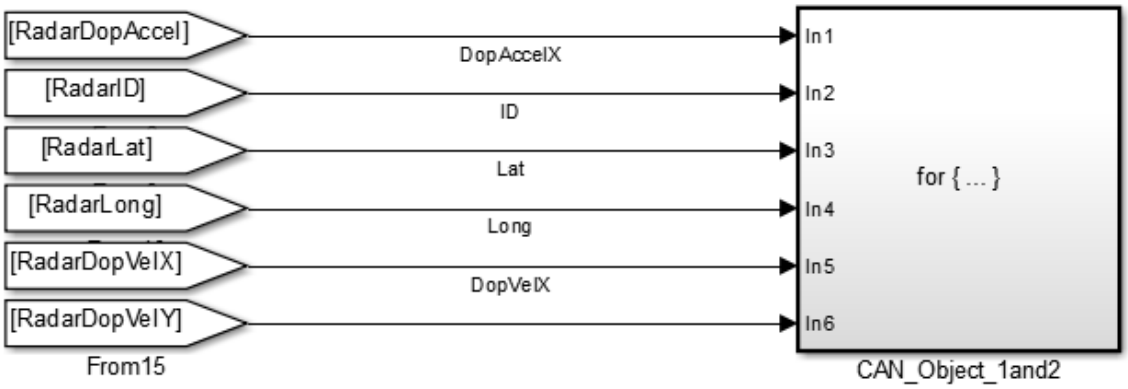


Ilustración 83. Bloques obstáculos 1 y 2 configurados

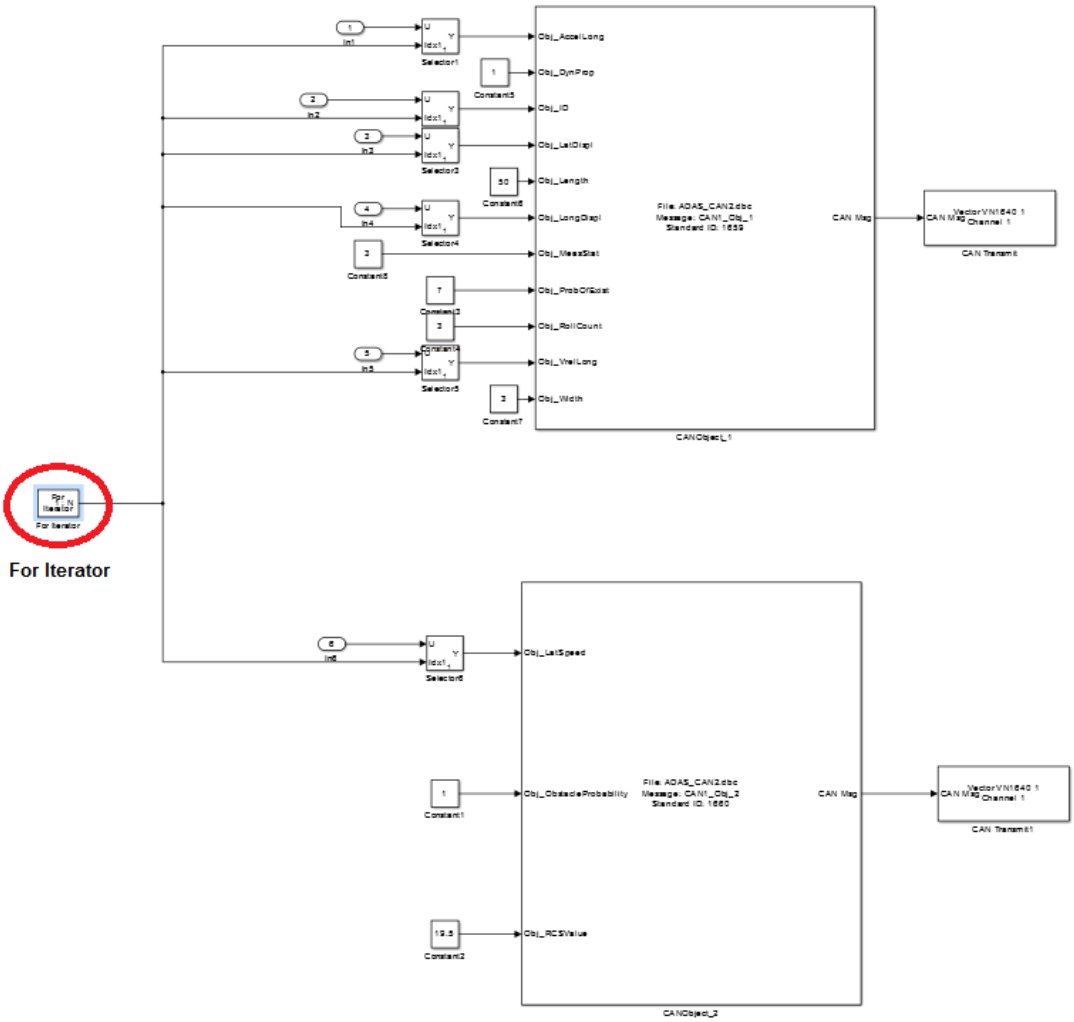


Ilustración 84. Bloques 1 y 2 configurados (Nivel inferior)

Finalmente, los mensajes CAN quedarían configurados correctamente para ser enviados.

5.4 Recepción, gestión y fusión de datos de los mensajes recibidos desde Simulink en Baselabs.

El último paso del proceso es verificar que se han recibido los mensajes CAN, usando la herramienta CANalyzer, y configurar BaseLabs, para el correcto funcionamiento del algoritmo de fusión de datos.

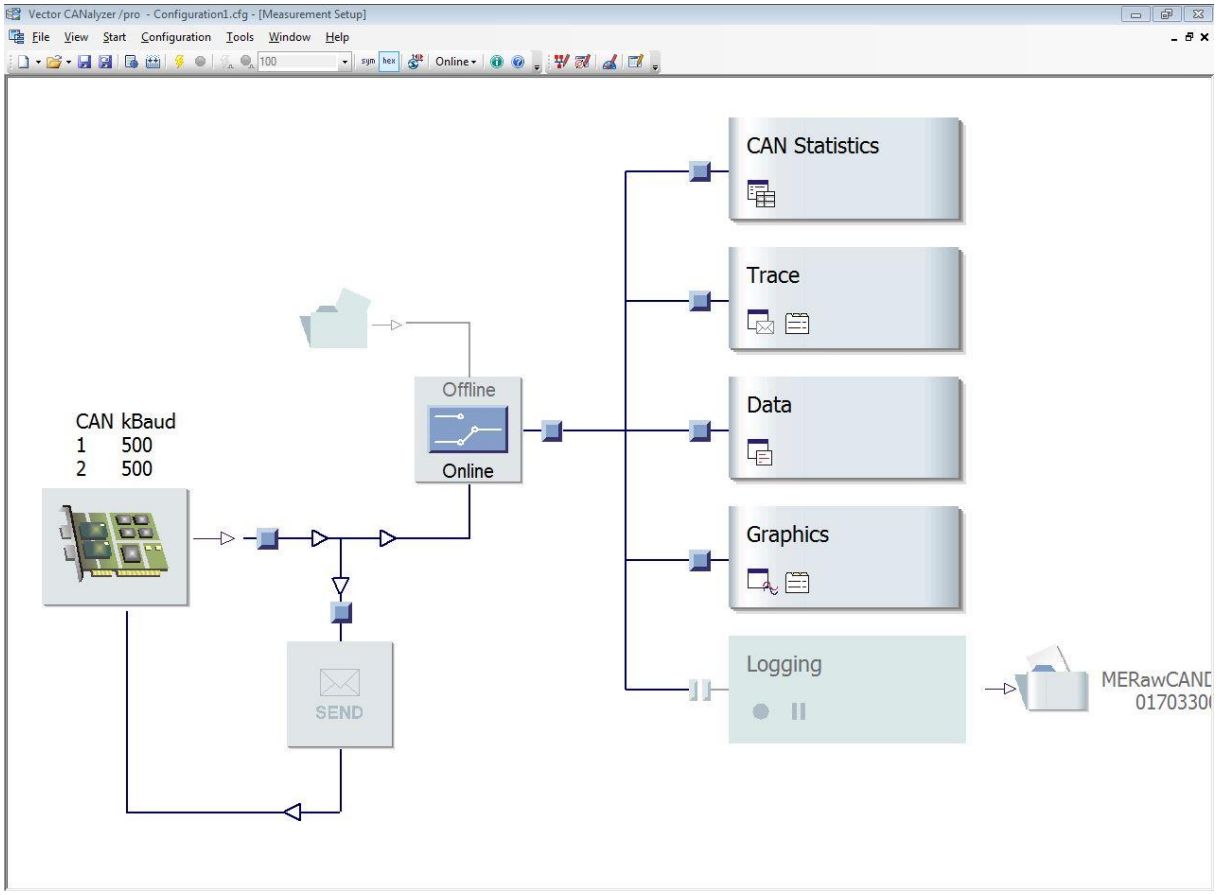
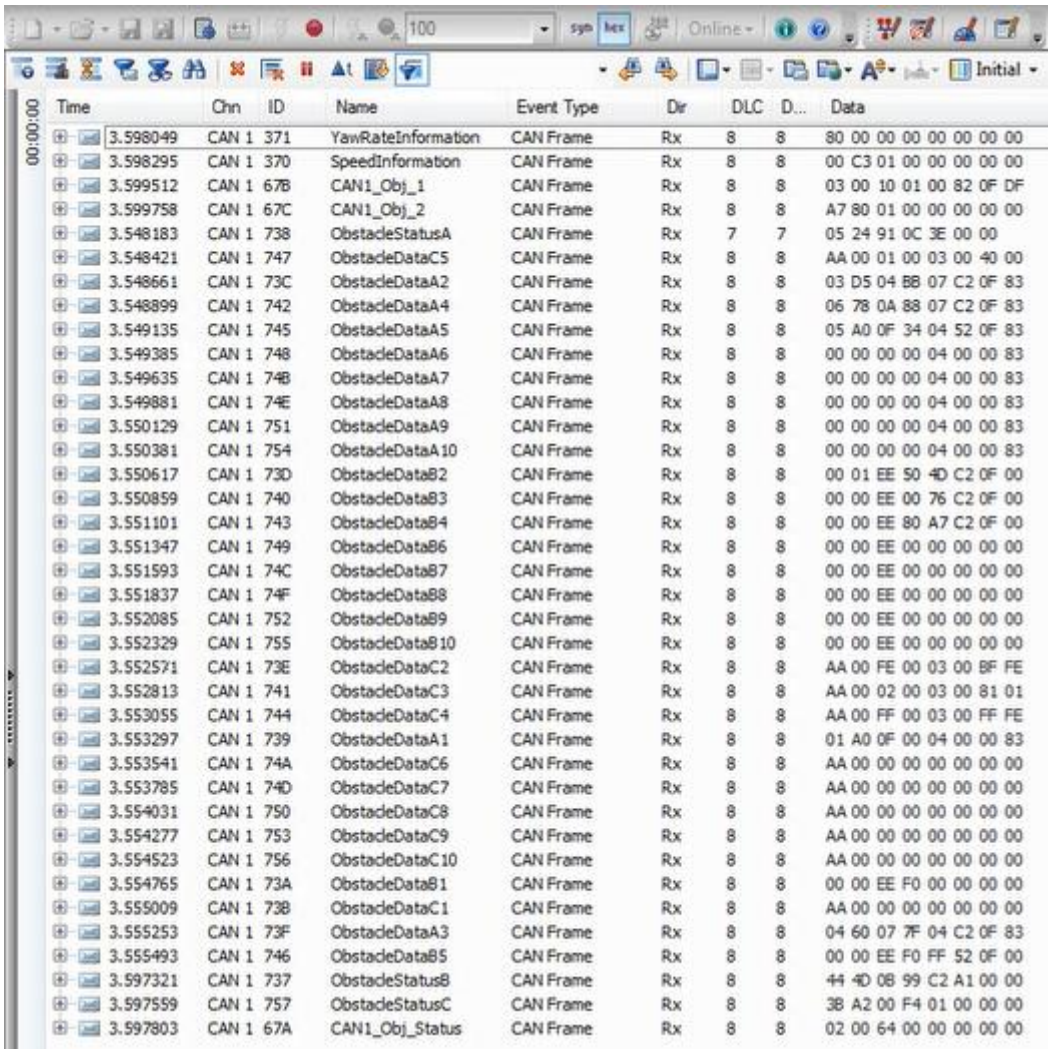


Ilustración 85. Configuración CANalyzer

Aquí podemos ver la ventana de configuración de CANalyzer, en ella se establece cómo se desea que llegue la información, esta herramienta permite reproducir grabaciones (**Offline**), o trabajar en directo (**Online**). Para dicho trabajo se ha habilitado el modo Online, ya que lo que se desea, es usando un cable de conexión CAN, recibir los mensajes que Simulink envía, y verificar que éstos llegan de manera correcta al ordenador de Baselabs.

El modo Trace, muestra los mensajes que se reciben y envían, vía CAN, además nos informa del tipo, velocidad, frecuencia y otras opciones de envío.

Aquí se adjunta una imagen de los mensajes que se reciben de Simulink.



Time	Chn	ID	Name	Event Type	Dir	DLC	D...	Data
3.598049	CAN 1	371	YawRateInformation	CAN Frame	Rx	8	8	80 00 00 00 00 00 00 00
3.598295	CAN 1	370	SpeedInformation	CAN Frame	Rx	8	8	00 C3 01 00 00 00 00 00
3.599512	CAN 1	67B	CAN1_Obj_1	CAN Frame	Rx	8	8	03 00 10 01 00 82 0F DF
3.599758	CAN 1	67C	CAN1_Obj_2	CAN Frame	Rx	8	8	A7 80 01 00 00 00 00 00
3.548183	CAN 1	738	ObstacleStatusA	CAN Frame	Rx	7	7	05 24 91 0C 3E 00 00
3.548421	CAN 1	747	ObstacleDataC5	CAN Frame	Rx	8	8	AA 00 01 00 03 00 40 00
3.548661	CAN 1	73C	ObstacleDataA2	CAN Frame	Rx	8	8	03 D5 04 BB 07 C2 0F 83
3.548899	CAN 1	742	ObstacleDataA4	CAN Frame	Rx	8	8	06 78 0A 88 07 C2 0F 83
3.549135	CAN 1	745	ObstacleDataA5	CAN Frame	Rx	8	8	05 A0 0F 34 04 52 0F 83
3.549385	CAN 1	748	ObstacleDataA6	CAN Frame	Rx	8	8	00 00 00 00 04 00 00 83
3.549635	CAN 1	74B	ObstacleDataA7	CAN Frame	Rx	8	8	00 00 00 00 04 00 00 83
3.549881	CAN 1	74E	ObstacleDataA8	CAN Frame	Rx	8	8	00 00 00 00 04 00 00 83
3.550129	CAN 1	751	ObstacleDataA9	CAN Frame	Rx	8	8	00 00 00 00 04 00 00 83
3.550381	CAN 1	754	ObstacleDataA10	CAN Frame	Rx	8	8	00 00 00 00 04 00 00 83
3.550617	CAN 1	73D	ObstacleDataB2	CAN Frame	Rx	8	8	00 01 EE 50 4D C2 0F 00
3.550859	CAN 1	740	ObstacleDataB3	CAN Frame	Rx	8	8	00 00 EE 00 76 C2 0F 00
3.551101	CAN 1	743	ObstacleDataB4	CAN Frame	Rx	8	8	00 00 EE 80 A7 C2 0F 00
3.551347	CAN 1	749	ObstacleDataB6	CAN Frame	Rx	8	8	00 00 EE 00 00 00 00 00
3.551593	CAN 1	74C	ObstacleDataB7	CAN Frame	Rx	8	8	00 00 EE 00 00 00 00 00
3.551837	CAN 1	74F	ObstacleDataB8	CAN Frame	Rx	8	8	00 00 EE 00 00 00 00 00
3.552085	CAN 1	752	ObstacleDataB9	CAN Frame	Rx	8	8	00 00 EE 00 00 00 00 00
3.552329	CAN 1	755	ObstacleDataB10	CAN Frame	Rx	8	8	00 00 EE 00 00 00 00 00
3.552571	CAN 1	73E	ObstacleDataC2	CAN Frame	Rx	8	8	AA 00 FE 00 03 00 BF FE
3.552813	CAN 1	741	ObstacleDataC3	CAN Frame	Rx	8	8	AA 00 02 00 03 00 81 01
3.553055	CAN 1	744	ObstacleDataC4	CAN Frame	Rx	8	8	AA 00 FF 00 03 00 FF FE
3.553297	CAN 1	739	ObstacleDataA1	CAN Frame	Rx	8	8	01 A0 0F 00 04 00 00 83
3.553541	CAN 1	74A	ObstacleDataC6	CAN Frame	Rx	8	8	AA 00 00 00 00 00 00 00
3.553785	CAN 1	74D	ObstacleDataC7	CAN Frame	Rx	8	8	AA 00 00 00 00 00 00 00
3.554031	CAN 1	750	ObstacleDataC8	CAN Frame	Rx	8	8	AA 00 00 00 00 00 00 00
3.554277	CAN 1	753	ObstacleDataC9	CAN Frame	Rx	8	8	AA 00 00 00 00 00 00 00
3.554523	CAN 1	756	ObstacleDataC10	CAN Frame	Rx	8	8	AA 00 00 00 00 00 00 00
3.554765	CAN 1	73A	ObstacleDataB1	CAN Frame	Rx	8	8	00 00 EE F0 00 00 00 00
3.555009	CAN 1	73B	ObstacleDataC1	CAN Frame	Rx	8	8	AA 00 00 00 00 00 00 00
3.555253	CAN 1	73F	ObstacleDataA3	CAN Frame	Rx	8	8	04 60 07 7F 04 C2 0F 83
3.555493	CAN 1	746	ObstacleDataB5	CAN Frame	Rx	8	8	00 00 EE F0 FF 52 0F 00
3.597321	CAN 1	737	ObstacleStatusB	CAN Frame	Rx	8	8	44 4D 0B 99 C2 A1 00 00
3.597559	CAN 1	757	ObstacleStatusC	CAN Frame	Rx	8	8	3B A2 00 F4 01 00 00 00
3.597803	CAN 1	67A	CAN1_Obj_Status	CAN Frame	Rx	8	8	02 00 64 00 00 00 00 00

Ilustración 86. Mensajes recibidos en CANalyzer

Como se observa, los mensajes llegan correctamente vía CAN, con una frecuencia aproximada de 100 Hz.

Una vez se ha corroborado que los mensajes llegan de correctamente al PC de Baselabs, el último paso será configurar Baselabs para su correcto funcionamiento.

Baselabs tiene dos modos de funcionamiento, al igual que CANalyzer:

- **Modo Play (Reproducción)**

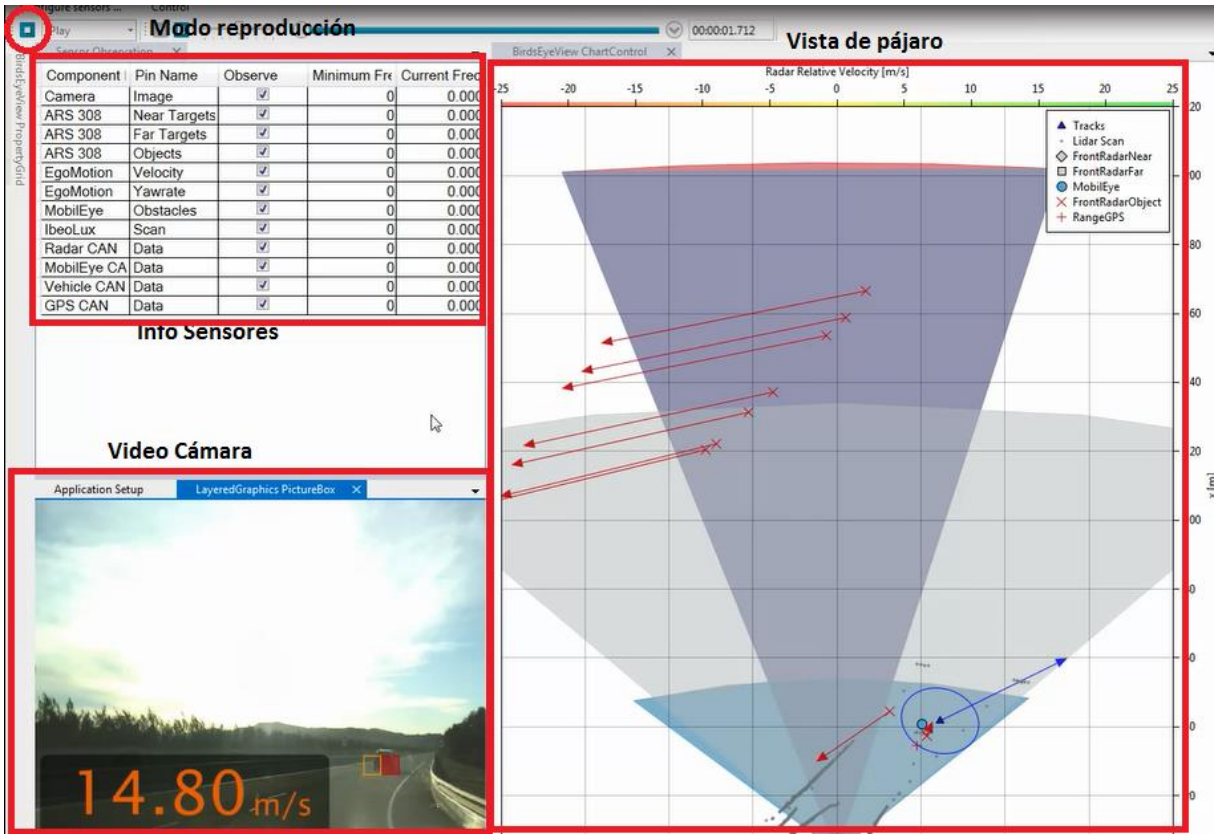


Ilustración 87. Baselabs modo reproducción

En este modo, Baselabs es capaz de reproducir una grabación hecha con los sensores reales, y realizar la fusión de datos.

- **Modo Live (En directo)**

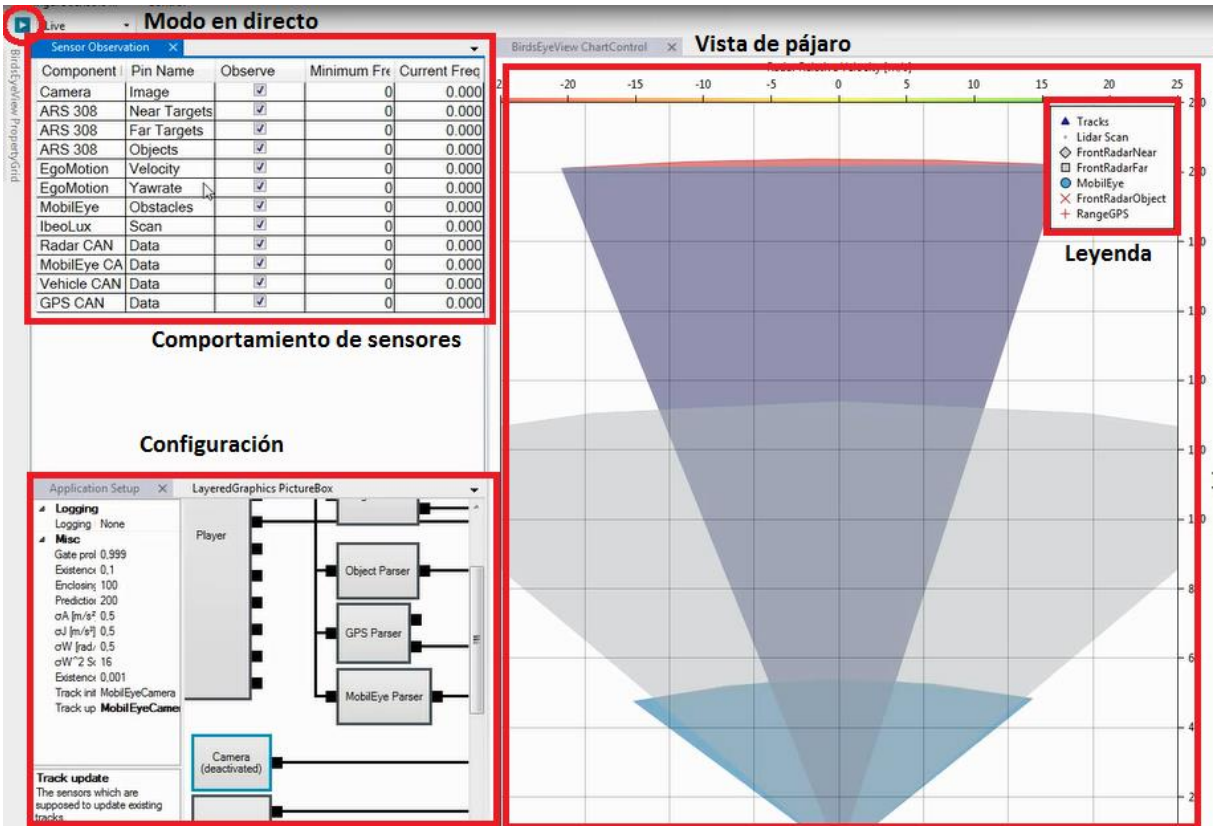


Ilustración 88. Baselabs modo LIVE

En este modo, Baselabs se conecta en tiempo real vía CAN, recibe los mensajes y realiza la fusión de datos.

Baselabs cuenta, además con una interfaz de configuración, en ella se establece, entre otras cosas, el tipo de sensor a usar para la fusión.

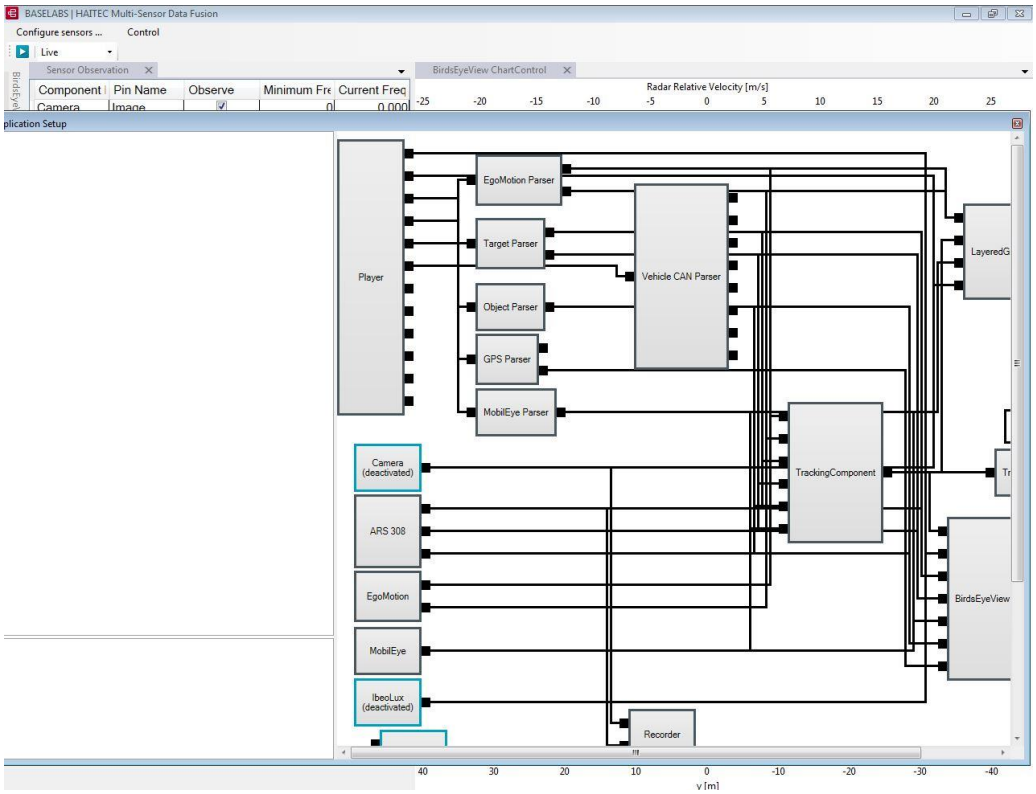


Ilustración 89. Configuración Baselabs

Finalmente, una vez configurado acorde a las especificaciones de este trabajo, se realiza la fusión de datos al recibir los mensajes CAN de Simulink.

Se adjuntan algunas imágenes donde se observa cómo se realiza esta fusión.

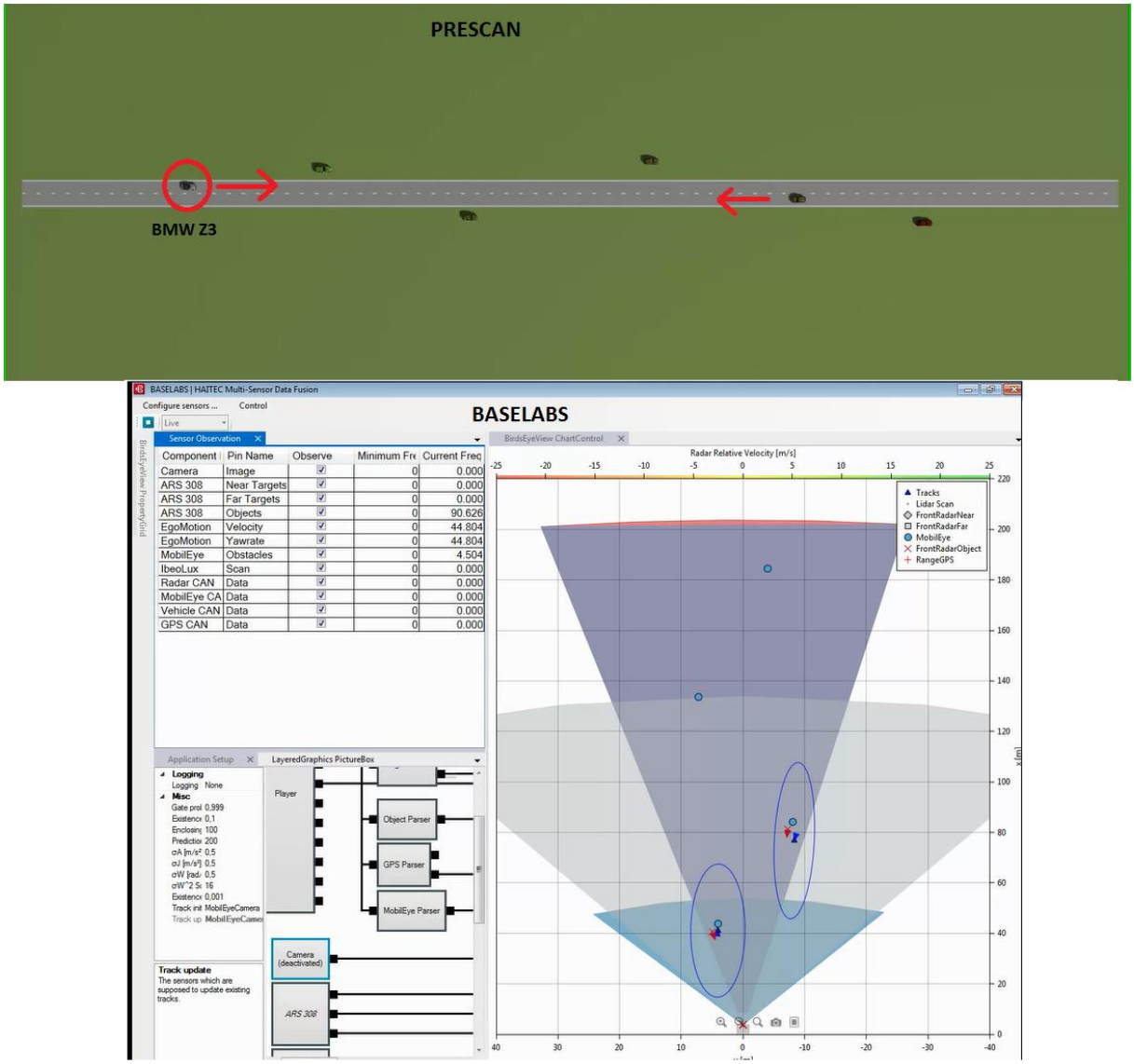


Ilustración 90. Simulación Completada 1

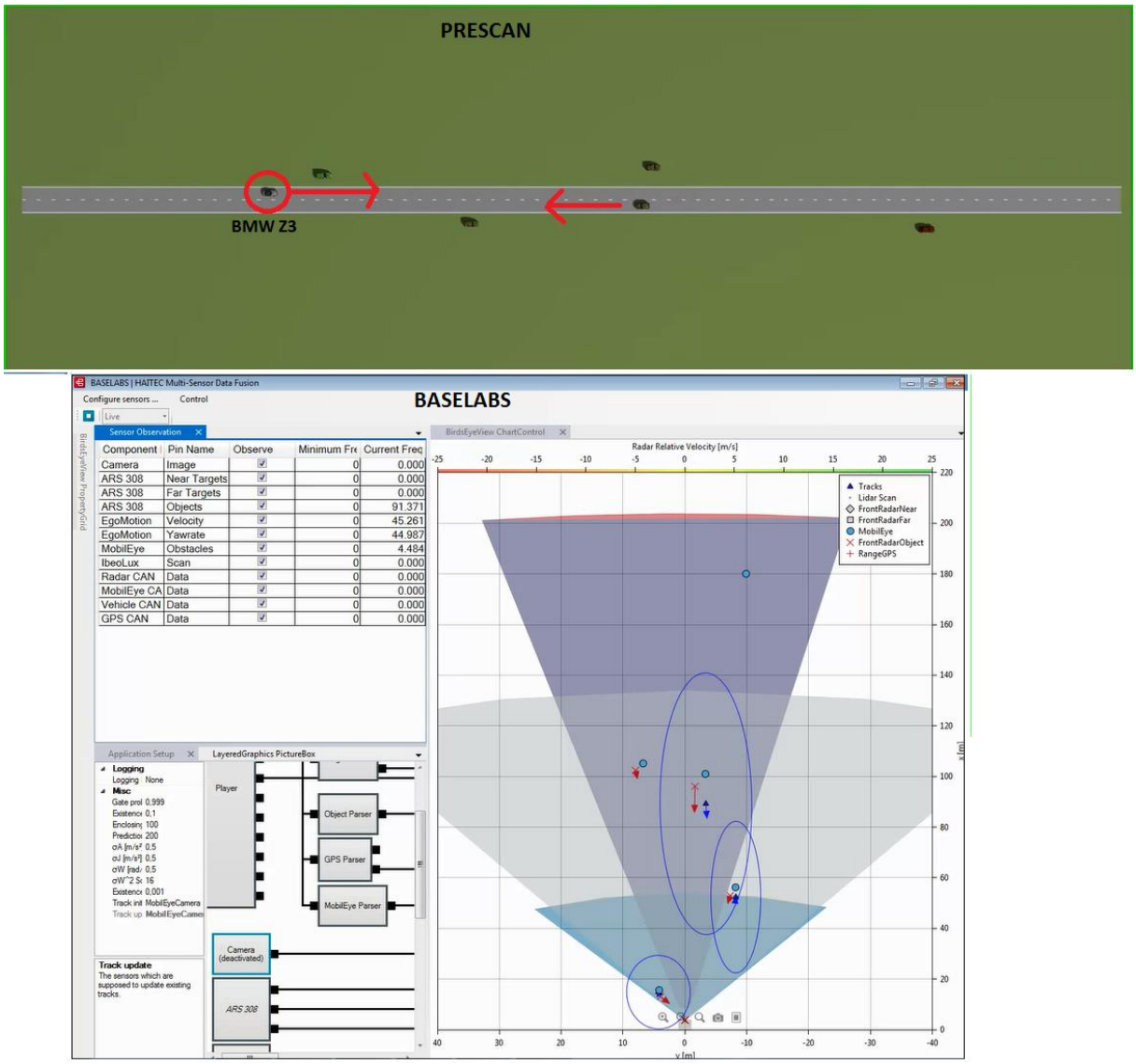


Ilustración 91. Simulación Completada 2

6. Conclusiones

Partiendo de los objetivos marcados al inicio del proyecto se puede afirmar que el sistema obtenido como resultado final ha cumplido todas las expectativas.

El sistema de simulación permite a Baselabs realizar la fusión de datos, sin apreciar que ésta se trata de un caso simulado, es decir, Baselabs no encuentra diferencia entre una prueba real o simulada.

Este sistema proporciona al usuario ciertas ventajas, ya que permitirá simular casos de uso que antes no encontraban viabilidad, debido al riesgo o coste que ellos implicaban.

Actualmente, el desarrollo del proyecto global se encuentra en la parte de simulación de un modelo dinámico de vehículo que será integrado en el PC en tiempo real PXI.

Se estima, para finales del mes de Julio, esté todo completamente integrado para su exposición.

Apéndices

Apéndice I

Mensajes Cámara

ObstacleStatusA

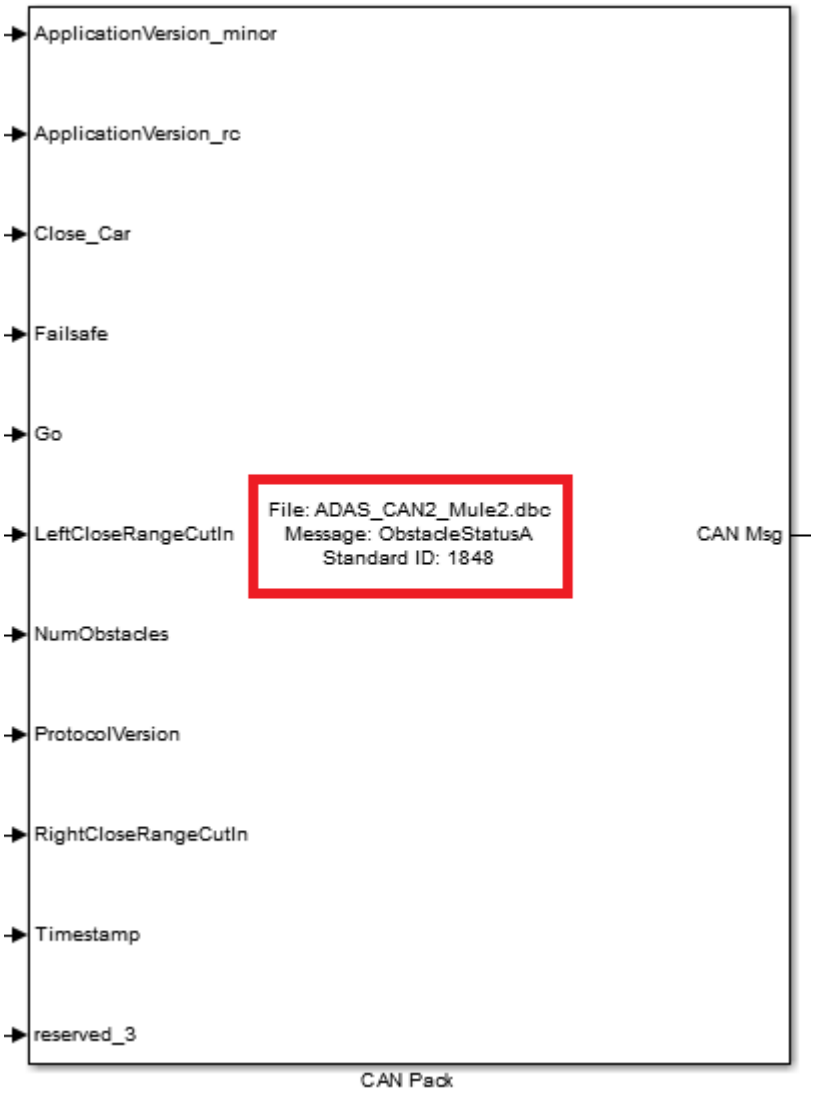


Ilustración 92. ObstacleStatusA

Application Version_minor	Versión de aplicación
ApplicationVersion_rc	-Vacío-
Close car	Identifica el objeto más cercano
Failsafe	Prueba de fallos
Go	Informa del estado del coche (Parado o

	en movimiento)
LeftCloseRangeCutIn	Invasión por el carril izquierdo
NumObstacles	Número de obstáculos
ProtocolVersion	Protocol Version
RightCloseRangeCutIn	Invasión por el carril izquierdo
TimeStamp	Tiempo de muestreo
Reserved_3	Mensaje reservado 3

ObstacleStatusB

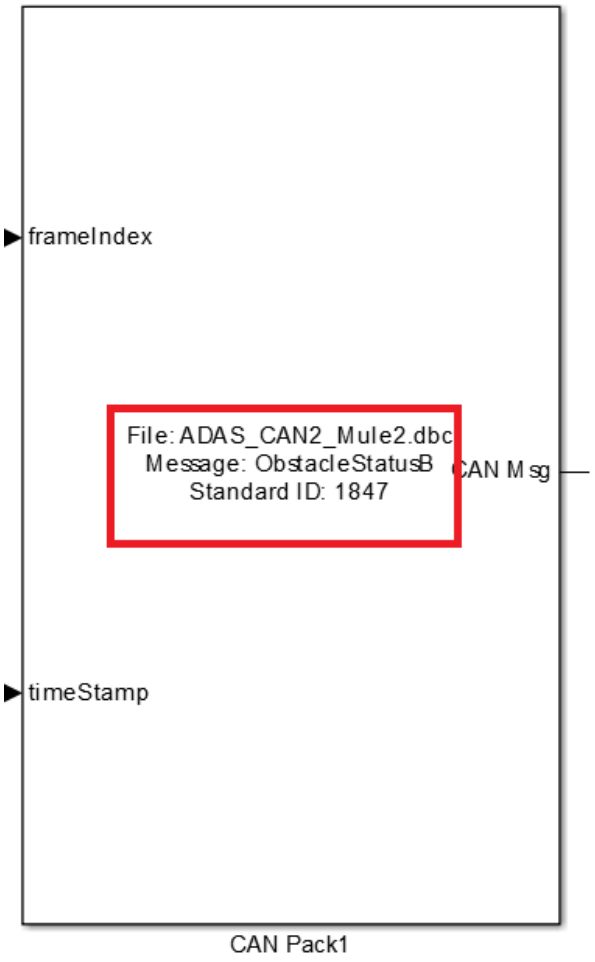


Ilustración 93. ObstacleStatusB

frameIndex	Número de frame
timeStamp	Tiempo de muestreo

ObstacleStatusC

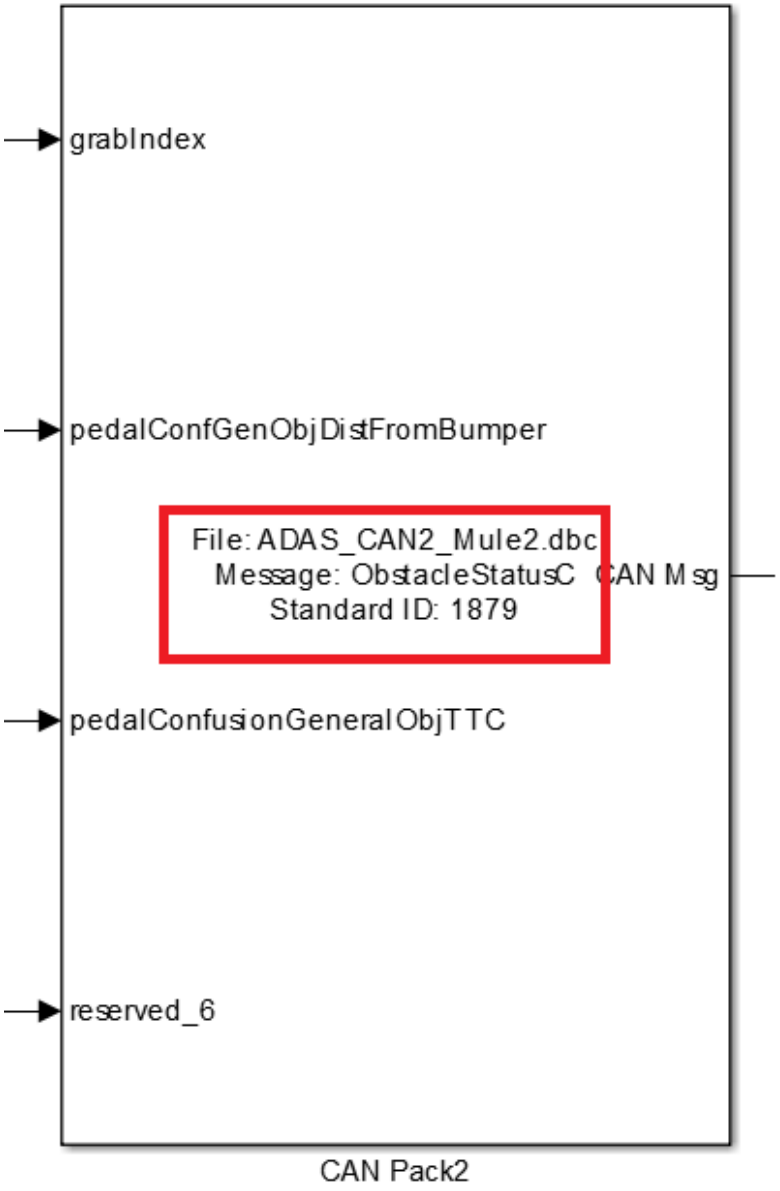


Ilustración 94. ObstacleStatusC

grabIndex	-Vacío-
pedalConfGenObjDistFromBumper	Distancia del sensor al parachoques
pedalConfusionGeneralObjTTC	Time to collision
reserved_6	-Vacío-

ObstacleDataA1-A2-…-A10

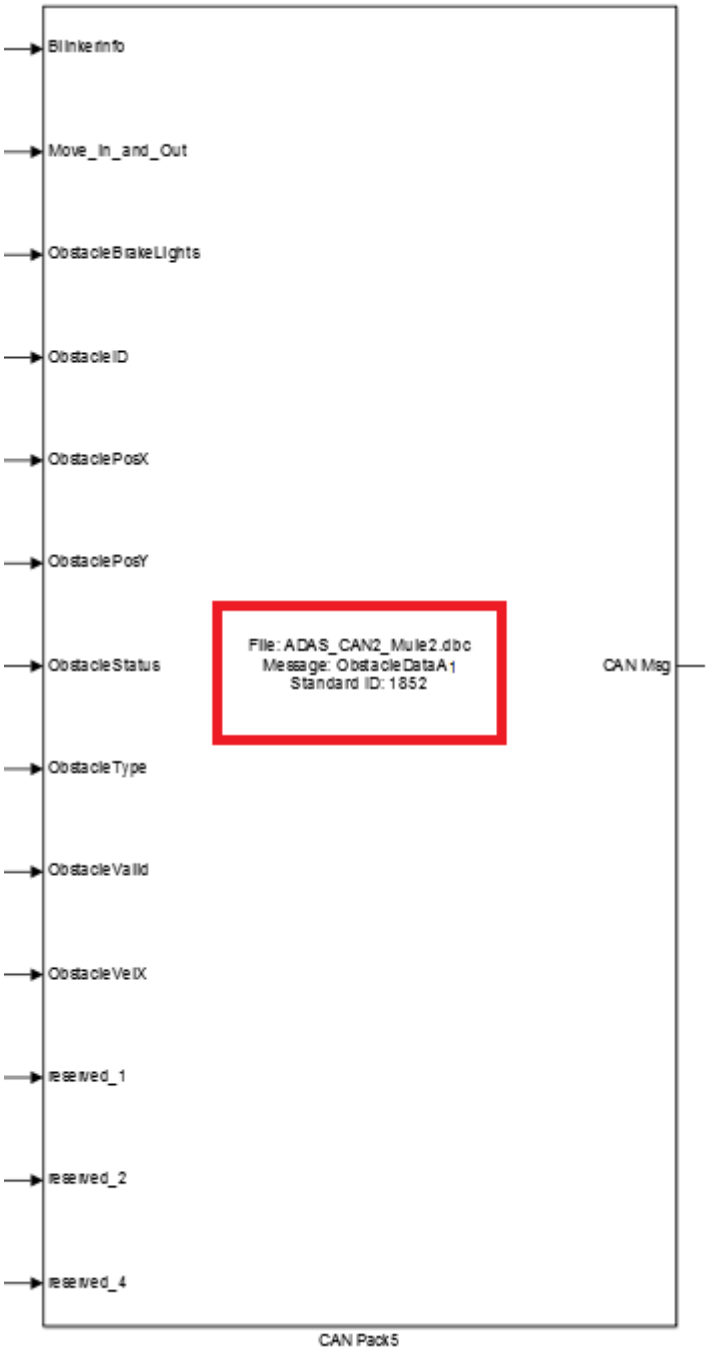


Ilustración 95. ObstacleDataA1

BlinkerInfo	-Vacío-
MoveInadnOut	-Vacío-
ObstacleBrakeLights	-Vacío-
ObstacleID	Identificador del obstáculo.

ObstaclePosX	Posición en eje X
ObstaclePosY	Posición en eje Y
ObstacleStatus	Estado del vehículo (En movimiento, parado, …)
ObstacleType	Tipo de obstáculo (persona, bicicleta, coche, camión)
ObstacleValid	Obstáculo válido
ObstacleVelX	Velocidad en eje X
Reserved_1	-Vacío-
Reserver_2	-Vacío-
Reserverd_3	-Vacío-

ObstacleDataB1-B2-…-B10

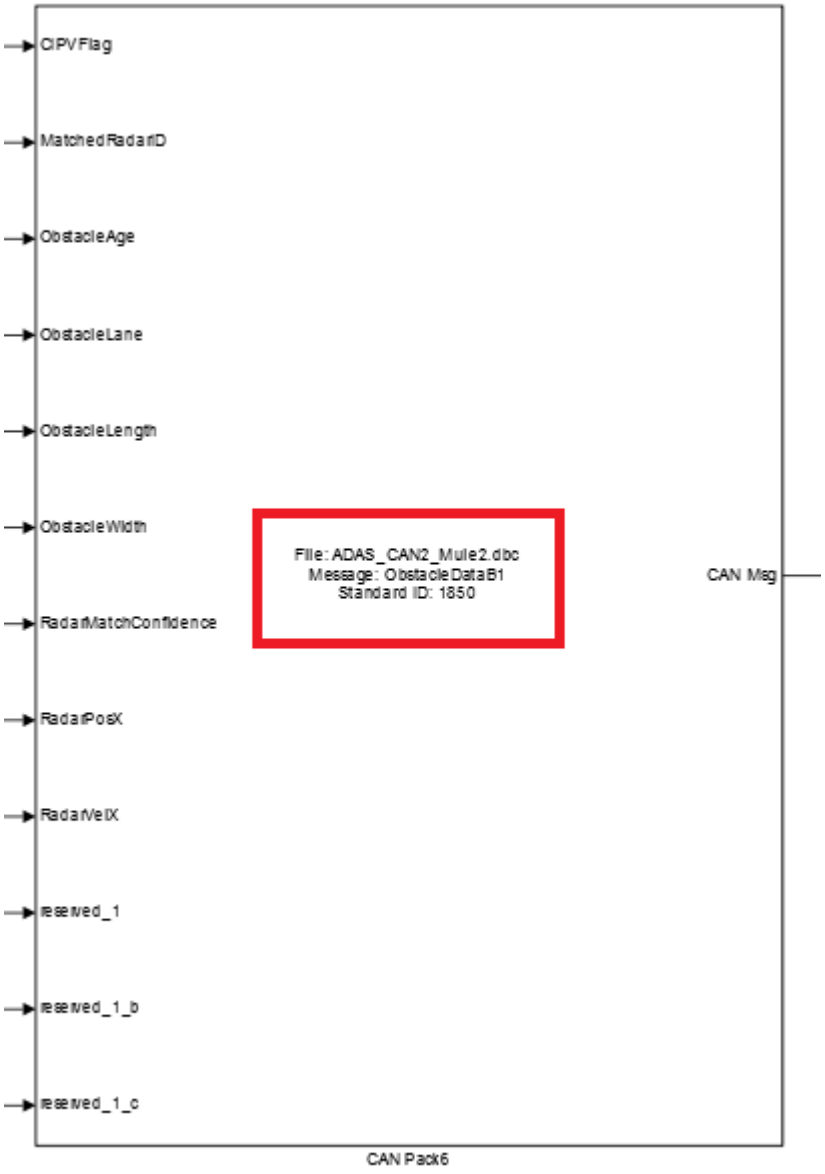


Ilustración 96. ObstacleDataB1

CIPV Flag (Closest in path vehicle)	Vehículo más próximo
MatchedRadarID	Radar emparejado
ObstacleAge	Edad del obstáculo
ObstacleLane	Carril del obstáculo
ObstacleLegth	Longitud del obstáculo
ObstacleWidth	Ancho del obstáculo
RadarMatchConfidence	Grado de confianza de radar emparejado
RadarPosX	Posición X obstáculo
RadarVelX	Velocidad X del obstáculo

Reserver_1	-Vacío-
Reserved_1b	-Vacío-
Reserved_1c	-Vacío-

ObstacleDataC1-2-…-10

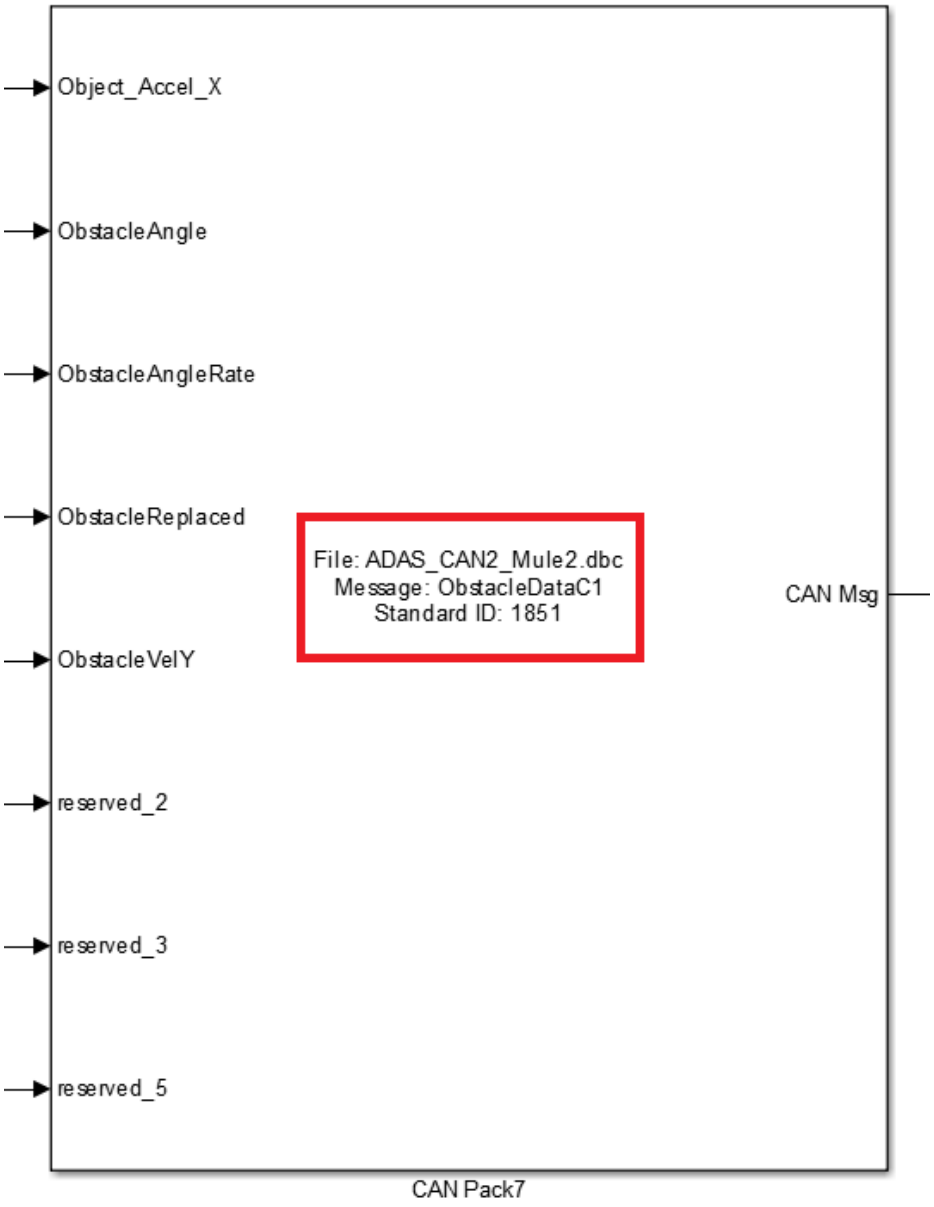


Ilustración 97. ObstacleDataC1

Object_Accel	Aceleración del obstáculo
Obstacle_Angle	Ángulo posición del obstáculo respecto al sensor
ObstacleAngleRate	Velocidad de movimiento del ángulo
ObstacleReplaced	Obstáculo sustituido
ObstacleVelY	Velocidad el obstáculo en el eje Y
Reserved_2	-Vacío-
Reserved_3	-Vacío-
Reserved_5	-Vacío-

Mensajes CAN Radar

CAN1_Obj_Status

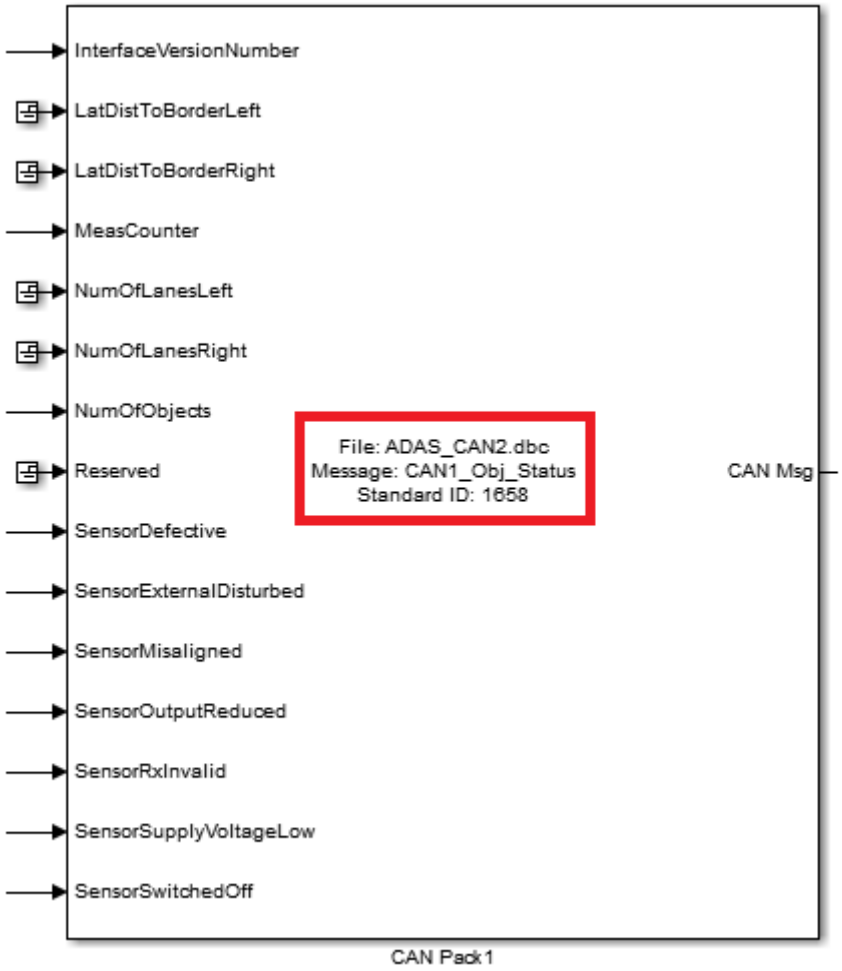


Ilustración 98. Bloque CAN1_Obj_Status

InterfaceVersionNumber	Número de versión de la interfaz
LastDistToBorderLeft	Distancia lateral al borde del carril izquierdo
LastDistToBorderRight	Distancia lateral al borde del carril derecho
MeasCounter	Contador
NumOfLanesLeft	Número de carriles por la izquierda
NumOfLanesRight	Número de carriles por la derecha
NumOfObjects	Numero de obstáculos
Reserved	Reservado
SensorDetefective	Sensor defectuoso
SensorExternalDisturbed	Señal externa perturbada
SensorMisaligned	Sensor mal alineado
SensorOutputReduced	Salida del sensor reducida
SensorRxInvalid	Sensor Inválido
SensorSupplyVoltageLow	Aportación baja de voltaje
SensorSwitchedOff	Sensor apagado

CAN1_Obj_1

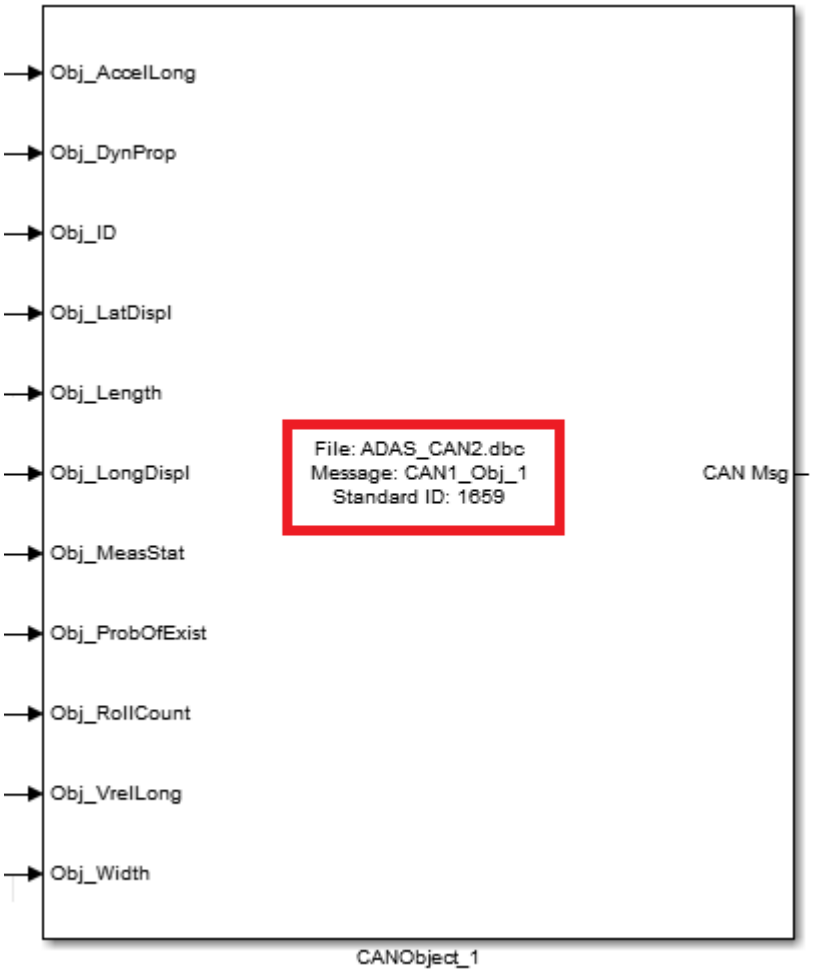


Ilustración 99. Bloque CAN1_Obj_1

Obj_AccelLong	Aceleración longitudinal relativa del obstáculo
Obj_DynProp	Estado del vehículo (Parado, en movimiento)
Obj_ID	Identificador del obstáculo
Obj_LatDispl	Distancia lateral (eje Y) del obstáculo
Obj_Length	Largo
Obj_LongDisp	Distancia longitudinal (Eje X)
Obj_MeasStat	Medida del estado
Obj_ProbOfExist	Probabilidad de existencia
Obj_RollCount	Contador
Obj_VrelLong	Velocidad relativa longitudinal
Obj_Width→ Ancho	

CAN1_Obj_2

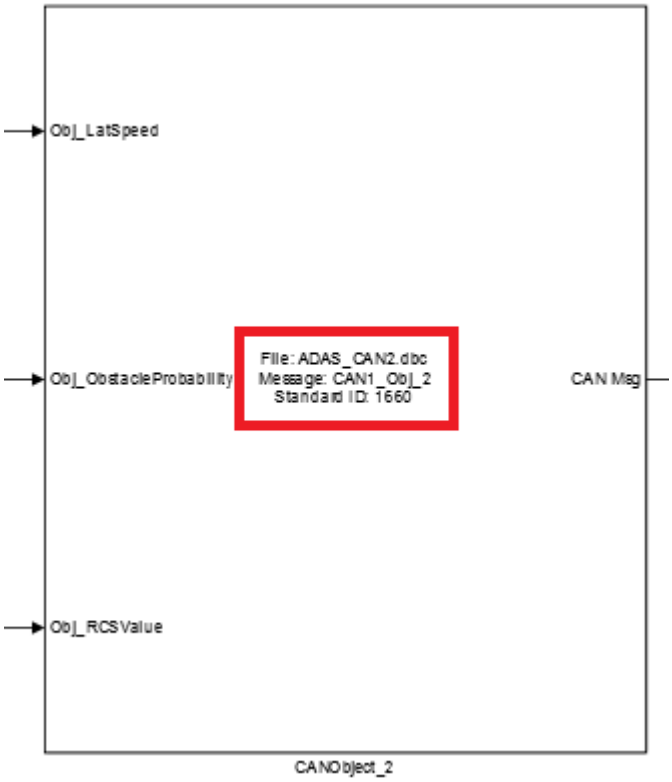


Ilustración 100. Bloque CAN1_Obj_2

Obj_LatSpeed	Velocidad Lateral Relativa (Eje Y)
Pbj_ObstacleProbability	Probabilidad del obstáculo
Obj_RCSValue	Valor RCS

Apéndice II

Bloques Adaptación Cámara

DopplerAccel

La función de este bloque es transformar la velocidad relativa sobre el eje X, para obtener la aceleración.

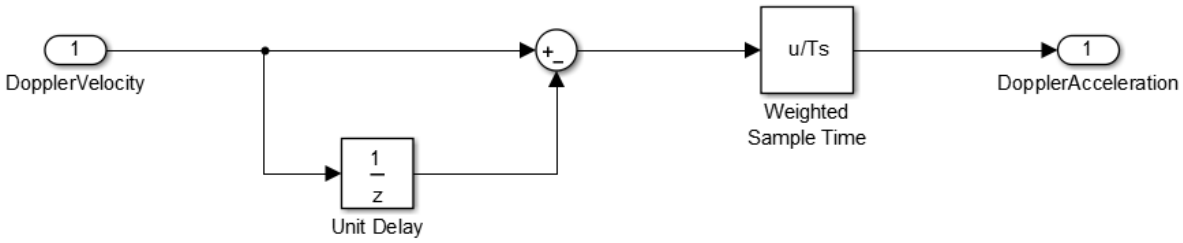


Ilustración 101. Bloque DopplerAccel (Nivel Inferior)

NumOfObjects

La función de este bloque es calcular el número de obstáculos que detecta la cámara, para ello, primero analiza el ObstacleID de cada objeto, si es distinto de 0 quiere decir que hay un obstáculo, por lo tanto, lo considera 1, finalmente suma todos los obstáculos y devuelve el valor como salida.

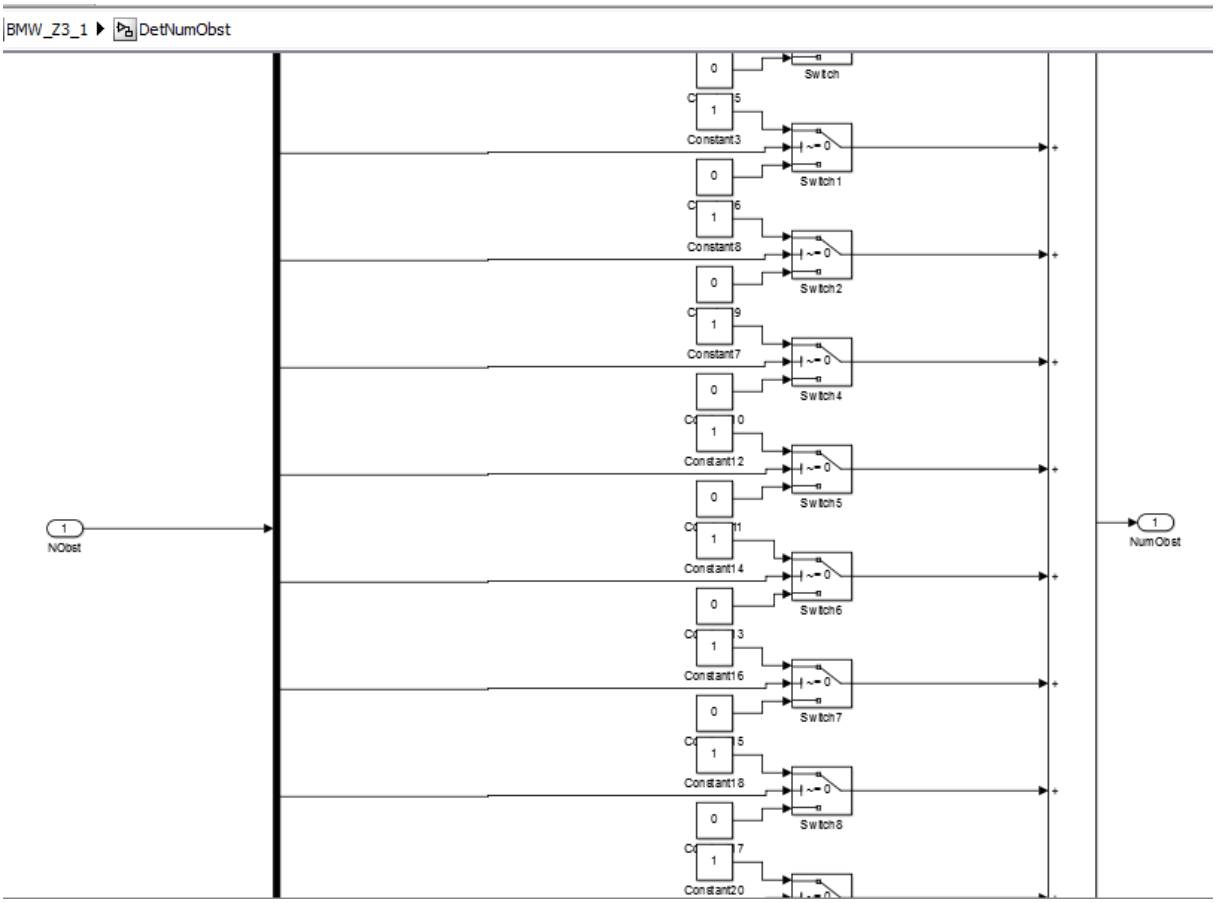


Ilustración 102. Bloque NumOfObjects (Nivel Inferior)

Bloques Adaptación Radar

Long_Lat_Range

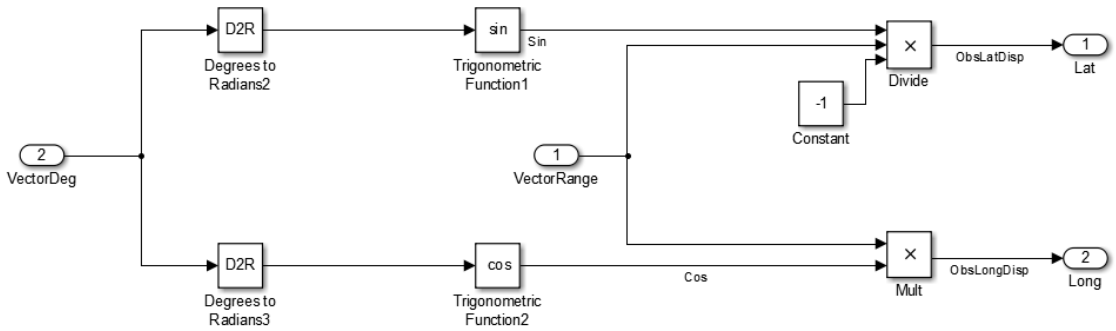


Ilustración 103. Bloque Adaptación distancia lateral y longitudinal

Partiendo de una distancia en metros y un ángulo, obtiene la posición relativa del obstáculo al sensor en el eje X e Y.

RelVelFromRange

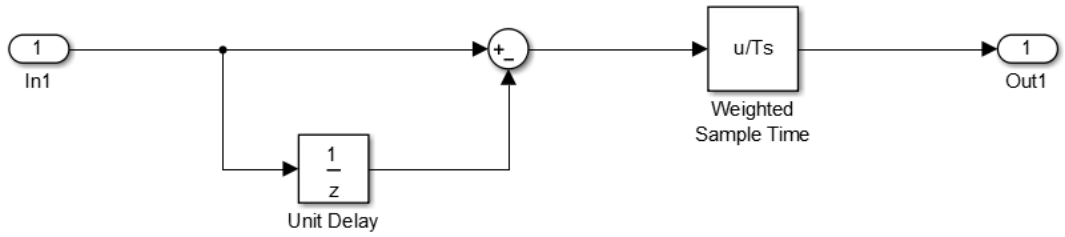


Ilustración 104. Bloque adaptación velocidad relativa

Partiendo de una distancia obtiene la velocidad relativa del obstáculo respecto al objeto.

DopplerAccel

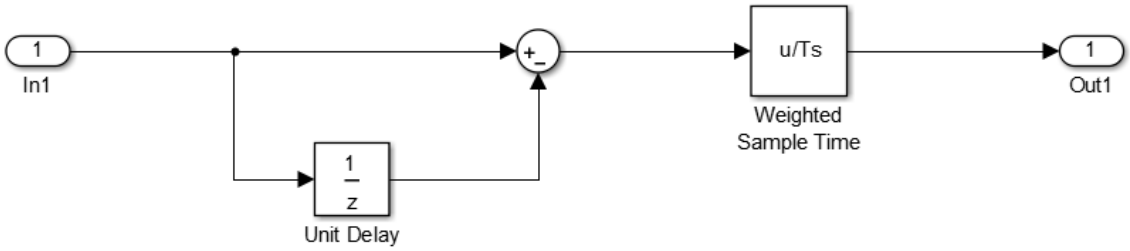


Ilustración 105. Bloque adaptación aceleración relativa

Partiendo de la velocidad relativa del obstáculo, obtiene la aceleración relativa del obstáculo respecto al sensor.

ObjectsCounter

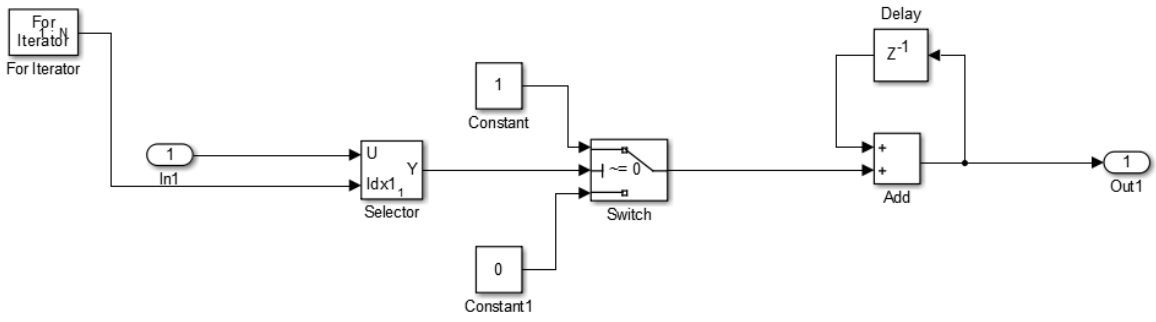


Ilustración 106. Bloque adaptación contador de objetos

Calcula el número de obstáculos detectados por el sensor.

MeasCounter

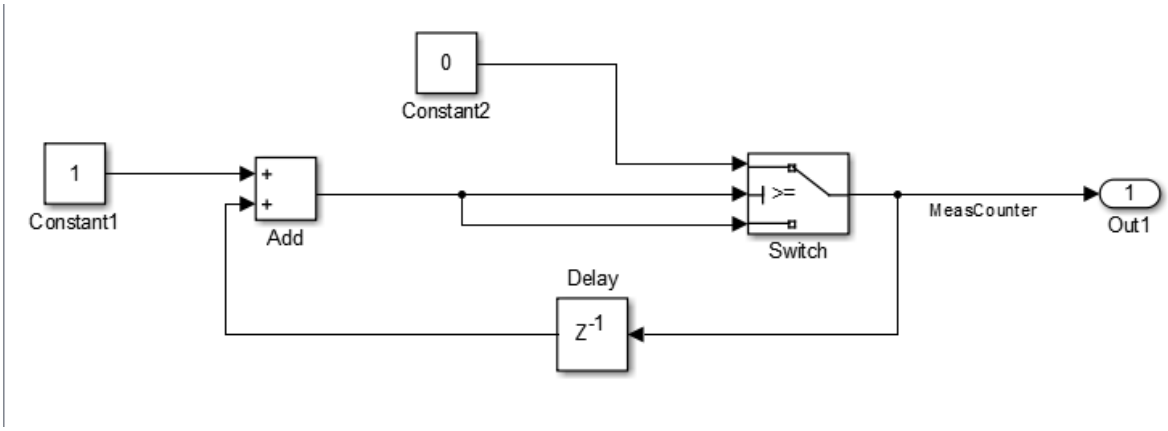


Ilustración 107. Bloque adaptación contador del frame actual

Es un contador, para saber el número de frame en el que se encuentra.

Apéndice III

Mensajes CAN Cámara configurados

La información de la cámara viene dada en un vector, donde cada posición corresponde a un obstáculo, el bloque “Selector”, extrae la información de la posición del vector especificada, de tal manera que para el bloque A1, se desea extraer la información de la primera posición del vector.

De la misma manera se hará para los obstacledata B1 y C1

Cada bloque se ha insertado en un subsistema para simplificar el modelo, como se puede observar.

ObstacleDataA1

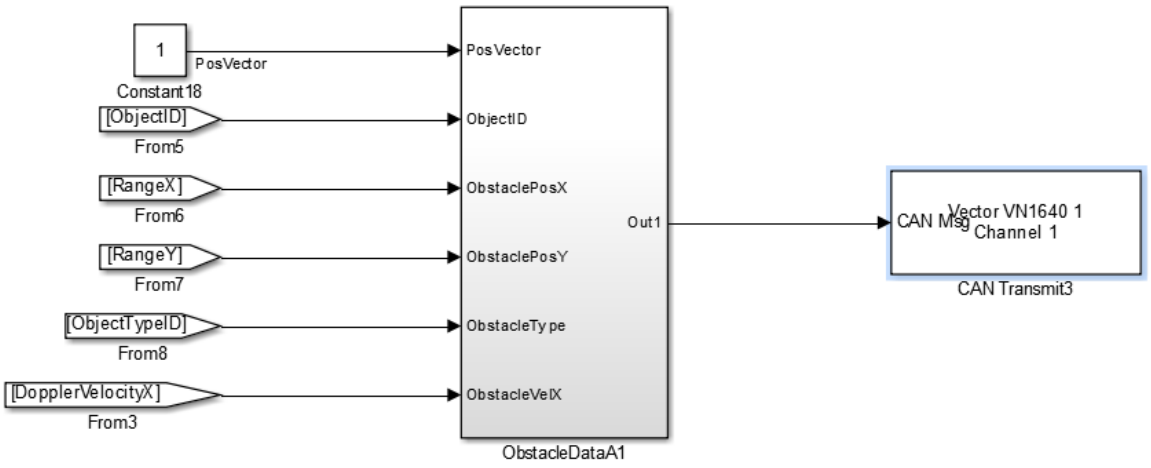


Ilustración 108. ObstacleDataA1 Configurado

Haciendo doble click sobre el subsistema:

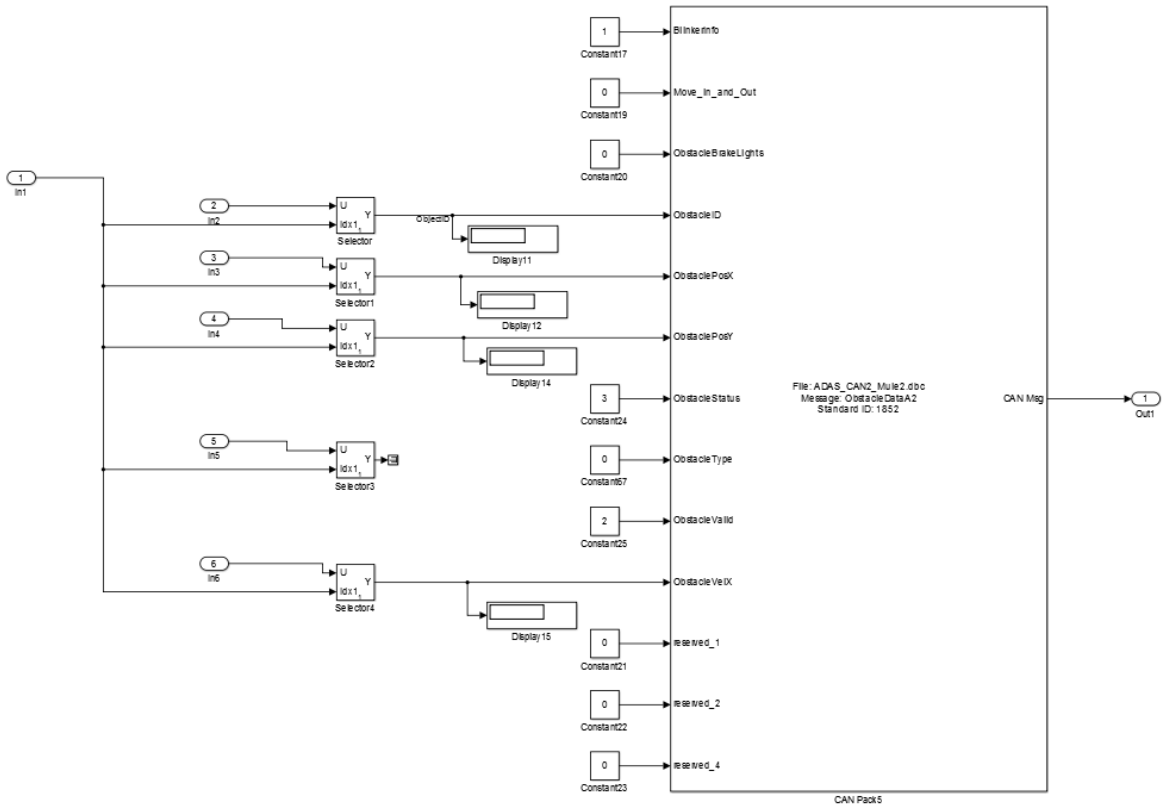


Ilustración 109. ObstacleDataA1 Configurado (Nivel Inferior)

Y ampliando la imagen:

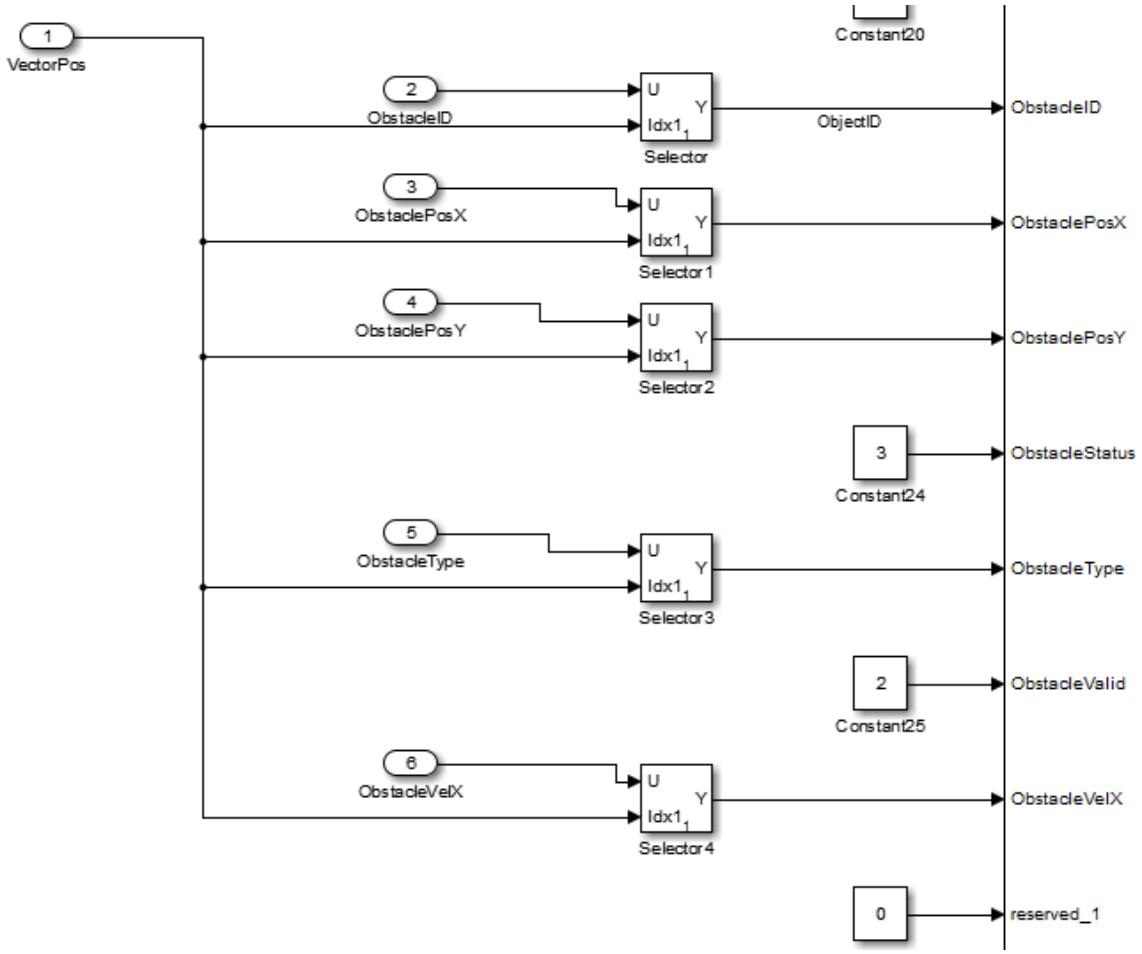


Ilustración 110. ObstacleDataA1 Configurado (Nivel Inferior Ampliado)

ObstacleDataB1

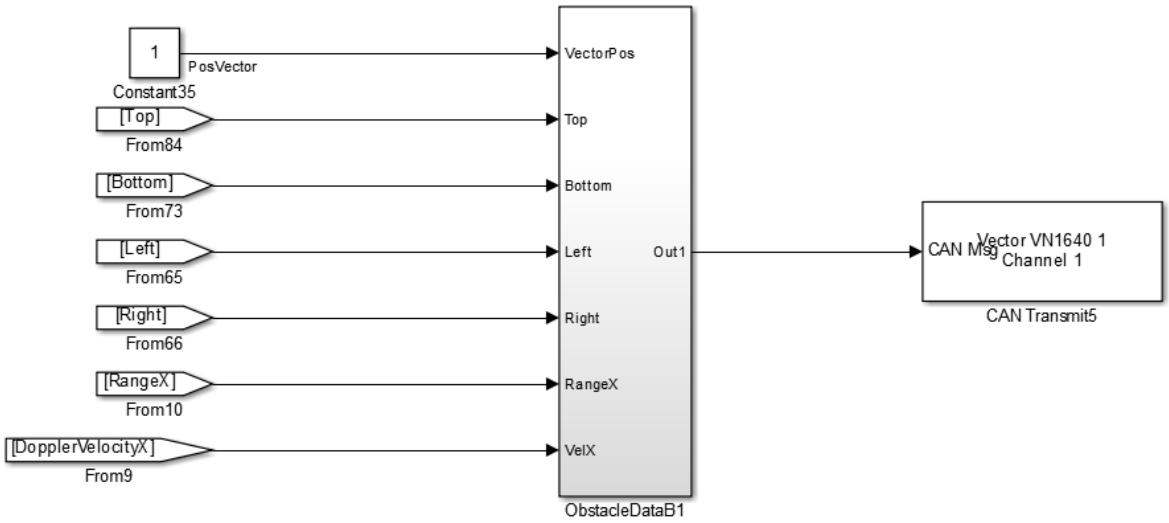


Ilustración 111. ObstacleDataB1 Configurado

Entrando en el subsistema:

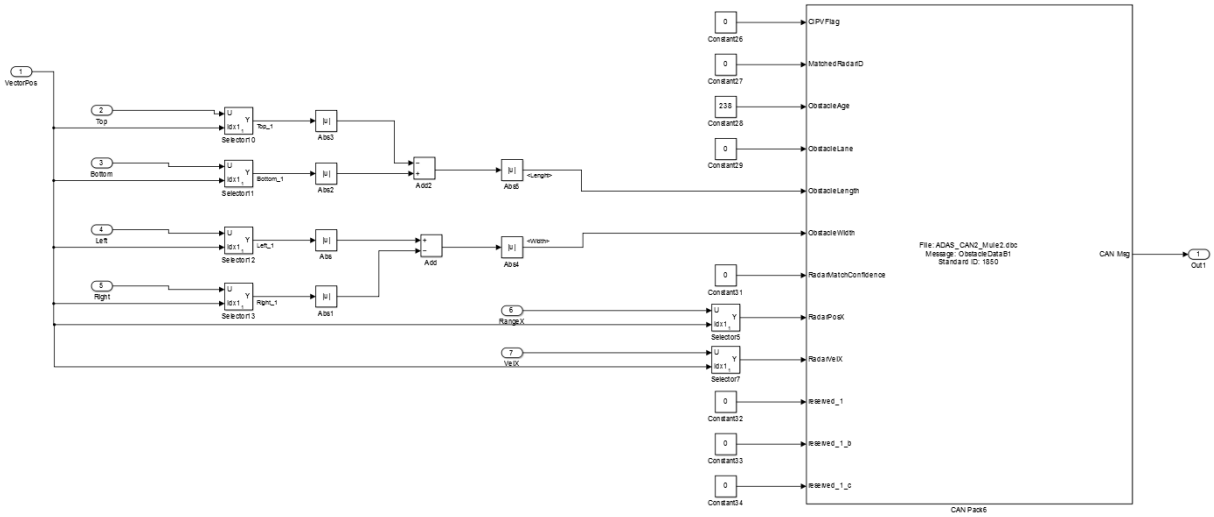


Ilustración 112. ObstacleDataB1 Configurado

Ampliando la imagen:

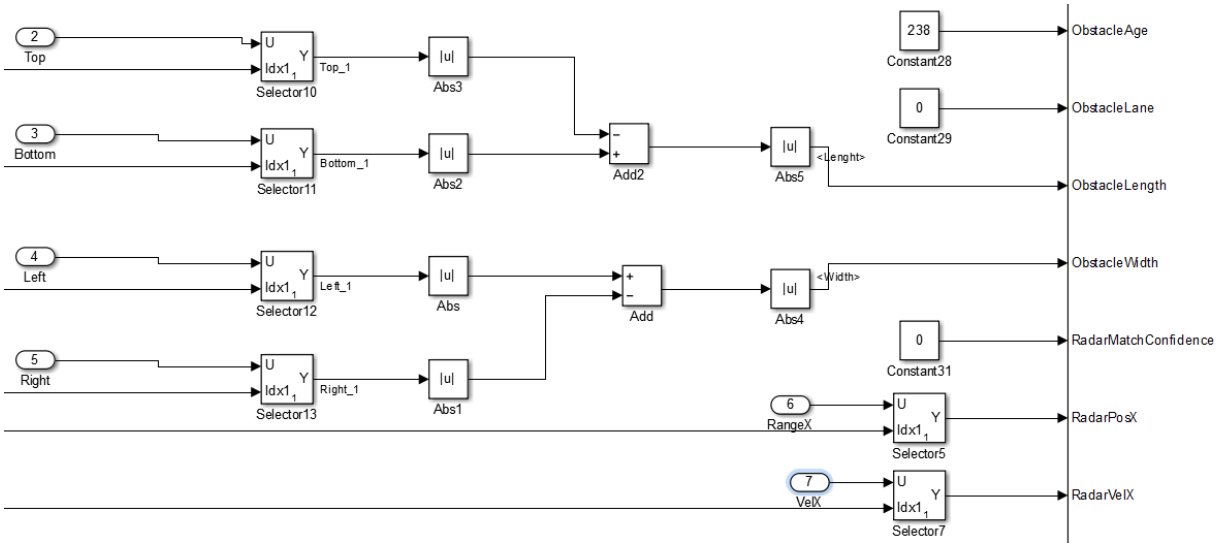


Ilustración 113. ObstacleDataB1 Configurado (Nivel Inferior)

ObstacleDataC1

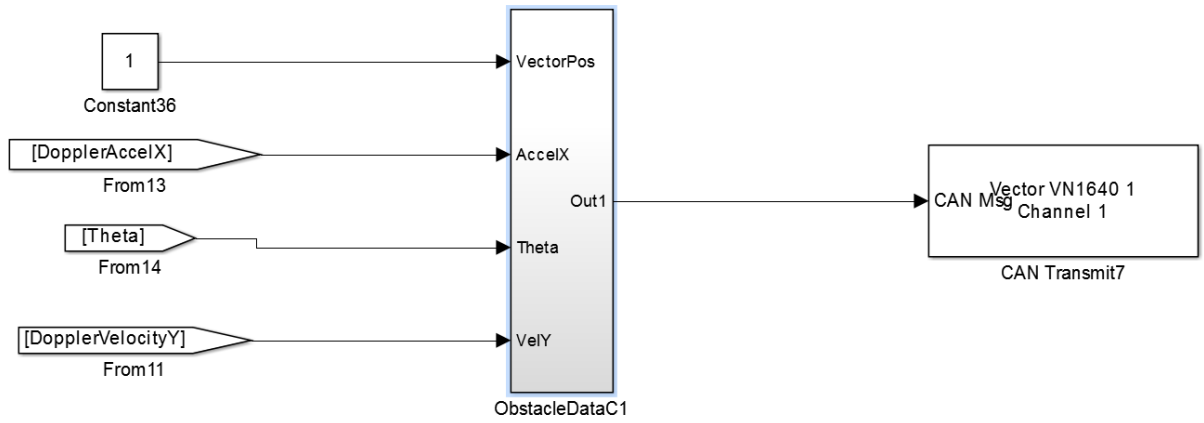


Ilustración 114. ObstacleDataC1 Configurado

Entrando en el subsistema:

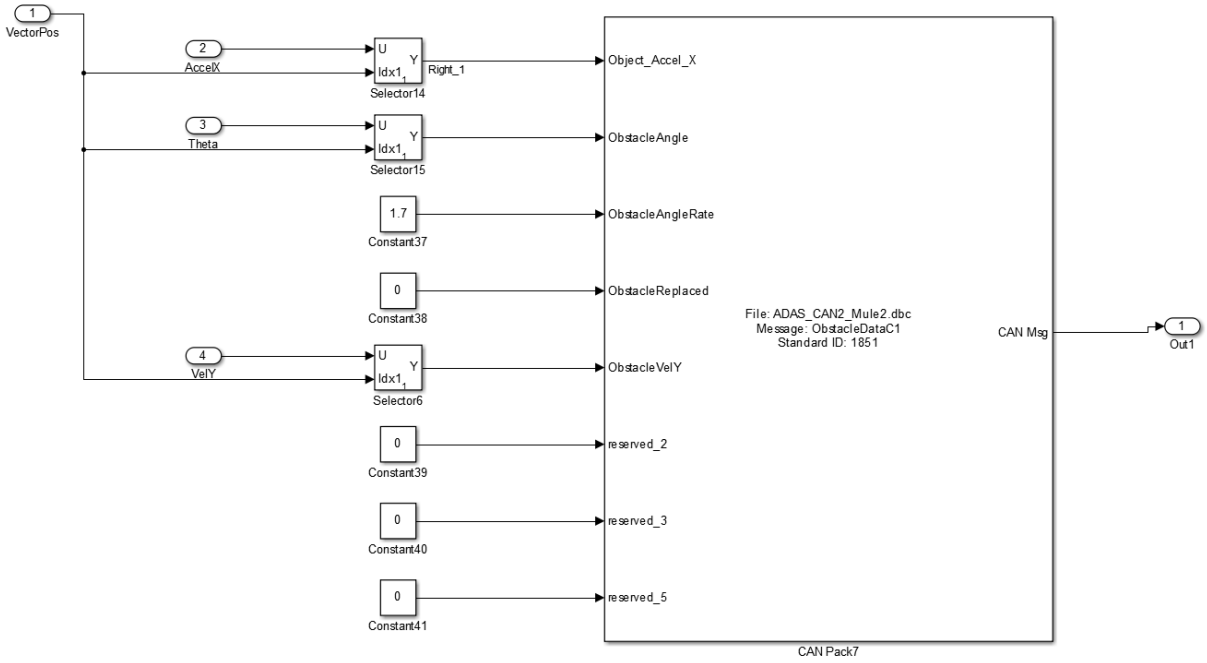


Ilustración 115. ObstacleDataC1 Configurado (Nivel Inferior)

Ampliando la imagen:

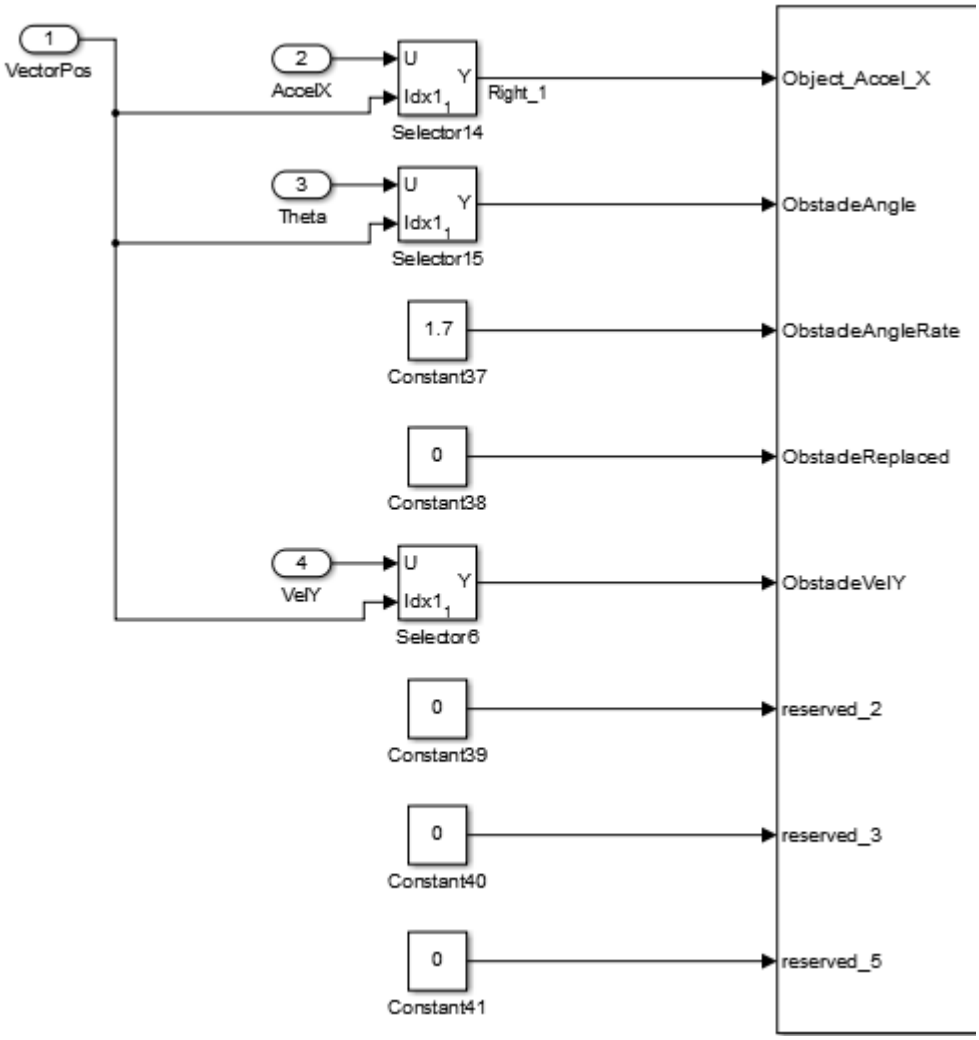


Ilustración 116. ObstacleDataC1 Configurado (Nivel Inferior)

Finalmente, todos los mensajes de la cámara quedarían ahora configurados para ser enviados por CAN.

Mensajes CAN Radar configurados

El radar se comporta de manera diferente a la cámara, ya que en el mismo barrido y con sólo 2 mensajes CAN, el sensor envía la información referente a los 40 obstáculos que es capaz de detectar.

Para simular el envío de estos mensajes, se ha utilizado un bucle For, que realiza una iteración para cada obstáculo y envía la información vía CAN, a través de los 2 mensajes correspondientes, como se puede observar más abajo.

También se ha incluido este módulo en un subsistema para facilitar su comprensión.

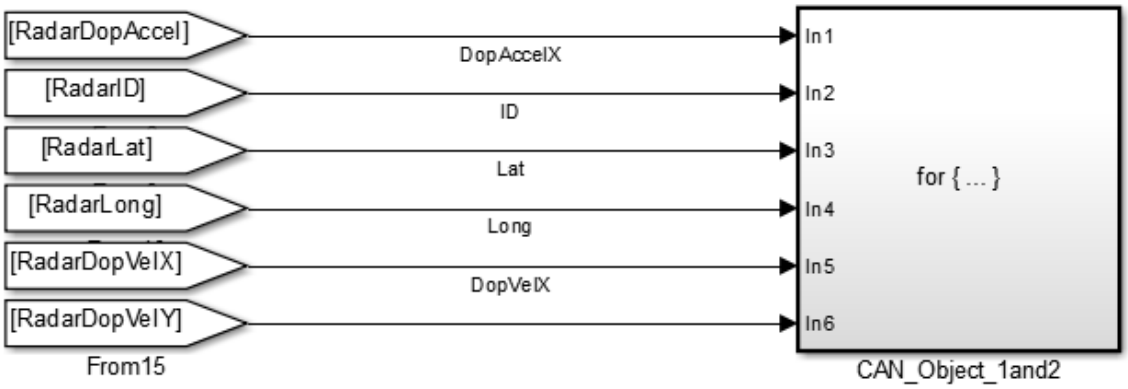


Ilustración 117. Bloques obstáculos 1 y 2 configurados

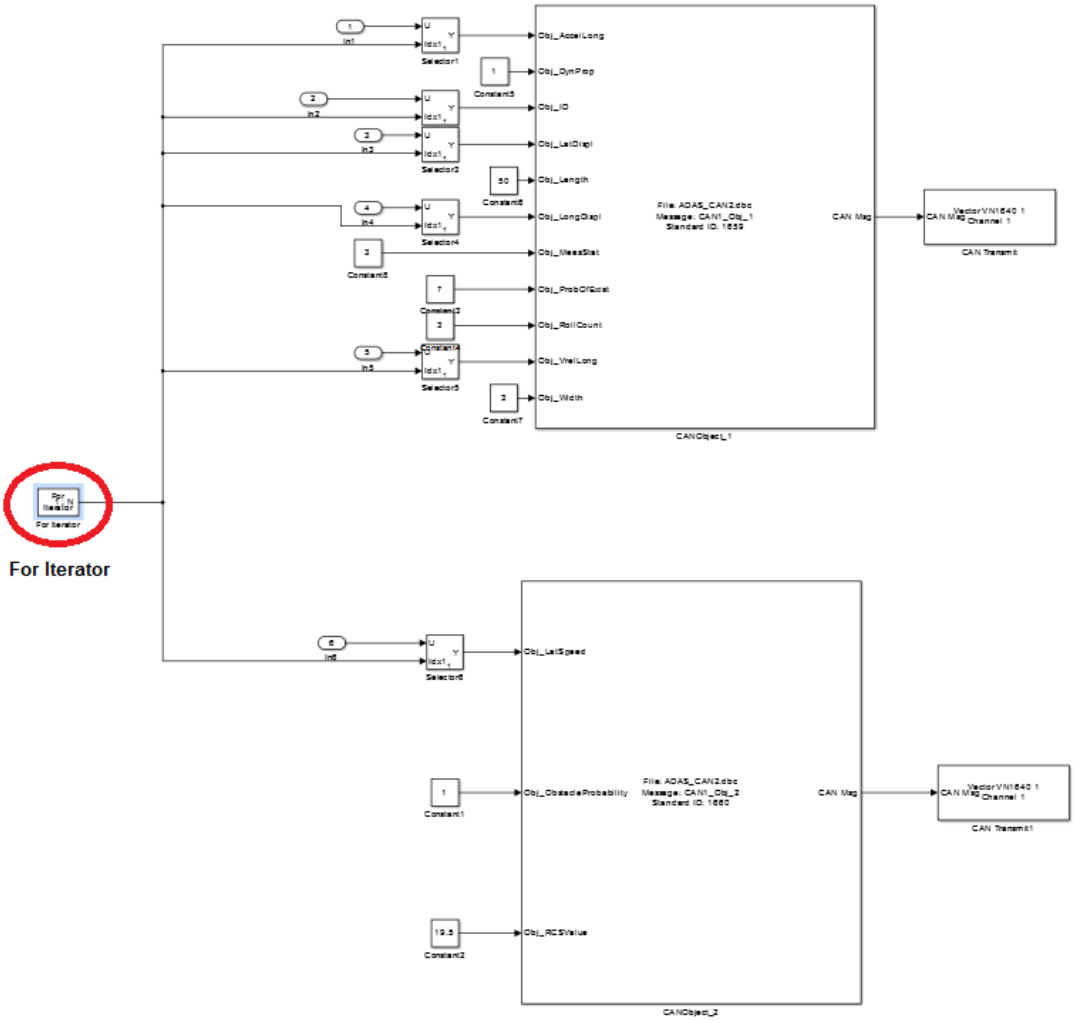


Ilustración 118. Bloques 1 y 2 configurados (Nivel inferior)

Bibliografía

Imagen Autonomous Emergency Brake. Entendiendo el Sistema de frenado de un vehículo https://www.pakwheels.com/blog/understanding-cars-braking-system/

Imagen Forward Collision Warning. Sistemas de vehículos inteligentes. http://www.eurofot-ip.eu/en/intelligent_vehicle_systems/fcw/

Imagen Adptative Cruise Control. https://www.google.es/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwiCnrn-8tnUAhWG1hoKHXKQCFkQjxwIAw&url=http%3A%2F%2Fwww.detroitnews.com%2Fstory%2Fbusiness%2Fautos%2F2015%2F03%2F03%2Fadaptive-cruise-control-growing%2F24352141%2F&psig=AFQjCNHS7OxmllqSvCA1OQ2cGkQ-udpy4w&ust=1498510765806807

MicroAutoBox II. https://www.dspace.com/en/pub/home/products/hw/micautob.cfm

PXI NI. http://www.ni.com/pxi/

CANCaseXL. https://vector.com/portal/medien/cmc/manuals/CANcaseXL_Manual_EN.pdf

PreScan. https://www.tassinternational.com/prescan

Baselabs. https://www.baselabs.de/

Simulink. https://es.mathworks.com/products/simulink.html

CANalyzer. https://vector.com/vi_canalyzer_en.html