Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Volumen I
Memoria – Apéndices

TRABAJO DE FINAL DE GRADO

APLICACIÓN PARA ANÁLISIS Y CLASIFICACIÓN DE DATOS EN ENSAYOS DE CHOQUE

TFG presentado para optar al título de GRADO en INGIENERÍA Electrónica Industrial y Automática
por **Sergi Delgado Rodríguez**

Barcelona, 11 de Enero de 2016

Tutor proyecto: Gerard Escudero Bakx
Departamento de LSI
Universitat Politècnica de Catalunya (UPC)

RESUM

L'objectiu d'aquest projecte és realitzar una aplicació de software capaç d'analitzar, classificar i extreure conclusions a partir de les dades que s'obtenen en assajos AEB "Autonomous Emergency Braking". D'aquesta manera, s'espera que el programa resulti en un estalvi d'hores de treball i esforç per part dels usuaris i, sobretot, cobreixi una necessitat en un camp on no existeixen alternatives o no es troben a l'abast de tothom. La idea és establir la base de futures aplicacions que s'adaptin a les necessitats canviants d'anàlisi d'una tecnologia en continua expansió i desenvolupament.

El programa constarà de dues parts diferenciades. La primera realitzarà el tracte de dades i els càlculs necessaris. La segona oferirà una eina interactiva per l'usuari que el permetrà visualitzar dades i resultats.

RESUMEN

El objetivo de este proyecto es realizar una aplicación de software capaz de analizar, clasificar y extraer conclusiones a partir de los datos que se obtienen en ensayos AEB "Autonomous Emergency Braking". De este modo, se espera que el programa resulte en un ahorro de horas de trabajo y esfuerzo por parte de los usuarios y, sobretodo, cubra una necesidad en un campo donde no existen alternativas o no se encuentran al alcance de todo el mundo. La idea es establecer las bases de futuras aplicaciones que se adapten a las necesidades cambiantes de análisis de una tecnología en continua expansión y desarrollo.

El programa constará de dos partes diferenciadas. La primera realizará el trato de datos y los cálculos necesarios. La segunda ofrecerá una herramienta interactiva para el usuario que le permitirá visualizar datos y resultados.

ABSTRACT

The aim of this project is to create a software application capable of analyze, classify and extract conclusions from AEB Autonomous Emergency Braking essays. In that way, the program may result, for the user, in time and work savings and, specially, it will cover a necessity in an area which have no real alternatives. The idea is to set the bases of future applications resulting in a gradual adaptation to new necessities of AEB technology, which is in constant expansion and development.

The program will have two different sections. The first one will be responsible of all the data management and results extraction. The second one will offer the user an interactive tool so as to visualize data and results.

Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Memoria

APLICACIÓN PARA ANÁLISIS Y CLASIFICACIÓN DE DATOS EN ENSAYOS DE CHOQUE

TFG presentado para optar al título de GRADO en INGIENERÍA Electrónica Industrial y Automática
por **Sergi Delgado Rodríguez**

Barcelona, 11 de Enero de 2015

Tutor proyecto: Gerard Escudero Bakx
Departamento de LSI
Universitat Politècnica de Catalunya (UPC)

ÍNDICE MEMORIA

Capítulo 1: Introducción 3
1.1. Motivaciones para la realización del proyecto 4
1.2. Objetivos 5
1.3. Visión de conjunto de la memoria 6
Capítulo 2: DESCRIPCIÓN DEL PROBLEMA 7
2.1. Análisis del problema 7
2.2. Herramientas de evaluación y protocolo 8
Capítulo 3: Especificaciones de la aplicación 17
3.1. Estructura de la aplicación 17
3.1.1. Bloque de Cálculo 17
3.1.2. Bloque de Interface 18
Capítulo 4: Estado del arte 19
4.1. Productos y aplicaciones presentes en el mercado 19
4.1.1. CANOE 20
4.1.2. ADTF 20
4.2. Evaluación de herramientas disponibles 21
4.2.1. Excel 21
4.2.2. DIADEM 21
4.2.3. Matlab 22
Capítulo 5: Configuración del sistema 23
5.1. Adquisición y formato de datos de entrada 23
5.1.1. Matriz REF 24
5.1.2. Matriz EGO 26
5.1.3. Matriz FKT 27
5.2. Formato de datos intermedios y de salida 29
5.2.1. Matriz textdata 29
5.2.2. Matriz colheaders 30
5.2.3. Matriz data 31
5.2.4. Vector M 31
5.2.5. Vector Mh 33

5.2.6. Matriz MM 34
5.2.7. Vector MMean 34
5.2.8. Vector MDisp 34
5.2.9. Vector MMeanh 35
Capítulo 6: Arquitectura del sistema 37
6.1. Arquitectura general del sistema 37
6.2. Arquitectura de cálculo 39
6.2.1. Cartel de presentación 42
6.2.2. Cargar archivos .csv de entrada 42
6.2.3. Cálculo de puntos de interés 43
6.2.4. Definición de variables booleanas 43
6.2.5. Creación de la matriz de resultados 44
6.2.6. Predicciones y evaluación 47
6.2.7. Secuencia de guardado 48
6.2.8. Ejecución de la interface GUIDE 49
6.3. Arquitectura de Interface Usuario 50
6.4. Interconexiones y comunicación 55
Capítulo 7: Planificación 57
7.1. Planificación temporal del desarrollo 57
7.2. Diagrama temporal final 59
Capítulo 8: Costes de desarrollo 61
Capítulo 9: Conclusiones 63
9.1. Evaluación de objetivos 63
9.2. Trabajos de futuro 65
- Mejoras en la visualización 66
- Detección de valores anómalos 66
- Creación automática de reportes 66
- Uso del programa para usuarios sin Matlab 67
- Almacenamiento de entornos de trabajo 67
Referencias bibliogáficas 69

CAPÍTULO 1: INTRODUCCIÓN

El primer automóvil equipado con un motor de combustión interna data de 1886 y fue diseñado por Karl Friedrich Benz en la ciudad de Manheim, Alemania. Este modelo apenas alcanzaba los 20 km/h. Desde ese momento la tecnología del sector del automóvil empezó a crecer y desarrollarse en busca de la producción en cadena, los motores más eficientes, los procesos y materiales más adecuados y económicos, el confort del usuario, llegando incluso a la inserción de elementos electrónicos como las interfaces de comunicación.

Sin embargo, ya en la década de 1920 se hizo patente el peligro que este invento traía intrínsecamente a las calles y carreteras del mundo, un riesgo cada día más presente al mismo tiempo que los motores aumentaban en potencia y el número de vehículos en circulación crecía. Durante ese periodo, aparecieron las primeras medidas de seguridad, como la iluminación eléctrica, o el diseño de parabrisas contra herida en caso de rotura del cristal. En 1956 Ford incluyó por primera vez el conocido cinturón de seguridad, importado de la industria de la aviación, que hoy en día es obligatorio en muchos países alrededor del mundo.

En las últimas décadas la seguridad en el automóvil ha evolucionado a pasos agigantados, en especial dentro del marco de la Seguridad Pasiva, aquellos elementos o sistemas destinados a minimizar los daños que se producen cuando un accidente es inevitable, principalmente los sufridos por los ocupantes del vehículo.

No obstante, en los últimos años y gracias a las emergentes tecnologías como radares y cámaras, aparece un nuevo concepto de seguridad, la Seguridad Activa, destinada a ir más allá y evitar los accidentes antes de que se produzcan. Son parte primordial los sistemas AEB (Autonomous Emergency Breaking), capaces de detectar una posible colisión y tomar el control de los frenos del vehículo para evitarla.

La relativa novedad de estos métodos y la gran cantidad de ensayos que requieren han hecho que muchos fabricantes de automóviles se encuentren con enormes volúmenes de datos por etiquetar, calificar y analizar y una ausencia de software específico con el que realizar estas tareas sin invertir largos periodos de tiempo.

En el contexto actual, la sociedad está cada día más concienciada de los peligros de la conducción, de forma que el cliente prima la seguridad a la hora de adquirir un vehículo. Las marcas, conscientes de este hecho, invierten capital y tiempo en dotar a sus productos de sistemas como el AEB con mejores prestaciones que sus competidores. Además los organismos reguladores y calificadores de los sistemas de seguridad en el automóvil exigen cada día mayores prestaciones. En definitiva, ninguna marca puede permitirse obviar estas nuevas medidas de seguridad a riesgo de perder ventas y perjudicar su imagen.

1.1. Motivaciones para la realización del proyecto

La motivación principal para este proyecto es desarrollar las habilidades necesarias para realizar un trabajo real con la responsabilidad de cumplir una serie de objetivos. Además, se encuentra estrechamente relacionado con la titulación que su autor cursa actualmente, Ingeniería Electrónica Industrial y Automática. El objetivo del proyecto es desarrollar la base de futuras aplicaciones que permitan el etiquetado, la calificación y el análisis de datos obtenidos en pruebas de campo de AEB.

Dado que estas tecnologías se encuentran actualmente en auge, cada vez son más las posibilidades que ofrecen y las funciones que se quieren implementar en los nuevos vehículos. Los posibles escenarios de accidentes que pueden contemplarse (velocidades de colisión, cuerpos implicados, condiciones climatológicas,…) son tan grandes que, para un usuario, llevar el seguimiento ensayo a ensayo sin herramientas que faciliten su tarea puede resultar imposible o muy costoso en horas y esfuerzo. Por este motivo el desarrollo de aplicaciones es necesario dentro del sector y este es el mejor momento para trabajar en ellas, ahora que las tecnologías AEB llevan relativamente poco tiempo en las calles.

A nivel personal para el autor, este proyecto proporciona la oportunidad de afianzar los conocimientos de trato de datos y programación en base matemática, así como la programación orientada a objetos. Todo ello para realizar una herramienta real y útil a la empresa. Al mismo tiempo, representa una inserción en el mundo laboral, viendo el funcionamiento de este, obligando al autor a mejorar sus habilidades comunicativas, de trabajo en equipo y trabajo bajo la supervisión de una empresa global, no solo en su propio beneficio. Todo este aprendizaje se considera de vital importancia en el futuro como ingeniero del autor.

1.2. Objetivos

El principal objetivo de este proyecto es el desarrollo de una aplicación capaz de computar grandes matrices de datos obtenidas durante los ensayos AEB. Esta herramienta debe calcular y proporcionar distintos valores típicos y necesarios, por medio de los cuales se evalúa el correcto funcionamiento de los sistemas AEB ensayados.

Por medio de una interfaz el usuario será capaz de visualizar los datos obtenidos en forma de gráfico, sobre el cual podrá buscar valores concretos, visualizar señales superpuestas, realizar ampliaciones de secciones y representar de forma sencilla el instante en que se activa el aviso de colisión y el instante en el que el vehículo empieza a frenar, puntos clave del análisis de los datos.

Además, en forma de tabla, podrá ver valores típicos de las curvas representadas, tales como los puntos máximo y mínimo de estas.

Por otro lado, la aplicación debe recoger todos los datos que provienen de distintas fuentes y clasificarlos en función del ensayo al que pertenecen. En cada ensayo debe calcular aquellos puntos, intervalos o equivalencias que aportan información sobre el ensayo y sacar un mínimo de conclusiones sobre lo acontecido en este. Todos estos valores deben agruparse en un vector, y posteriormente, al incluir más ensayos, una matriz de resultados, por medio de la cual se puedan comparar unos con otros y se puedan almacenar en un formato más cómodo.

Considerando todos los aspectos hasta el momento comentados, los puntos más relevantes en el desarrollo de la aplicación son:

- **Automatización de análisis de ensayos AEB:** objetivo principal del proyecto, dada la falta de software específico a tal fin y el tiempo que requiere el análisis manual.
- **Interfaz de usuario:** permitiendo que cualquier usuario pueda utilizar el software de forma sencilla y sin necesidad de acceder a la programación.
- **Visualización previa de los datos:** es importante que el usuario pueda acceder a una visualización simple de cada magnitud adquirida durante el ensayo para realizar hipótesis y detectar anomalías antes del análisis más profundo.
- **Facilidad de ampliación:** el programa debe tener una estructura modular que permita la aplicación de funciones de forma sencilla. Este punto es muy importante dada la constante innovación en el área.
- **Adaptación a los formatos necesarios:** Como se puede apreciar en el capítulo 2, son muchas las condiciones de formato y requerimientos de software específico para este proyecto que vienen impuestos por las necesidades de la empresa que lo acoge.

- **Herramienta centralizada:** Esta herramienta debe ser propia de la empresa que invierte en ella y por tanto es importante limitar el acceso o la funcionalidad de esta si fuera usada por entidades externas.

1.3. Visión de conjunto de la memoria

Este proyecto se estructura en capítulos enfocados a mostrar al lector como se ha desarrollado el producto final. Cabe destacar que este producto se divide en dos partes claramente diferenciadas, los programas de cálculo, etiquetado y gestión de datos y la interfaz gráfica de comunicación con el usuario. Estos dos sectores deben comunicarse entre sí pero se diferencian entre ellos a lo largo de toda la memoria.

En el capítulo 2, se detallan las especificaciones que, en un inicio, se establecen para el producto por parte del autor y la empresa que lo solicita, tanto a la hora de elegir el software de desarrollo y los formatos de soporte como las características de la aplicación en sí misma.

En el capítulo 3, *Estado del arte*, se presenta un repaso por los productos similares que existen actualmente en el mercado, recalcando el hecho de que se trata de software específico de cada empresa y raramente se ponen a disposición de otros usuarios. También se hace un repaso a la tecnología y protocolos actuales dentro del sector de AEB, algunos de los cuales se tienen en cuenta para este proyecto.

A continuación se presenta la planificación que se siguió a lo largo de la realización del proyecto y la forma final con los tiempos reales que se sucedieron.

Durante los siguientes capítulos, se detalla la configuración del sistema, parametrizando las matrices de datos iniciales y finales, las secciones del sistema, tanto de la parte de cálculo como de la interfaz usuario, incluyendo la conectividad entre ambas y los métodos empleados.

El capítulo 7, evalúa los costes de desarrollo y también las previsiones de ahorro, temporal y económico, derivadas de los resultados finales.

Como último punto, se extraen las conclusiones obtenidas al finalizar el proyecto. Se evalúan los métodos empleados, el enfoque que se ha dado a la resolución de cada objetivo y si se han cumplido o no las expectativas iniciales. Posteriormente se recogen todos aquellos trabajos u opciones que pueden considerarse en el futuro y que representan ampliaciones y mejoras para el programa base.

CAPÍTULO 2: DESCRIPCIÓN DEL PROBLEMA

2.1. Análisis del problema

Antes de definir las características del programa es necesario entender como funcionan los sistemas AEB que se van a analizar.

Los automóviles que presentan un sistema de frenado autónomo, incorporan un radar que les transmite información de su entorno inmediato. Este radar lee las distancias a los cuerpos cercanos al coche y las velocidades de todos ellos, de forma que es capaz de prever que el vehículo está en riesgo de colisionar con otro cuerpo y le transmite esta información, iniciando el proceso de frenado autónomo de emergencia.

A modo de resumen, el sistema AEB, una vez se le transmite el aviso de colisión, activa una señal visual y sonora de aviso al conductor que recibe el nombre de "warning" y que se captura como un flanco de subida en su canal correspondiente. Al mismo tiempo, se inicial el pre-llenado del freno para que el conductor, en caso de reaccionar a tiempo, pueda frenar de forma más eficaz. Si persiste el peligro, se activa la señal conocida como TBR. Esta señal activa el freno automático y mientras está activa, genera un perfil de frenada que el coche debe seguir para evitar la colisión. Si la reducción de la velocidad derivada de este perfil resulta insuficiente se activa la señal conocida como ZBR, que consiste en una señal parecida a TBR y que aumenta muy rápidamente la exigencia del perfil de frenada como último recurso para evitar la colisión.

Estas señales son las más importantes a la hora de realizar un ensayo, pero existen una enorme cantidad de señales interesantes para analizar, como las variables internas que hacen posibles todos estos pasos.

También durante los ensayos se emplean GPS diferenciales que ayudan a la evaluación de los sistemas que incluye el vehículo y que generan gran cantidad de datos.

Así pues, en relación a esta pequeña explicación, las características necesarias para el programa son las siguientes.

- Debe reconocer todas las señales que se producen en un ensayo AEB, tanto si son estándar como si no se usan por el momento, pero pueden llegar a usarse en el futuro.

- Los datos se obtienen de programas de captura en tiempo real. La comunicación más eficiente con estos programas se realiza por medio de la exportación de los datos en formato .csv. Este debe ser por tanto el formato de entrada de los ensayos.

- La herramienta de desarrollo debe soportar todos los requisitos establecidos para el programa. Además, es conveniente que se trabaje con un software que ya se utilice en la empresa, para agilizar los procesos de compra de licencias, instalación de módulos y ejecución del programa desde otros terminales.

- Los datos obtenidos, además de ser visualizados en la interface de usuario, deben exportarse automáticamente en formato .xlsx con una edición adecuada para la creación de informes y el almacenamiento de datos.

- Dado el carácter privado y confidencial de la aplicación para su uso por el cliente, los datos que se introducen deben ser indexados con sus nombres internos que solo se conocen en la empresa. De ese modo, solo alguien que conozca los procesos internos del programa podría adaptarlo a su uso en otras compañías o ámbitos distintos.

2.2. Herramientas de evaluación y protocolo

Como se ha comentado anteriormente, la mayoría de las marcas del sector han desarrollado sus propias pruebas y protocolos de ensayo en el ámbito del AEB para evaluar, tanto la actuación de los sistemas de radar y cámaras, como el comportamiento de sus vehículos en situaciones reales. De hecho, cada marca da prioridad a parámetros distintos y realizan predicciones y análisis centrados en aquella información extraída de los ensayos que consideran de mayor utilidad.

Tanto es así, que cada marca mantiene en secreto sus protocolos y procedimientos. La empresa que encarga este proyecto no es una excepción y por tanto, gran parte de los criterios de evaluación de resultados que utiliza y que se aplicarían gracias a las matrices de datos obtenidas con este proyecto no pueden mostrarse en este documento.

Sin embargo, existe un organismo independiente de evaluación de seguridad automovilística que publica sus protocolos de forma pública y cuyos test forman parte del volumen de ensayos que gran parte de los fabricantes realizan para sus modelos.

Se trata del "European New Car Assessment Programme", más conocido por sus siglas Euro NCAP. Es un programa de seguridad para automóviles con sede en Bélgica. Se fundó en 1997 por el Laboratorio de Transporte e Investigación dentro del Departamento de Transporte del Reino Unido. Actualmente recibe el soporte de varios gobiernos europeos, muchos fabricantes importantes de vehículos y otras organizaciones del sector del automóvil.

Por medio de un sistema de evaluación propio y en constante evolución, realiza test tipificados a los nuevos coches del mercado y les otorga una puntuación. El objetivo es motivar a los fabricantes de automóviles a mejorar el rendimiento de todas las áreas de protección de pasajeros y peatones, tanto niños como adultos. El resultado de estos test se publica finalmente siguiendo un sistema de puntuación por estrellas fácil de interpretar para los potenciales clientes.

Figura 2.1. *Logo oficial del European New Car Assessment Programme (Bibliografía 2).*

El sistema de puntuación por estrellas de Euro NCAP va de 1 a 5 estrellas, siendo 5 la mejor puntuación. En el cálculo de este valor tiene relevancia la tecnología de prevención de colisiones, objeto de este estudio. (Extraído de bibliografía 2).

ESCENARIOS EURO NCAP

El protocolo "Euro NCAP AEB test protocol v101" define los ensayos de AEB city y AEB Inter-Urban.

Los escenarios establecen como el vehículo a ensayar debe acercarse a un objetivo a distintas velocidades constantes. Al detectar dicho objetivo, debe detenerse o reducir al máximo su velocidad antes de colisionar con este.

El objetivo (EVT, "Euro NCAP Vehicle Target") es una réplica blanda de la parte trasera de un coche, con los faros, bandas reflectantes y geometría propia de este. El "target" ha de ser detectado por cámaras y radares como detectarían un vehículo real, y por tanto, todas sus medidas han sido tipificadas y protocoladas.

Figura 2.2. *Objetivo o target EVT preparado para ensayos Euro NCAP. (Bibliografía 9).*

En función de la posición del EVT se clasifican los ensayos Euro NCAP con las siguientes nomenclaturas:

- **CCRs (Car-to-Car Rear Stationary) scenario**

El EVT se encuentra quieto en la trayectoria del vehículo ensayado. Las velocidades van desde 10 a 50 km/h para ensayos AEB City y entre 30 y 80 km/h para ensayos AEB Inter-Urban, con intervalos de 5 o 10 km/h.

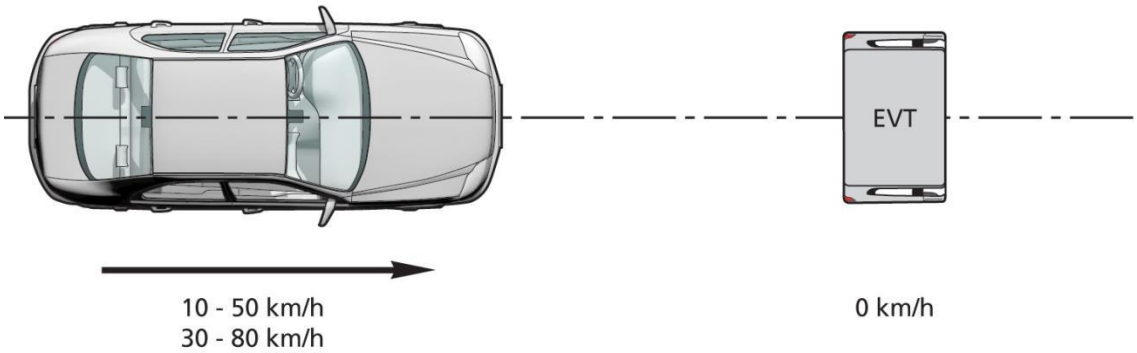


Figura 2.3. *Esquema de ensayo CCRs. (Bibliografía 9).*

- **CCRm (Car-to-Car Rear Moving) scenario**

El EVT se encuentra en la trayectoria del vehículo ensayado desplazándose a una velocidad constante de 20 km/h, siempre menor a las velocidades de ensayo, que están comprendidas entre los 30 y los 80 km/h con intervalos de 10 km/h.

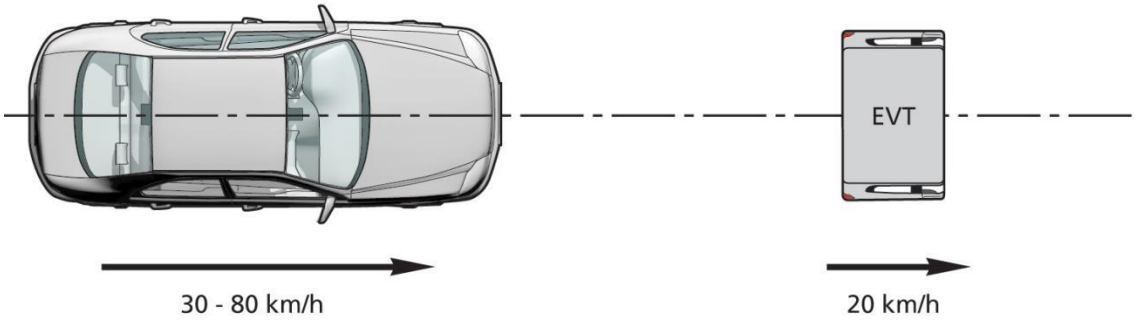


Figura 2.4. *Esquema de ensayo CCRm. (Bibliografía 9).*

- **CCRb (Car-to-Car Rear Breaking) scenario**

El EVT, inicialmente a 50 km/h, reduce su velocidad en 2 o 6 m/s2. El vehículo se acerca al objetivo a 50 km/h desde una distancia de 12 o 40 m.

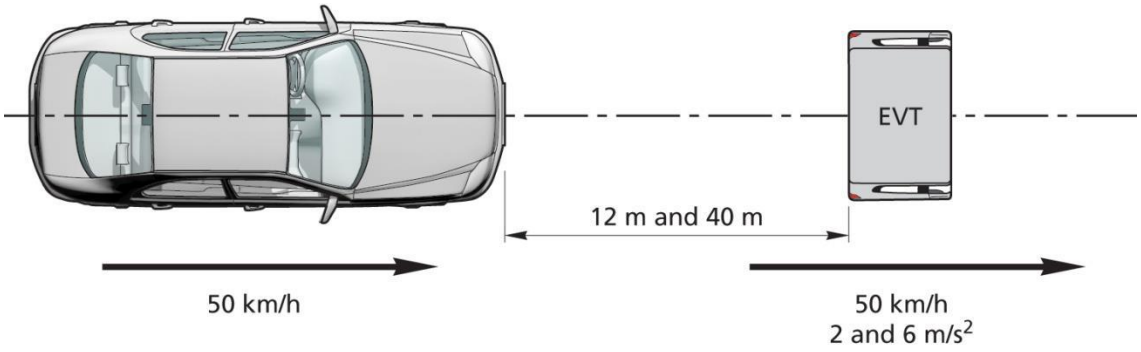


Figura 2.5. *Esquema de ensayo CCRb. (Bibliografía 9).*

Existe otra modalidad de ensayo llamada FCW (Forward Collision Warning) que se basa en la activación de una señal de aviso audiovisual para el conductor en el punto en que el sistema AEB detecta una posible colisión frontal. El conductor debe tener tiempo para frenar completa o parcialmente en el período comprendido entre el aviso y el choque. Para evaluar esta característica se repiten los ensayos anteriores de AEB Inter-Urban incorporando un robot de frenado que acciona el pedal de freno 1.2 segundos después de la activación del aviso audiovisual.

La siguiente tabla sirve a modo de resumen de todos los ensayos comentados hasta el momento.

	CCRs			
	AEB + FCW combined		AEB only	FCW only
	AEB	FCW		
AEB City	10-50 km/h	-	10-50 km/h	-
AEB Inter-Urban	-	30-80 km/h	30-80 km/h	30-80 km/h

	CCRm			
	AEB + FCW combined		AEB only	FCW only
	AEB	FCW		
AEB Inter-Urban	30-70 km/h	50-80 km/h	30-80 km/h	50-80 km/h

		CCRb	
		AEB+FCW combined, AEB only & FCW only	
		2 m/s²	6 m/s²
AEB Inter-Urban	**12m**	50 km/h	50 km/h
	40m	50 km/h	50 km/h

Tabla 1. *Resumen de ensayos "Car-to-Car Euro NCAP". (Bibliografía 9).*

El protocolo "Euro NCAP AEB VRU test protocol v101" define los ensayos VRU (Vulnerable Road Users). Se trata de test que recrean situaciones de cruce de peatones por delante del vehículo. Este debe poder detectarlos y ejecutar un freno de emergencia autónomo para evitar una colisión.

En este caso también se utiliza un "target" que simula las dimensiones y geometría de un peatón, tanto adulto como niño, EPTa y EPTc ("Euro NCAP Pedestrian Target adult" y "child") respectivamente. El objetivo recrea las propiedades típicas de un cuerpo que es reconocido como peatón por radares y cámaras.

Figura 2.6. *Objetivos o "target" EPTadult (izquierda) y EPTchild (derecha). (Bibliografía 10).*

En función de la incorporación del peatón en la trayectoria del vehículo a ensayar se clasifican los test AEB VRU con la siguiente nomenclatura.

- **CVFA (Car-to-VRU Farside Adult) scenario**

Escenario en que el peatón cruza corriendo la calzada por delante del vehículo desde el lado más lejano al conductor. Si no se aplicara ningún tipo de acción de frenado, el peatón impactaría en el punto medio de la geometría horizontal del coche.

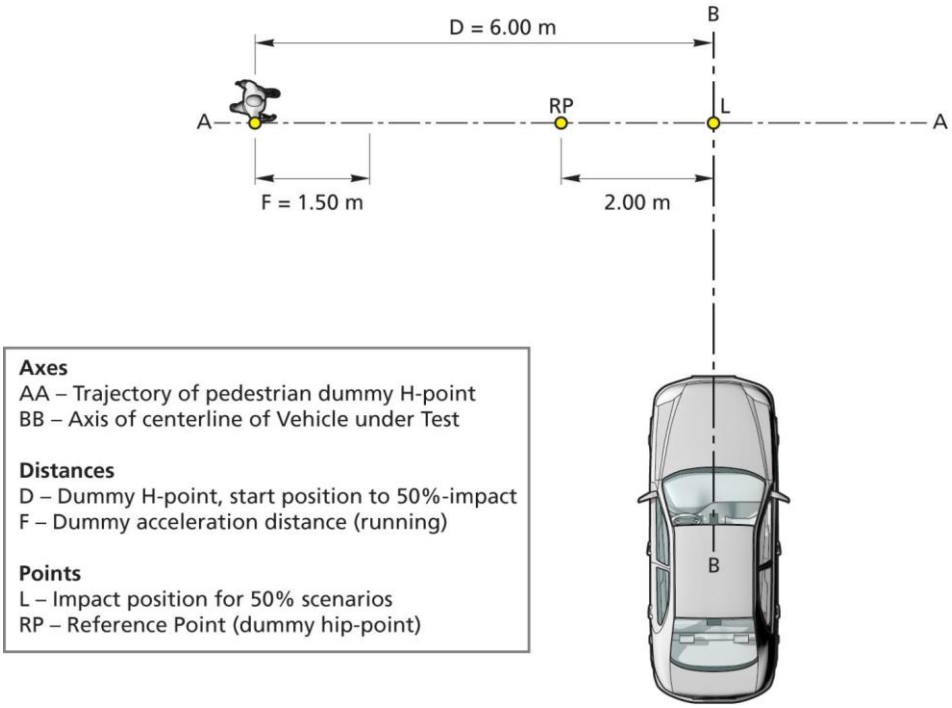


Figura 2.7. *Esquema de ensayo CVFA. (Bibliografía 10).*

- **CVNA-25 (Car-to-VRU Nearside Adult) scenario**

Escenario en que el peatón cruza la calzada por delante del vehículo desde el lado más cercano al conductor. Si no se aplicara ningún tipo de acción de frenado el peatón impactaría en un punto situado al 25% de la anchura horizontal del coche.

- **CVNA-75 (Car-to-VRU Nearside Adult) scenario**

Del mismo modo que en el ensayo anterior, el peatón impactaría a un 75% de la anchura horizontal del coche.

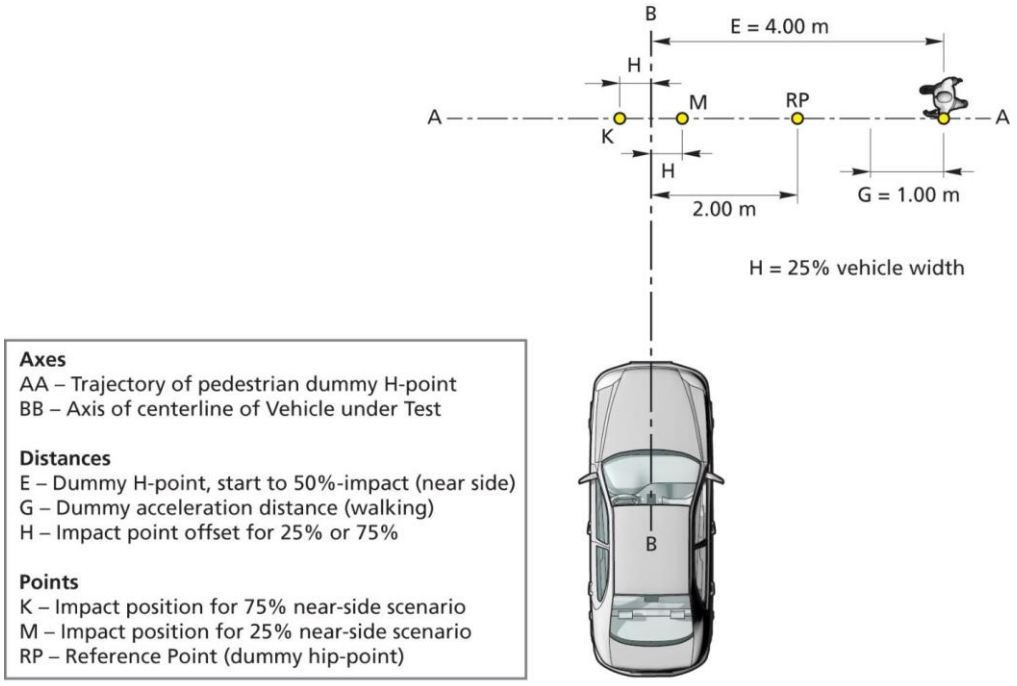


Figura 2.8. *Esquema de ensayo CVNA-75. (Bibliografía 10).*

- **CVNC (Car-to-VRU Nearside Child) scenario**

Escenario en el que el peatón de tamaño infantil cruza a 5 km/h la calzada por delante del vehículo desde el lado más cercano al conductor pero accediendo a la carretera por detrás de dos obstáculos (vehículos estacionados) que dificultan la visión de radares y cámaras. Si no se aplicara ningún tipo de acción de frenado, el peatón impactaría en el punto medio de la parte frontal del coche.

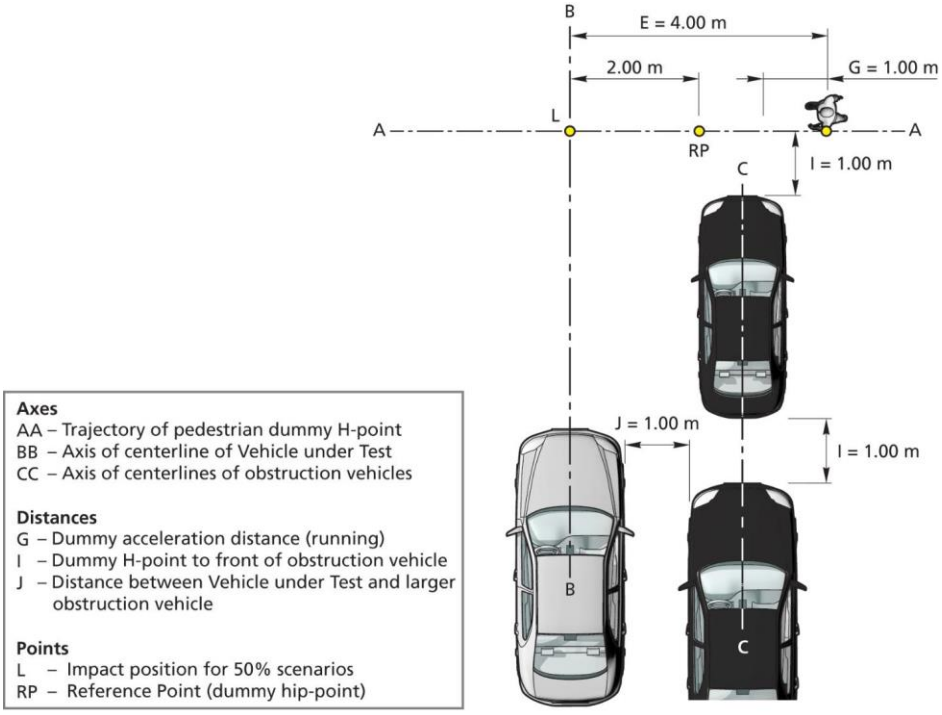


Figura 2.9. *Esquema de ensayo CVNC. (Bibliografía 10).*

Todos los test se realizan con un intervalo de velocidades de 20 a 60 km/h con incrementos de 5 km/h.

- **Test en el futuro**

Hasta el momento, estos son los ensayos tipificados por los protocolos Euro NCAP para evaluar los sistemas AEB y AEB VRU, pero en el futuro se barajan gran cantidad de nuevos escenarios que pretenden representar situaciones de la conducción cotidiana que hasta ahora no se habían tenido en cuenta. De aplicación a partir de 2018 se prevén, entre otros, ensayos que incluyen intersecciones de tráfico, incorporación a carriles, adelantamientos, conducción nocturna, detección de bicicletas, lectura de señales de tráfico, etc.

CAPÍTULO 3: ESPECIFICACIONES DE LA APLICACIÓN

3.1. Estructura de la aplicación

La aplicación a desarrollar se estructura en dos bloques, a los que se ha nombrado como "Bloque de Cálculo" y "Bloque de Interface". Cada uno de ellos posee funciones específicas pero es necesaria la comunicación entre sí, sobre todo la transmisión de datos de cálculo a interface, ya que esta presenta los resultados obtenidos de forma cómoda y permite gestionarlos sin necesidad de acceder a la programación.

3.1.1. Bloque de Cálculo

En este bloque se engloban todos los cálculos que permiten el traspaso de los datos desde los archivos de entrada con matrices sin editar a las matrices de resultados finales. Se trata de un archivo .m generado con Matlab, que contiene todo el código necesario.

Sus funciones son entonces, la carga de datos y la división de las matrices iniciales en texto y valores numéricos, el indexado, la ordenación y el almacenamiento de dichos datos como variables del entorno de trabajo, el cálculo de los puntos de Warning, TBR y ZBR, la detección de flancos de subida y bajada en señales concretas, la concatenación y división de fragmentos de texto, la creación de matrices de resultados a partir de los cálculos realizados, la deducción de conclusiones obtenidas con estos resultados y, en definitiva, todas las funciones que permiten obtener los resultados de interés de cada ensayo, siempre de forma iterativa, para poder introducir ensayos distintos y compararlos entre ellos.

Este bloque debe interactuar el mínimo posible con el usuario, es decir, no debe ser necesario acceder al código del programa para modificar parámetros, añadir archivos de entrada ni ver resultados. Para ello se crea el bloque de interface de comunicación con el usuario, donde se pueden manipular los datos y resultados de forma dinámica y sencilla.

3.1.2. Bloque de Interface

Este bloque debe ser la "cara" del programa y la herramienta de comunicación con el usuario. Por medio del componente Matlab GUIDE, se genera una ventana de visualización.

En esta ventana debe poderse visualizar uno a uno los canales de datos obtenidos en los ensayos con sus valores máximo y mínimo, así como sobreponer unos canales con otros para su comparación. Por medio de un selector se debe poder cambiar entre ensayos distintos previamente cargados. Los instantes donde se produce el Warinig y se inicia la acción del TBR se deben poder representar con color distinto dentro del gráfico en cualquier momento y para cualquier señal visualizada.

En tablas bien definidas se debe presentar la matriz de resultados denominada *MM* que presenta todos los parámetros que nos permiten sacar conclusiones y evaluar los ensayos. Con el mismo objetivo se presentan las matrices *MMeans* y *MDisp* que presentan medias y desviaciones típicas de algunas de las columnas de *MM*.

Un botón debe permitir la exportación de la matriz MM a un archivo externo de formato fácilmente leíble y edición sencilla.

Finalmente, la aplicación debe disponer de herramientas de navegación dentro de los gráficos, como son el "zoom" o la introducción de un cursor interactivo.

CAPÍTULO 4: ESTADO DEL ARTE

4.1. Productos y aplicaciones presentes en el mercado

Muchas marcas utilizan software específico con más o menos prestaciones. Sin embargo algunas aplicaciones son poco especializadas y pertenecen a la empresa que las ha desarrollado, de forma que no las comparten con otras entidades, por lo menos, no sin un coste de licencia de uso muy elevado y prestaciones limitadas, siempre sin mostrar ninguna información sobre los protocolos. Por este motivo no se pueden repasar estas herramientas ni sus prestaciones ya que son confidenciales y no representan una opción real para usuarios interesados que no pertenezcan a la empresa en cuestión.

No obstante, existen algunos programas a los que se puede acceder sin ser empleado de una empresa determinada. Hasta el momento permiten la visualización de datos pero exigen la intervención del usuario en el rastreo de valores y puntos de interés buscando con cursores sobre las gráficas.

4.1.1. CANOE

CANoe es un software desarrollado por *Vector Informatik GmbH* que se utiliza principalmente por fabricantes de equipos originales (OEM) de automóviles y proveedores de ECU ("Electronic Control Unit") para el desarrollo, análisis, simulación, test, diagnóstico y puesta apunto de los sistemas de comunicación internos en base CAN bus.

Tiene soporte sobre todas las versiones de Windows y permite una visualización completa de todos los canales de CAN que se introduzcan. Presenta opciones de exportación y análisis interesantes para el proyecto, pero el verdadero potencial del programa reside en el diagnóstico, simulación y desarrollo de estos sistemas de comunicación y no tanto en la evaluación de resultados obtenidos con un sistema ya creado. Resulta útil como paso intermedio para extraer los datos, pero no tanto para su análisis, que debe realizarse de forma manual.

(Extraído de bibliografía 5).

4.1.2. ADTF

La herramienta ADTF (Automotive Data and Time-Triggered Framework) es un software desarrollado por Elektrobit que se centra en el desarrollo, validación y visualización de sistemas de asistencia a la conducción. Este software accede a los sistemas de comunicación internos del vehículo y se centra en aquellos canales relevantes en los ensayos de asistencia a la conducción, por ejemplo los AEB.

Se trata de un software utilizado a nivel de campo para capturar datos, ofreciendo muchas y diversas opciones de calibración y sistemas de referencia de GPS. Con los datos capturados puede hacerse un seguimiento de las curvas y valores concretos y al mismo tiempo realizar cálculos predeterminados como el TTC ("time to collision").

Si bien es cierto que este software es muy útil durante los ensayos para registrar datos, no ofrece tanta potencia de análisis posterior, siendo este manual en su mayoría. Además, las licencias de este programa tienen un coste elevado y la programación es muy hermética y poco accesible, complicando enormemente la adaptación del programa a casos concretos.

Por este motivo se emplea ADTF para extraer las matrices de datos iniciales en las que se basa la aplicación a desarrollar, las cuales son datos sin edición notable y sin cálculos realizados sobre ellos, pero se descarta ADTF para su análisis.

(Extraído de bibliografía 6).

4.2. Evaluación de herramientas disponibles

4.2.1. Excel

Excel es un conocido software desarrollado por Microsoft y presente en la mayoría de equipos. Se trata de un programa de cálculo que ha tenido diversas versiones desde 1993 y que basa su funcionamiento en hojas de cálculo sobre las que colocar valores.

Ofrece un formato muy cómodo de almacenamiento de datos, ya que pueden dotarse de un formato práctico y pueden consultarse en la mayoría de ordenadores comerciales que incluyen este software de serie. Sin embargo, y en referencia a las necesidades y requerimientos del proyecto comentados en el apartado 2, presenta los siguientes inconvenientes que lo hacen ser descartado como soporte de programación y utilizado solo como formato de entrada/salida de datos.

- Los archivos resultantes ocupan mucho espacio de memoria.
- Presenta un nivel de seguridad bajo, ya que se puede acceder a sus archivos fácilmente.
- El trato de cantidades grandes de datos requiere muchos recursos de computación.
- La programación en base Visual Basic que ofrece es complicada y se necesita de conocimientos muy avanzados para sacar el mayor rendimiento.
- Se basa en la representación constante de los datos y por tanto, trabaja con índices de página de modo que dar nombre a variables y conjuntos de datos resulta complicado.
- Las funciones tipificadas que presenta no suponen una ventaja para la aplicación en cuestión.

(Extraído de bibliografía 8).

4.2.2. DIADEM

DIADEM es un software desarrollado por National Instruments enfocado en la visualización y la creación de reportes de datos obtenidos en ensayos y simulaciones. Por sus características podría ser un buen candidato. Para analizarlo se presentan sus ventajas, que pese a ser notables, no superan las que ofrece la opción elegida.

- Permite crear rutinas de búsqueda para localizar archivos en el disco duro o en la red.
- Ofrece compatibilidad con muchos formatos comerciales como .m o .xls.
- Permite visualizaciones muy precisas y diversas opciones de formato para gráficos y tablas de valores.
- Permite la creación de rutinas de programación en base Visual Basic.
- Ofrece una herramienta sencilla de redacción de reportes.

No obstante presenta los siguientes inconvenientes que deben tenerse en consideración.

(Extraído de bibliografía 7).

- Los cálculos se realizan por canales (magnitudes concretas adquiridas en un tiempo concreto), es decir, que la administración de variables simples o la creación de matrices de datos es compleja y dificulta el análisis.
- La carga de muchos ensayos sobre el programa implica una ingente cantidad de canales almacenados. Administrar todos estos datos resulta costoso y lento.
- La creación de reportes es difícil de adaptar a ensayos variables.
- La programación en Visual Basic no es práctica para algunos cálculos que deben realizarse.

4.2.3. Matlab

Matlab es un conocido software de programación y cálculo en lenguaje de alto nivel desarrollado por MathWorks. Ha resultado ser la mejor opción después de evaluar sus características dentro de las opciones que la empresa ha puesto en disposición de este proyecto. Su elección se basa en las ventajas que presenta y que son las siguientes.

- Compatibilidad con la mayoría de formatos comerciales como .dat, .xls, .csv, .txt, etc.
- Permite crear rutinas de búsqueda para localizar archivos en el disco duro o en la red.
- Ofrece opciones de visualización de datos muy parametrizables, reescribidles e interactivas.
- Permite la creación de interfaces de comunicación con el usuario por medio del complemento Matlab GUIDE, sin necesidad de intervenir en la programación.
- Pueden compilarse programas y aplicaciones para crear ejecutables que no necesiten del software Matlab instalado para su ejecución.
- Trabaja indistintamente con variables, matrices de conjuntos de datos, archivos externos, vectores, y otras entidades que pueden ordenarse de forma clara y sencilla.
- Puede trabajar dentro del entorno de programación sin necesidad de mostrar los datos en ningún momento.
- Tiene gran potencia de computación, soportando miles de datos en un mismo "workspace".
- Presenta una opción de creación de reportes automáticos llamada Matlab Report Generator, que puede resultar muy útil en ampliaciones de la aplicación objetivo.

Sí posee un inconveniente notable y es el elevado coste de su licencia.

(Extraído de bibliografía 1).

CAPÍTULO 5: CONFIGURACIÓN DEL SISTEMA

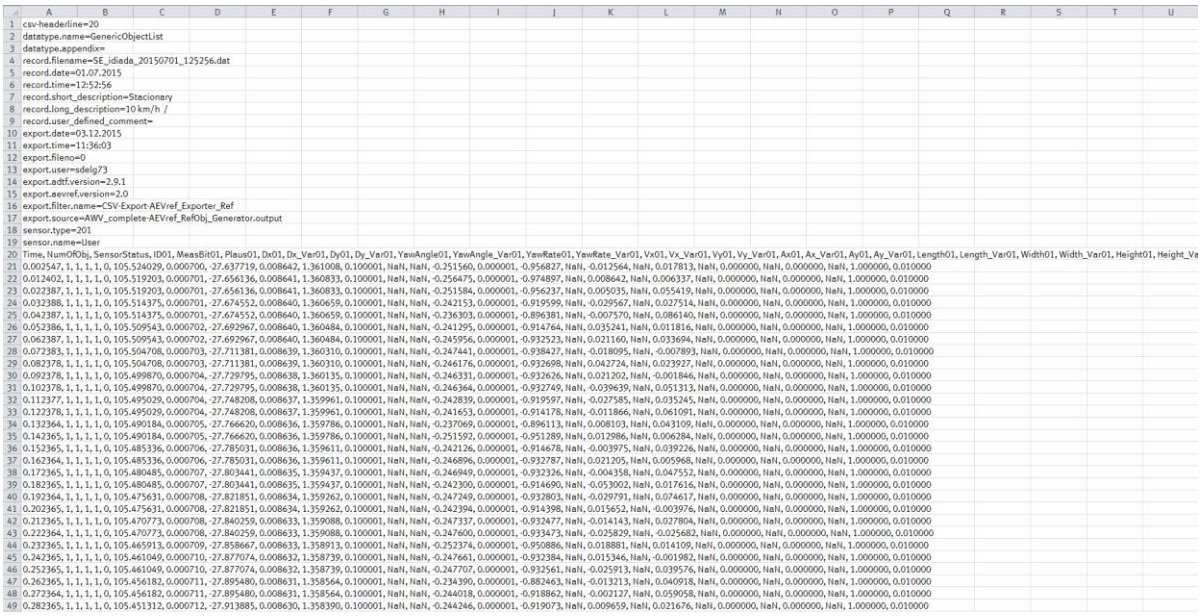
5.1. Adquisición y formato de datos de entrada

La adquisición de los datos para cada ensayo proviene de distintas fuentes. Dado que la propia empresa, en este caso, realiza los ensayos de sus modelos, tienen total acceso a los sistemas eléctricos de comunicación del vehículo, así como a valores externos medidos con instrumentos auxiliares.

Para cada ensayo AEB del fabricante se capturan los parámetros desde dos fuentes distintas. La primera es el GPS que se monta sobre el vehículo. Este dispositivo proporciona datos de posicionamiento del vehículo, absolutos o relativos al target, como velocidad, aceleración, rotación, etc.

La segunda fuente de datos es el sistema interno de comunicación del automóvil, el conocido como bus CAN (Controller Area Network) que une todos los sensores y centralitas presentes y permite la comunicación entre ellas. Accediendo a este bus se pueden extraer y capturar miles de variables internas, desde la velocidad hasta el ángulo de giro del volante, en cada instante de tiempo. Para estos ensayos se seleccionan las variables que más interesan y se extrae una matriz ordenada con todas ellas.

Desde estas fuentes se extrae toda la información necesaria para los análisis en forma de tres conjuntos de datos concretos que se analizan a continuación.



```
1  csv-headerline=20
2  datatype.name=GenericObjectList
3  datatype.appendix=
4  record.filename=SE_idiada_20150701_125256.dat
5  record.date=01.07.2015
6  record.time=12:52:56
7  record.short_description=Stacionary
8  record.long_description=10 km/h /
9  record.user_defined_comment=
10 export.date=03.12.2015
11 export.time=11:36:03
12 export.fileno=0
13 export.user=sdelg73
14 export.adtf.version=2.9.1
15 export.aevref.version=2.0
16 export.filter.name=CSV-Export-AEVref_Exporter_Ref
17 export.source=AWV_complete-AEVref_RefObj_Generator.output
18 sensor.type=201
19 sensor.name=User
20 Time, NumOfObj, SensorStatus, ID01, MeasBit01, Plaus01, Dx01, Dx_Var01, Dy01, Dy_Var01, YawAngle01, YawAngle_Var01, YawRate01, YawRate_Var01, Vx01, Vx_Var01, Vy01, Vy_Var01, Ax01, Ax_Var01, Ay01, Ay_Var01, Length01, Length_Var01, Width01, Width_Var01, Height01, Height_Va
21 0.002547, 1, 1, 1, 1, 0, 105.524029, 0.000700, -27.637719, 0.008642, 1.361008, 0.100001, NaN, NaN, -0.251560, 0.000001, -0.956827, NaN, -0.012564, NaN, 0.017813, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
22 0.012402, 1, 1, 1, 1, 0, 105.519203, 0.000701, -27.656136, 0.008641, 1.360833, 0.100001, NaN, NaN, -0.256475, 0.000001, -0.974897, NaN, 0.008642, NaN, 0.006337, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
23 0.022387, 1, 1, 1, 1, 0, 105.519203, 0.000701, -27.656136, 0.008641, 1.360833, 0.100001, NaN, NaN, -0.251584, 0.000001, -0.956237, NaN, 0.005035, NaN, 0.055419, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
24 0.032388, 1, 1, 1, 1, 0, 105.514375, 0.000701, -27.674552, 0.008640, 1.360659, 0.100001, NaN, NaN, -0.242153, 0.000001, -0.919599, NaN, -0.029567, NaN, 0.027514, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
25 0.042387, 1, 1, 1, 1, 0, 105.514375, 0.000701, -27.674552, 0.008640, 1.360659, 0.100001, NaN, NaN, -0.236303, 0.000001, -0.896381, NaN, -0.007570, NaN, 0.086140, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
26 0.052386, 1, 1, 1, 1, 0, 105.509543, 0.000702, -27.692967, 0.008640, 1.360484, 0.100001, NaN, NaN, -0.241295, 0.000001, -0.914764, NaN, 0.035241, NaN, 0.011816, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
27 0.062387, 1, 1, 1, 1, 0, 105.509543, 0.000702, -27.692967, 0.008640, 1.360484, 0.100001, NaN, NaN, -0.245956, 0.000001, -0.932523, NaN, 0.021160, NaN, 0.033694, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
28 0.072383, 1, 1, 1, 1, 0, 105.504708, 0.000703, -27.711381, 0.008639, 1.360310, 0.100001, NaN, NaN, -0.247441, 0.000001, -0.938427, NaN, -0.018095, NaN, -0.007893, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
29 0.082378, 1, 1, 1, 1, 0, 105.504708, 0.000703, -27.711381, 0.008639, 1.360310, 0.100001, NaN, NaN, -0.246176, 0.000001, -0.932698, NaN, 0.042724, NaN, 0.023927, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
30 0.092378, 1, 1, 1, 1, 0, 105.499870, 0.000704, -27.729795, 0.008638, 1.360135, 0.100001, NaN, NaN, -0.246331, 0.000001, -0.932626, NaN, 0.021202, NaN, -0.001846, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
31 0.102378, 1, 1, 1, 1, 0, 105.499870, 0.000704, -27.729795, 0.008638, 1.360135, 0.100001, NaN, NaN, -0.246364, 0.000001, -0.932749, NaN, -0.039639, NaN, 0.051313, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
32 0.112377, 1, 1, 1, 1, 0, 105.495029, 0.000704, -27.748208, 0.008637, 1.359961, 0.100001, NaN, NaN, -0.242839, 0.000001, -0.919597, NaN, -0.027585, NaN, 0.035245, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
33 0.122378, 1, 1, 1, 1, 0, 105.495029, 0.000704, -27.748208, 0.008637, 1.359961, 0.100001, NaN, NaN, -0.241653, 0.000001, -0.914178, NaN, -0.011866, NaN, 0.061091, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
34 0.132364, 1, 1, 1, 1, 0, 105.490184, 0.000705, -27.766620, 0.008636, 1.359786, 0.100001, NaN, NaN, -0.237069, 0.000001, -0.896113, NaN, 0.008103, NaN, 0.043109, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
35 0.142365, 1, 1, 1, 1, 0, 105.490184, 0.000705, -27.766620, 0.008636, 1.359786, 0.100001, NaN, NaN, -0.251592, 0.000001, -0.951289, NaN, 0.012986, NaN, 0.006284, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
36 0.152365, 1, 1, 1, 1, 0, 105.485336, 0.000706, -27.785031, 0.008636, 1.359611, 0.100001, NaN, NaN, -0.242126, 0.000001, -0.914678, NaN, -0.003975, NaN, 0.039226, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
37 0.162364, 1, 1, 1, 1, 0, 105.485336, 0.000706, -27.785031, 0.008636, 1.359611, 0.100001, NaN, NaN, -0.246896, 0.000001, -0.932787, NaN, 0.021205, NaN, 0.005968, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
38 0.172365, 1, 1, 1, 1, 0, 105.480485, 0.000707, -27.803441, 0.008635, 1.359437, 0.100001, NaN, NaN, -0.246949, 0.000001, -0.932326, NaN, -0.004358, NaN, 0.047552, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
39 0.182365, 1, 1, 1, 1, 0, 105.480485, 0.000707, -27.803441, 0.008635, 1.359437, 0.100001, NaN, NaN, -0.242300, 0.000001, -0.914690, NaN, -0.053002, NaN, 0.017616, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
40 0.192364, 1, 1, 1, 1, 0, 105.475631, 0.000708, -27.821851, 0.008634, 1.359262, 0.100001, NaN, NaN, -0.247249, 0.000001, -0.932803, NaN, -0.029791, NaN, 0.074617, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
41 0.202365, 1, 1, 1, 1, 0, 105.475631, 0.000708, -27.821851, 0.008634, 1.359262, 0.100001, NaN, NaN, -0.242394, 0.000001, -0.914398, NaN, 0.015652, NaN, -0.003976, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
42 0.212365, 1, 1, 1, 1, 0, 105.470773, 0.000708, -27.840259, 0.008633, 1.359088, 0.100001, NaN, NaN, -0.247337, 0.000001, -0.932477, NaN, -0.014143, NaN, 0.027804, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
43 0.222364, 1, 1, 1, 1, 0, 105.470773, 0.000708, -27.840259, 0.008633, 1.359088, 0.100001, NaN, NaN, -0.247600, 0.000001, -0.933473, NaN, -0.025829, NaN, -0.025682, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
44 0.232365, 1, 1, 1, 1, 0, 105.465913, 0.000709, -27.858667, 0.008633, 1.358913, 0.100001, NaN, NaN, -0.252374, 0.000001, -0.950886, NaN, 0.018881, NaN, 0.014109, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
45 0.242365, 1, 1, 1, 1, 0, 105.461049, 0.000710, -27.877074, 0.008632, 1.358739, 0.100001, NaN, NaN, -0.247661, 0.000001, -0.932384, NaN, 0.015346, NaN, -0.001982, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
46 0.252365, 1, 1, 1, 1, 0, 105.461049, 0.000710, -27.877074, 0.008632, 1.358739, 0.100001, NaN, NaN, -0.247707, 0.000001, -0.932561, NaN, -0.025913, NaN, 0.039576, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
47 0.262365, 1, 1, 1, 1, 0, 105.456182, 0.000711, -27.895480, 0.008631, 1.358564, 0.100001, NaN, NaN, -0.234390, 0.000001, -0.882463, NaN, -0.013213, NaN, 0.040918, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
48 0.272364, 1, 1, 1, 1, 0, 105.456182, 0.000711, -27.895480, 0.008631, 1.358564, 0.100001, NaN, NaN, -0.244018, 0.000001, -0.918862, NaN, -0.002127, NaN, 0.059058, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
49 0.282365, 1, 1, 1, 1, 0, 105.451312, 0.000712, -27.913885, 0.008630, 1.358390, 0.100001, NaN, NaN, -0.244246, 0.000001, -0.919073, NaN, 0.009659, NaN, 0.021676, NaN, 0.000000, NaN, 0.000000, NaN, 1.000000, 0.010000
```

Figura 5.1. *Ejeplo de datos de entrada en formato .csv.*

5.1.1. Matriz REF

La matriz denominada REF es una matriz de datos que se genera a partir de los datos de geoposicionamiento GPS pero con un punto de referencia (0) colocado sobre el target de choque, de forma que los valores de distancia, velocidad y aceleración se miden relativamente a este punto.

Contiene, al inicio, 19 espacios de una sola fila (filas de 1 a 19) reservados para anotaciones de texto que incluyen información sobre el número de ensayo, la velocidad establecida, la fecha, las rutas internas utilizadas para la obtención de los datos y otro contenido útil al que se debe poder acceder. Esta zona no se puede considerar parte de la matriz de datos, ya que no contiene valores numéricos y solo tiene una dimensión.

La fila 20 contiene un vector de nombres con los encabezados de cada una de las magnitudes que se presentan a continuación. A partir de este punto el indexado de la matriz pasa a ser vertical, siendo los nombres de la fila 20 los títulos de cada columna. El resto son valores numéricos formando una matriz de dimensiones variables según el tiempo y canales adquiridos. Muchos no son utilizados posteriormente o no contienen ninguna captura, pero es importante dejar el espacio por si en ensayos futuros se requiere capturar estos datos. Esto ocurre con todos los conjuntos de datos que se presentan a continuación.

Columna 1, *Time*: valores temporales de muestreo en los que se ha adquirido datos [s].

Columna 2, *NumOfObj*: número de objetos observados, en este caso solo 1, el vehículo.

Columna 3, *SensorStatus*: Indica 1 si el sensor está operativo.

Columna 4, *ID01*:

Columna 5, *MeasBit01*:

Columna 6, *Plaus01*:

} Variables internas de radar que no se capturan pero a las que se podría requerir el acceso en el futuro.

Columna 7, *Dx01*: posición en el eje X [m] del coche respecto al target, siendo este el punto 0.

Columna 8, *Dx_Var01*: variancia de la posición en X [m^2] respecto a su media.

Columna 9, *Dy01*: posición en el eje Y [m] del coche respecto al target, siendo este el punto 0.

Columna 10, *Dy_Var01*: variancia de la posición en Y [m^2] respecto a su media.

Columna 11, *YawAngle01*: ángulo Yaw [rad] según los ángulos náuticos (o Cardan) de la posición del vehículo.

Columna 12, *YawAngle_Var01*: variancia del ángulo Yaw [rad^2] respecto a su media.

Columna 11, *YawRate01*: velocidad angular Yaw [rad/s]según los ángulos náuticos (o Cardan) que sufre el vehículo.

Columna 12, *YawRate_Var01*: variancia de la velocidad angular Yaw [rad^2/s] respecto a su media.

Columna 13, *Vx01*: velocidad en el eje X [m/s] del coche respecto al eje cero marcado por el target.

Columna 14, *Vx_Var01*: variancia de la velocidad en el eje X [m^2/s] respecto a su media.

Columna 15, *Vy01*: velocidad en el eje Y [m/s] del coche respecto al eje cero marcado por el target.

Columna 16, *Vy_Var01*: variancia de la velocidad en el eje Y [m^2/s] respecto a su media.

Columna 17, *Ax01*: aceleración en el eje X [m/s^2] del coche respecto al eje cero marcado por el target.

Columna 18, *Ax_Var01*: variancia de la aceleración en el eje X [m^2/s^2] respecto a su media.

Columna 19, *Ay01*: aceleración en el eje Y [m/s^2] del coche respecto al eje cero marcado por el target.

Columna 20, *Ay_Var01*: variancia de la aceleración en el eje Y [m^2/s^2] respecto a su media.

Columna 21, *Length01*:

Columna 22, *Length_Var01*:

Columna 23, *Width01*:

Columna 24, *Width_Var01*:

Columna 25, *Height01*:

Columna 25, *Heigh_Vart01*:

Medidas dimensionales del target, en principio se mantienen constantes para todo el ensayo pero pueden tener relevancia en ensayos futuros

5.1.2. Matriz EGO

Con una configuración similar a la matriz REF, la matriz EGO muestra las coordenadas GPS del vehículo en valores absolutos, es decir, utilizando como punto de referencia las el norte geográfico. Estos valores sirven, previo cálculo, para crear la matriz de datos REF.

Las filas de 1 a 24 son información textual sobre el ensayo en una sola columna. La fila 25 contiene los títulos de las columnas siguientes. El resto son valores numéricos que forman una matriz de datos de dimensiones variables según ensayo.

Columna 1, *Time*: valores temporales de muestreo en los que se ha adquirido datos [s].

Columna 2, *UTC_Time*: Segunda base de valores temporales. No suele usarse.

Columna 3, *Pos_X*: Coordenada X de la posición del vehículo [m].

Columna 4, *Pos_Y*: Coordenada Y de la posición del vehículo [m].

Columna 5, *Pos_Z*: Coordenada Z de la posición del vehículo [m].

Columna 6, *Pos_Cov_XX*: covarianza de la posición en X respecto a la posición anterior.

Columna 7, *Pos_Cov_YY*: covarianza de la posición en Y respecto a la posición anterior.

Columna 8, *Pos_Cov_ZZ*: covarianza de la posición en Z respecto a la posición anterior.

Columna 9, *Vel_X: Componente X de la velocidad del vehículo [m/s].*

Columna 10, *Vel_Y*: *Componente Y de la velocidad del vehículo [m/s].*

Columna 11, *Vel_Z*: *Componente Z de la velocidad del vehículo [m/s].*

Columna 12, *Vel_Cov_XX*: covarianza de la velocidad en X respecto a la posición anterior.

Columna 11, *Vel_Cov_YY*: covarianza de la velocidad en Y respecto a la posición anterior.

Columna 12, *Vel_Cov_ZZ*: covarianza de la velocidad en Z respecto a la posición anterior.

Columna 13, *Acc_X*: *Componente X de la aceleración del vehículo [m/s^2].*

Columna 14, *Acc_Y*: *Componente Y de la aceleración del vehículo [m/s^2].*

Columna 15, *Acc_Z*: *Componente Z de la aceleración del vehículo [m/s^2].*

Columna 16, *Acc_Cov_XX*: covarianza de la aceleración en X respecto a la posición anterior.

Columna 17, *Acc_Cov_YY*: covarianza de la aceleración en Y respecto a la posición anterior.

Columna 18, *Acc_Cov_ZZ*: covarianza de la aceleración en Z respecto a la posición anterior.

Columna 19, *Rate_X*: Columna 20, *Rate_Y*: Columna 21, *Rate_Z*: Columna 22, *Rate_Cov_XX*: Columna 23, *Rate_Cov_YY*: Columna 24, *Rate_Cov_ZZ*:	Velocidades angulares de giro en los tres ejes. No se capturan para los ensayos actuales pero pueden requerirse en un futuro.
Columna 25, *Roll_X*: Columna 26, *Pitch_Y*: Columna 27, *Yaw_Z*: Columna 28, *Angle_Cov_XX*: Columna 29, *Angle_Cov_YY*: Columna 30, *Angle_Cov_ZZ*:	Ángulo Roll, Pitch y Yaw [rad] según los ángulos náuticos (o Cardan) de la posición del vehículo, y sus respectivas covarianzas respecto a los ángulos anteriores.

Columna 31, *GPSAuxDiffAge*: control de conexión del GPS con el satélite.

Columna 32, *SatsVis*: número del satélite utilizado por el GPS.

Columna 33, *GPSStatus*: marca de precisión del GPS. Puede variar entre dos posiciones (valores de precisión) distintas marcadas con un valor tipificado.

Columna 34, *INSStatus*: calidad de la señal.

5.1.3. Matriz FKT

La tercera matriz de datos se obtiene del bus CAN interno del vehículo. Para ello se han seleccionado las variables de interés y el resultado es un archivo de texto del mismo formato que las matrices anteriores. De nuevo, las primeras 17 filas son información textual sobre el ensayo en una sola columna. La fila 18 contiene los títulos de las columnas siguientes. El resto son valores numéricos que forman una matriz de datos de dimensiones variables según ensayo.

Columna 1, *Time*: valores temporales de muestreo en los que se ha adquirido datos [s].

Columna 2, *Vorwarnung_Sensor:* captura del aviso acústico del robot de freno.

Columna 3, *Vorwarnung_Extern*: aviso acústico de choque inminente para ser oído por el robot de freno.

Columna 4, *Hauptwarnung_Sensor*: estado del sensor principal de aviso.

Columna 5, *HBA_Stufe_Sensor*: indicación de los parámetros de presión que deben aplicarse al pedal para el pre-llenado de emergencia.

Columna 6, *Prefill_Sensor*: activación del pre-llenado de emergencia.

Columna 7, *ATB_Freigabe_Sensor*: activación del sistema de frenado de emergencia autónomo.

Columna 8, *ZBR_Freigabe_Sensor*: activación del sistema auxiliar de frenado de emergencia autónoma.

Columna 9, *Sollverzoegerung_Sensor:* perfil de frenada que el sistema requiere al coche para evitar la colisión.

Columna 10, *TTC_Ego_Target_Sensor*: "time to collision" interno del sistema.

Columna 11, *Obj_ID_Fusion*:
Columna 12, *Obj_ID_Radar*:
Columna 11, *Obj_ID_Video*:

} Identificadores de componentes externos, no son valores interesantes pero se respeta su espacio de memoria.

Columna 12, *ABS_Eingriff*: habilitación de los sistemas AEB.

Columna 13, *v_ESP_Fzg*: medida de velocidad del vehículo a partir de los sensores rotatorios de las ruedas.

Columna 14, *v_ref_ESP_Fzg*: perfil de frenada real que experimenta el vehículo y que intenta adaptarse al perfil solicitado.

Columna 15, *ESP_ATB_Aktiv*: estado del sensor principal de aviso (canal repetido).

Columna 16, *ESP_Hauptwarn_Aktiv*: Activación general de todos los sistemas de freno autónomo.

Columna 17, *Lenkradwinkel_Fzg*: ángulo de giro del vehículo [rad].

Columna 18, *Lenkradwinkel_Geschw_Fzg*: velocidad de giro del volante [rad/s].

Columna 19, *Fahrpedallrohwert_Fzg*: posición del pedal de aceleración.

Columna 20, *Fahrpedalgradient_Fzg*: gradiente de posición del pedal de aceleración respecto a la posición de reposo.

Columna 21, *Bremslichtschalter_Fzg*: indicador de luz de freno (encendida o apagada).

Columna 22, *Gierrate_Fzg*: Desvío del vehículo respecto a su trayectoria.

Columna 23, *Bremsdruck_Fzg*: Posición del pedal de freno.

Columna 24, *Kontaktsensor*: Columna 25, *BrakeApplRate*: Columna 26, *BrakeApplPos*: Columna 27, *BrakeApplForce*: Columna 28, *BrakeApplForceNominal*:	Variables de trabajo del robot de freno, como la presión a aplicar en el pedal. No se capturan de momento dado que el funcionamiento del robot se asegura por el proveedor, pero puede tener importancia en otras aplicaciones.

5.2. Formato de datos intermedios y de salida

Todas las matrices iniciales se presentan en formato .csv en tres archivos distintos. Como se puede apreciar en las figuras anteriores, existen partes diferenciadas de dimensiones y contenidos muy distintos. Por este motivo, para facilitar el análisis, se opta por la separación de campos en tres conjuntos de datos diferentes para cada archivo. Estos conjuntos de datos son los que quedan registrados en el entorno de trabajo de Matlab y son los mismos para cada archivo añadiendo la terminación REF, EGO o FKT. Las dimensiones de todos ellos son difíciles de prever exactamente dado que el tiempo de captura de datos puede ser diferente para cada ensayo.

El motivo principal de la separación entre archivos una vez introducidos en Matlab, es que cada uno de ellos posee una base de tiempo propia, y la unión de todos ellos en una sola matriz de datos resultaría muy costosa.

5.2.1. Matriz textdata

Esta matriz de una sola columna contiene las primeras n filas de cada archivo, que presentan, en forma de texto, la información pertinente de los ensayos. Este vector de texto se utiliza durante la ejecución del programa para dotar de nombre a los ensayos sobre los que se está trabajando. La última de las filas son los títulos de las columnas que se cargarán posteriormente. El resto de la matriz se llena de espacios vacíos para respetar las dimensiones.

Variables - textdataREF_1

textdataREF_1

20x28 cell

	1	2	3	4	5	6	7	8	9	10
1	'csv-headerline=20'	[]	[]	[]	[]	[]	[]	[]	[]	[]
2	'datatype.name=GenericObjectList'	[]	[]	[]	[]	[]	[]	[]	[]	[]
3	'datatype.appendix='	[]	[]	[]	[]	[]	[]	[]	[]	[]
4	'record.filename=SE_idiada_20150701_125256.dat'	[]	[]	[]	[]	[]	[]	[]	[]	[]
5	'record.date=01.07.2015'	[]	[]	[]	[]	[]	[]	[]	[]	[]
6	'record.time=12:52:56'	[]	[]	[]	[]	[]	[]	[]	[]	[]
7	'record.short_description=Stacionary'	[]	[]	[]	[]	[]	[]	[]	[]	[]
8	'record.long_description=10 km/h / '	[]	[]	[]	[]	[]	[]	[]	[]	[]
9	'record.user_defined_comment='	[]	[]	[]	[]	[]	[]	[]	[]	[]
10	'export.date=03.12.2015'	[]	[]	[]	[]	[]	[]	[]	[]	[]
11	'export.time=11:36:03'	[]	[]	[]	[]	[]	[]	[]	[]	[]
12	'export.fileno=0'	[]	[]	[]	[]	[]	[]	[]	[]	[]
13	'export.user=sdelg73'	[]	[]	[]	[]	[]	[]	[]	[]	[]
14	'export.adtf.version=2.9.1'	[]	[]	[]	[]	[]	[]	[]	[]	[]
15	'export.aevref.version=2.0'	[]	[]	[]	[]	[]	[]	[]	[]	[]
16	'export.filter.name=CSV-Export-AEVref_Exporter_Ref'	[]	[]	[]	[]	[]	[]	[]	[]	[]
17	'export.source=AWV_complete-AEVref_RefObj_Generator.output'	[]	[]	[]	[]	[]	[]	[]	[]	[]
18	'sensor.type=201'	[]	[]	[]	[]	[]	[]	[]	[]	[]
19	'sensor.name=User'	[]	[]	[]	[]	[]	[]	[]	[]	[]
20	'Time'	' NumOfObj'	' SensorStat...	' ID01'	' MeasBit01'	' Plaus01'	' Dx01'	' Dx_Var01'	' Dy01'	' Dy_Var01'
21										
22										
23										

Figura 5.2. *Matriz textdataREF de un ensayo de ejemplo.*

5.2.2. Matriz colheaders

Dentro de colheaders se guarda un vector de m columnas con todos los títulos de las magnitudes de cada archivo. Se respeta el orden en que se guardarán los datos en la siguiente matriz para que los índices de estas coincidan en todo momento y se puedan identificar fácilmente.

Variables - colheadersREF_1

colheadersREF_1

1x28 cell

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Time	NumOfObj	SensorStatus	ID01	MeasBit01	Plaus01	Dx01	Dx_Var01	Dy01	Dy_Var01	YawAngle01	YawAngle_...	YawRate01	YawRate_V...
2														
3														
4														

Figura 5.3. *Matriz colheadersREF de un ensayo de ejemplo.*

5.2.3. Matriz data

La matriz data es la que contiene todos los datos exclusivamente numéricos de cada uno de los archivos de entrada. El hecho de separar de ella todo el texto facilita en gran medida las operaciones posteriores. Se conserva el orden de las columnas de forma que el índice de cada una coincide con el nombre de la magnitud en la matriz colheaders.

Variables - dataREF_1

dataREF_1

4470x28 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.0025	1	1	1	1	0	105.5240	7.0000e-04	-27.6377	0.0086	1.3610	0.1000	0
2	0.0124	1	1	1	1	0	105.5192	7.0100e-04	-27.6561	0.0086	1.3608	0.1000	0
3	0.0224	1	1	1	1	0	105.5192	7.0100e-04	-27.6561	0.0086	1.3608	0.1000	0
4	0.0324	1	1	1	1	0	105.5144	7.0100e-04	-27.6746	0.0086	1.3607	0.1000	0
5	0.0424	1	1	1	1	0	105.5144	7.0100e-04	-27.6746	0.0086	1.3607	0.1000	0
6	0.0524	1	1	1	1	0	105.5095	7.0200e-04	-27.6930	0.0086	1.3605	0.1000	0
7	0.0624	1	1	1	1	0	105.5095	7.0200e-04	-27.6930	0.0086	1.3605	0.1000	0
8	0.0724	1	1	1	1	0	105.5047	7.0300e-04	-27.7114	0.0086	1.3603	0.1000	0
9	0.0824	1	1	1	1	0	105.5047	7.0300e-04	-27.7114	0.0086	1.3603	0.1000	0
10	0.0924	1	1	1	1	0	105.4999	7.0400e-04	-27.7298	0.0086	1.3601	0.1000	0
11	0.1024	1	1	1	1	0	105.4999	7.0400e-04	-27.7298	0.0086	1.3601	0.1000	0
12	0.1124	1	1	1	1	0	105.4950	7.0400e-04	-27.7482	0.0086	1.3600	0.1000	0
13	0.1224	1	1	1	1	0	105.4950	7.0400e-04	-27.7482	0.0086	1.3600	0.1000	0
14	0.1324	1	1	1	1	0	105.4902	7.0500e-04	-27.7666	0.0086	1.3598	0.1000	0
15	0.1424	1	1	1	1	0	105.4902	7.0500e-04	-27.7666	0.0086	1.3598	0.1000	0
16	0.1524	1	1	1	1	0	105.4853	7.0600e-04	-27.7850	0.0086	1.3596	0.1000	0
17	0.1624	1	1	1	1	0	105.4853	7.0600e-04	-27.7850	0.0086	1.3596	0.1000	0
18	0.1724	1	1	1	1	0	105.4805	7.0700e-04	-27.8034	0.0086	1.3594	0.1000	0
19	0.1824	1	1	1	1	0	105.4805	7.0700e-04	-27.8034	0.0086	1.3594	0.1000	0
20	0.1924	1	1	1	1	0	105.4756	7.0800e-04	-27.8219	0.0086	1.3593	0.1000	0
21	0.2024	1	1	1	1	0	105.4756	7.0800e-04	-27.8219	0.0086	1.3593	0.1000	0
22	0.2124	1	1	1	1	0	105.4708	7.0800e-04	-27.8403	0.0086	1.3591	0.1000	0
23	0.2224	1	1	1	1	0	105.4708	7.0800e-04	-27.8403	0.0086	1.3591	0.1000	0
24	0.2324	1	1	1	1	0	105.4659	7.0900e-04	-27.8587	0.0086	1.3589	0.1000	0
25	0.2424	1	1	1	1	0	105.4610	7.1000e-04	-27.8771	0.0086	1.3587	0.1000	0
26	0.2524	1	1	1	1	0	105.4610	7.1000e-04	-27.8771	0.0086	1.3587	0.1000	0

Figura 5.4. *Matriz dataREF de un ensayo de ejemplo.*

Con todos los datos introducidos en el programa se puede proceder a la operación y generación de resultados. Los resultados se presentan en nuevos conjuntos de datos de la forma siguiente.

5.2.4. Vector M

En el capítulo 6 se detalla los cálculos y procedimientos que se emplean para obtener, a partir de las matrices intermedias, datos numéricos de interés para la evaluación de los ensayos. Para cada ensayo se realizan estos cálculos y se obtienen toda una serie de variables que se distribuyen de forma ordenada dentro de un vector denominado M. Entonces, se genera un vector M por cada ensayo introducido en el programa.

Variables - M

M

1x24 cell

	1	2	3	4	5	6	7	8	9	10	11	12
1	'SE_idiada_20150701_140935_40'	'40'	-9.9179	0	1.2203	12.4276	9.6598	-36.6636	35.4600	0	0	0
2												
3												
4												
5												

Figura 5.5. *Vector M de un ensayo de ejemplo.*

En orden, estas columnas son las siguientes, combinando texto con valores numéricos.

Columna 1, *Test name*: Nombre del ensayo del que se han extraído los valores.

Columna 2, *Test speed:* Velocidad teórica del ensayo.

Columna 3, *Impact Speed [km/h]*: Velocidad a la que se produce el impacto.

Columna 4, *Dist.Long.(x) en STOP [m]*: Distancia a la que queda el vehículo del target una vez se ha detenido.

Columna 5, *Warinig TTC [s]*: "Time To Collision" en el momento en que se activa el aviso de impacto inminente.

Columna 6, *Warning Distance [m]*: Distancia del vehículo respecto al target en el instante en que se activa el aviso de impacto inminente.

Columna 7, *Warning Duration [s]*: Duración del aviso de impacto inminente.

Columna 8, *Warning V_Signal [km/h]*: Velocidad que marcan las ruedas del vehículo en el momento del aviso.

Columna 9, *Warning speed [km/h]:* Velocidad que marca el GPS en el momento del aviso.

Columna 10, *ATB TTC [s]*: "Time To Collision" en el momento en que se activa el frenado de emergencia automático.

Columna 11, *ATB Distance [m]*: Distancia del vehículo respecto al target en el instante en que se activa el frenado de emergencia automático.

Columna 12, *ATB Duration [s]*: Duración de la señal de frenada de emergencia.

Columna 13, *ATB V_Signal [km/h]*: Velocidad que marcan las ruedas del vehículo en el momento del frenado de emergencia.

Columna 14, *ATB speed [km/h]:* Velocidad que marca el GPS en el momento del frenado de emergencia.

Columna 15, *ZBR TTC [s]*: "Time To Collision" en el momento en que se activa el frenado de emergencia suplementario automático.

Columna 16, *ZBR Distance [m]*: Distancia del vehículo respecto al target en el instante en que se activa el frenado de emergencia suplementario automático.

Columna 17, *ZBR Duration [s]*: Duración de la señal de frenada de emergencia.

Columna 18, *ZBR V_Signal [km/h]*: Velocidad que marcan las ruedas del vehículo en el momento del frenado de emergencia.

Columna 19, *ZBR speed [km/h]:* Velocidad que marca el GPS en el momento del frenado de emergencia.

Columna 20, *Braking Profile ANB MAX [g]:* Valor máximo del perfil de frenada que el sistema exige al vehículo para que se detenga evitando el impacto. Expresado en g.

Columna 21, *Braking Profile ANB MAX [m/s^2]:* Valor máximo del perfil de frenada que el sistema exige al vehículo para que se detenga evitando el impacto. Expresado en m/s^2.

Columna 22, *Braking Profile ESP MAX [g]:* Valor máximo del perfil de frenada que realmente experimenta el vehículo antes del impacto. Expresado en g.

Columna 23, *Delay [ms]:* Tiempo entre los máximos de los dos perfiles de frenada.

Columna 24, *Comments:* Comentarios que se tomaron durante el ensayo in situ por los ingenieros responsables.

5.2.5. Vector Mh

Una vez creado el vector M se necesita una forma de identificar cada uno de los términos de este. Para ello se define el vector Mheaders, abreviado como Mh, en el que se escriben en el mismo orden los nombres de las variables de M para ser indexadas.

Variables - Mh

Mh

1x25 cell

	1	2	3	4	5	6	7	8	9	10
1	Test name	Test speed [km/h]	Impact speed [km/h]	Dist,Long.(x) en STOP	Warning TTC[s]	Warning Distance[m]	Warning Duration [s]	Warning V_Signal[km/h]	Warning Sp...	TBR TTC [s]
2										
3										
4										

Figura 5.6. Vector Mh de un ensayo de ejemplo.

5.2.6. Matriz MM

Dado que uno de los objetivos finales del programa es la comparación entre ensayos, se contempla la introducción de más de un ensayo distinto. De cada uno de ellos debe generarse un vector de valores interesantes M, pero para evitar la superposición de este vector o la creación de infinitos vectores distintos, la matriz MM recoge de forma ordenada las consecuentes matrices M de cada ensayo, en el orden en que se han introducido y con el título adecuado en cada fila. Esta será la matriz final que se exportará a formatos externos a Matlab (.xlsx) unida al vector Mh como la primera de sus filas.

Variables - MM

MM

5x24 cell

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	'SE_idiada_2...	'10'	-2.2010	0	0	0	0	0	0	0	0	0	0	0
2	'SE_idiada_2...	'20'	0	0.3132	0.6714	3.4533	2.9398	-18.5161	18.5200	0.7310	3.7636	1.3002	-18.5359	18.5700
3	'SE_idiada_2...	'25'	-3.8629	0	0.6044	4.1236	3.0804	-24.5612	24.6000	0.6641	4.5343	0.6208	-24.5787	24.6000
4	'SE_idiada_2...	'35'	-6.7444	0	0.5413	4.7830	5.5171	-31.8130	31.6800	0.5700	5.0514	0.7201	-31.9031	31.8000
5	'SE_idiada_2...	'40'	-9.9179	0	1.2203	12.4276	9.6598	-36.6636	35.4600	0	0	0	0	0
6														
7														

Figura 5.7. *Matriz MM de un ensayo de ejemplo.*

5.2.7. Vector MMean

Se trata de un vector de dimensiones fijas donde se presentan los valores medios de algunos de los parámetros resultantes en la matriz MM.

Variables - MMean

MMean

1x8 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-4.5452	0.0626	0.6075	4.9575	0.3930	2.6699	0.0207	0.0997						
2														

Figura 5.8. *Vector MMeans de un ensayo de ejemplo.*

5.2.8. Vector MDisp

Este vector también de dimensiones fijas, muestra los valores de desviación estándar que dan una muestra de la variabilidad entre ensayos. Utiliza los mismos parámetros en la matriz MM que el vector MMean, ya que son los más relevantes.

Variables - MDisp

MDisp

1x8 double

	1	2	3	4	5	6	7	8	9	10
1	3.8834	0.1401	0.4339	4.5653	0.3633	2.4799	0.0464	0.2229		
2										

Figura 5.9. *Vector MDisp de un ensayo de ejemplo.*

5.2.9. Vector MMeanh

Vector de 8 columnas que, del mismo modo que el vector Mh, incluye los títulos de las los valores contenidos en MMean, en el mismo orden de columnas para poder ser referenciadas en las visualizaciones posteriores. También sierve para indexar los valores de MDisp ya que se calculan sobre las mismas columnas de MM.

Variables - MMeanh

MMeanh

1x8 cell

	1	2	3	4	5	6	7	8	9	10
1	Impact speed [km/h]	Dist,Long.(x) en STOP	Warning TTC[s]	Warning Distance[m]	TBR TTC [s]	TBR Distance[m]	ZBR TTC [s]	ZBR Distance[m]		
2										
3										
4										

Figura 5.10. *Vector MMeans de un ensayo de ejemplo.*

Todas estas matrices de datos pueden verse relacionadas en forma de esquema en la figura 6.4.

CAPÍTULO 6: ARQUITECTURA DEL SISTEMA

6.1. Arquitectura general del sistema

Para entender el funcionamiento del sistema es importante recalcar el lugar que ocupa dentro del proceso de análisis de ensayos AEB y el desarrollo de estas tecnologías. El esquema siguiente resume de forma sencilla el proceso normal de evaluación de un ensayo AEB típico como los que se explican en el apartado 3.3.

En un primer paso se realiza el ensayo y por medio de programas de captura "on board" se recogen los datos internos del bus de comunicaciones del vehículo y de los sistemas GPS que se utilizan de manera complementaria. Estos datos pasan a ser analizados en un segundo bloque, que constituye el objeto de este proyecto, donde se extraen conclusiones, se realizan cálculos y comparaciones y se procede a la visualización de los datos para análisis detallados, como por ejemplo, intentar dar explicaciones a comportamientos anómalos registrados.

Finalmente, todos los resultados obtenidos deben almacenarse junto con las conclusiones extraídas de los datos, por ejemplo el correcto o incorrecto funcionamiento del radar o los sistemas internos del vehículo, la elección de un software de detección de choques u otro, la necesidad de corrección de ciertos errores o parámetros, y un largo etcétera.

Gracias a la aplicación desarrollada en este proyecto no solo se automatiza el segundo paso sino que se agiliza en gran medida el tercero, generando reportes en tablas Excel de forma automática, siendo un formato cómodo de almacenamiento, y detectándose de forma automática algunas de las correctas o incorrectas actuaciones del sistema.

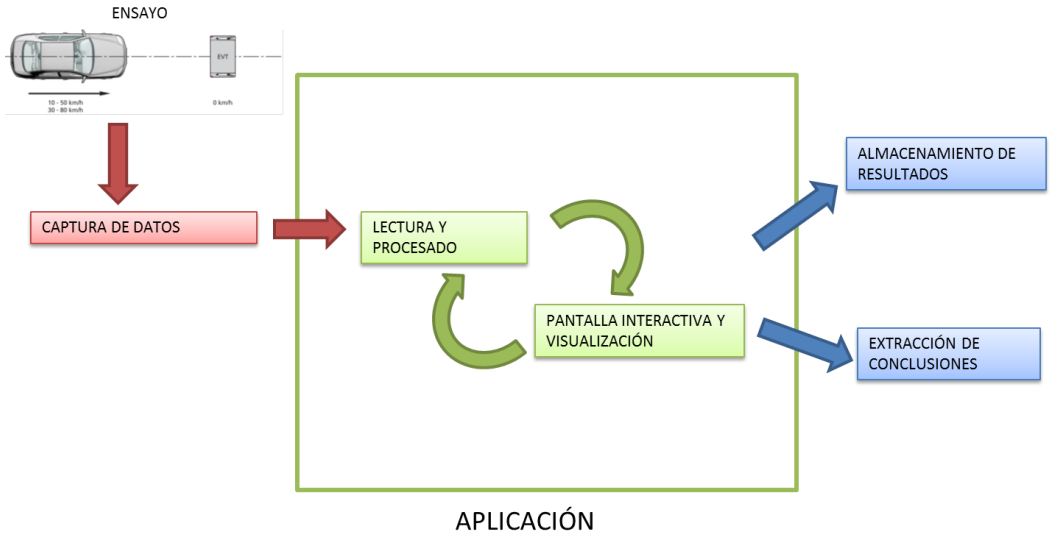


Figura 6.1. *Esquema de los pasos en un ensayo AEB.*

Una vez situado el programa dentro de contexto, se procede a analizar su funcionamiento concreto. Como ya se ha comentado anteriormente, el sistema consta de dos partes diferenciadas pero unidas. Cada parte tiene unos objetivos distintos que se detallan a continuación. El siguiente esquema resume las prestaciones de cada una de sus partes.

Figura 6.2. *Esquema de prestaciones de cada uno de los bloques.*

6.2. Arquitectura de cálculo

El bloque de cálculo consta de un archivo Matlab (.m) de texto que contiene diferentes secciones de programación. El objetivo final de este programa es introducir, analizar y extraer datos de salida.

El siguiente diagrama de flujo explica de forma muy general los pasos que sigue el programa de principio a fin.

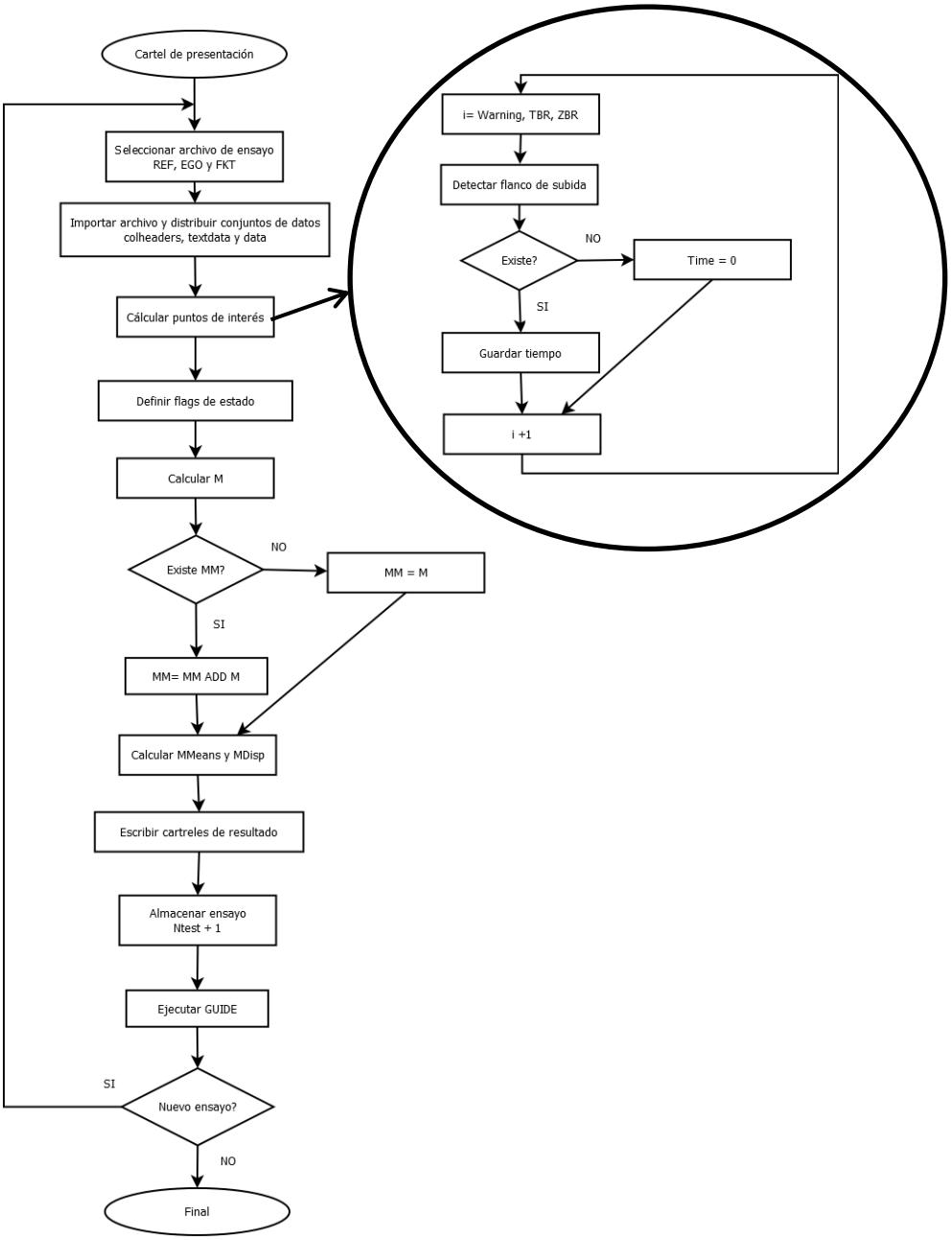


Figura 6.3. *Diagrama de flujo aproximado del bloque de cálculo.*

Consta de 8 secciones consecutivas que se separan a modo de capítulos por título y una pequeña descripción de su cometido. Es importante destacar que todos los pasos y líneas de texto están contenidas en este archivo y no se han tipificado funciones de programación. El motivo es que durante la ejecución del programa no se producen repeticiones que requieran llamar a una función más de una vez por ensayo. Los pocos casos en que esto ocurre utilizan una cantidad de variables de entrada y salida con nombres creados a partir del número de ensayo que, la tipificación de estos fragmentos, lejos de aportar claridad a la programación, derivaría en una complicada matriz de variables intermedias que dificultaría mucho la comprensión del programa.

Durante toda la ejecución del programa se procura eliminar las variables intermedias que dejan de tener utilidad para que, una vez compilado todo el programa, no se ocupen espacios de memoria innecesarios.

El siguiente esquema muestra la secuencia que siguen las matrices de datos relevantes del sistema, aquellas que se consideran de entrada, intermedias y de salida, así como las variables que la ventana interface utiliza para trabajar.

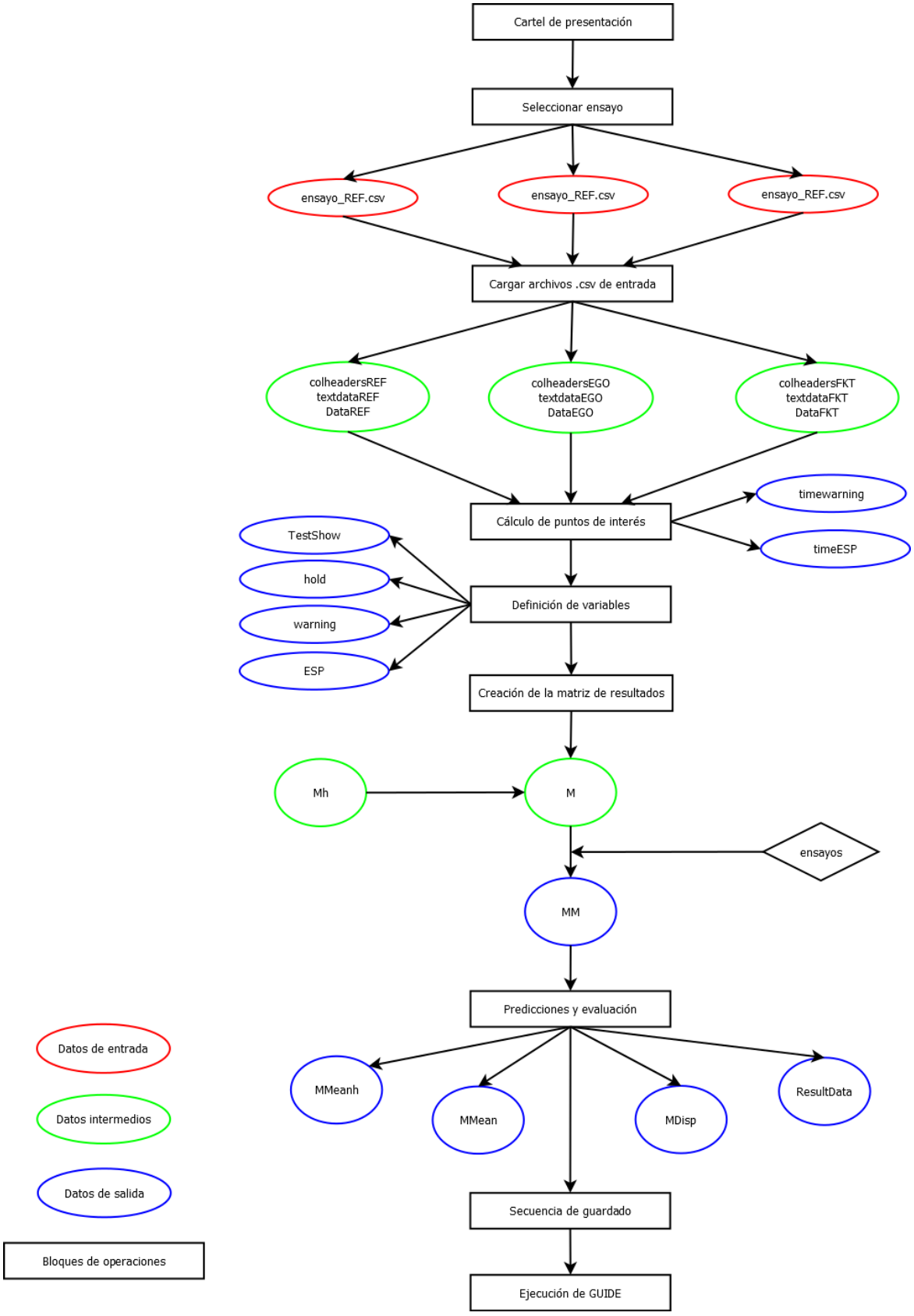


Figura 6.4. *Esquema de variables y matrices de datos dentro del bloque de cálculo del programa.*

Con estas consideraciones iniciales se procede a analizar el programa por capítulos, tal y como se separan en el archivo .m original (anexo A).

6.2.1. Cartel de presentación

Anexo A, líneas de 1 a 9.

El primer paso es mostrar un cartel de presentación que contiene el nombre de la aplicación y su creador. Después se presenta un cartel que informa al usuario de que debe elegir el archivo REF del ensayo que desea cargar.

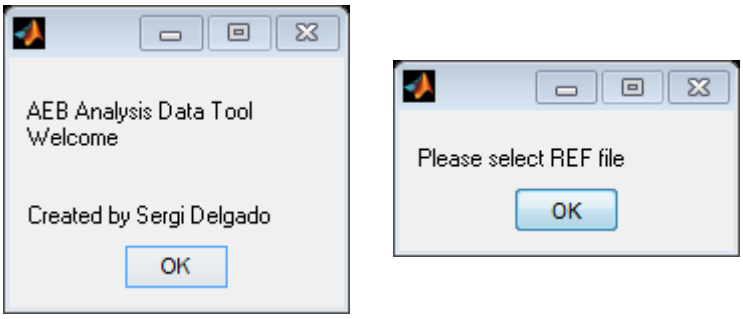


Figura 6.5. *Carteles de presentación y de inicio de ensayo.*

6.2.2. Cargar archivos .csv de entrada

Anexo A, líneas de 10 a 87.

A partir de este momento, se procede a la introducción de los archivos .csv de entrada. El primero de todos es el archivo REF, aquel que contiene los datos del GPS con coordenadas diferenciales respecto al target. A partir del nombre de este archivo el programa es capaz de crear los nombres de los archivos EGO (que contiene los datos del GPS en valor absoluto) y FKT (que contienen los datos que provienen del bus CAN del vehículo) para cargar así el ensayo completo en el entorno de trabajo.

En total quedan registradas 9 matrices de datos en el sistema para cada ensayo, las conocidas como *colheaders, textdata* y *data*, con su terminación correspondiente.

6.2.3. Cálculo de puntos de interés

Anexo A, líneas de 88 a 150.

En este capítulo se calculan los puntos de activación de las señales de warning, TBR y zBR, de vital importancia para el análisis posterior y la representación de las señales.

En primer lugar, y dado que los primeros segundos de cada ensayo suelen registrar valores de puesta a punto que generan errores de análisis y falsos positivos, se eliminan los primeros 5 segundos de ensayo.

Estas tres señales tienen solo dos valores típicos, 0 si no se registra actividad o 1 si aparece señal. Por tanto, para determinar el momento en que se activan las señales debe encontrarse el instante en el que se produce un flanco de subida en el canal, es decir, el primer instante en que se registra el valor máximo (1) que puede adquirir. El programa detecta este flanco de subida, evalúa en que instante se produce y lo guarda en la variable *timewarning*, *timeTBR o timeZBR* según convenga ya que el procedimiento es el mismo en los tres casos

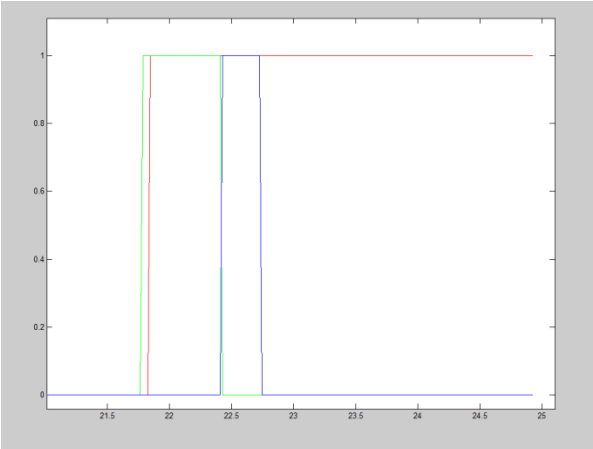


Figura 6.6. *Canales de warning (rojo), TBR (verde) y ZBR (azul) mostrando su forma típica de pulso, con flanco de subida y bajada.*

6.2.4. Definición de variables booleanas

Anexo A, líneas de 151 a 165.

En este capítulo se definen una serie de variables booleanas que son necesarias en la interface de usuario. Estas son *hold, warning y ESP*. Sirven para indicar en todo momento si estas opciones de la visualización están o no activas.

También se define la variable impacto, que indica para el ensayo concreto si el programa ha deducido que se produjo un impacto en el ensayo o se consiguió frenar el vehículo a tiempo, y *TestShow*, que indicará en todo momento que ensayo se está visualizando en la interface.

6.2.5. Creación de la matriz de resultados

El siguiente capítulo es el más extenso dada la gran cantidad de cálculos que se realizan. El objetivo es crear la matriz de resultados llamada MM, compuesta a partir del vector de resultados de cada ensayo llamado M.

En primer lugar se crea la estructura base del vector M. Esta posee valores tanto de texto como numéricos. A partir de este punto se empiezan a calcular los datos que deben ocupar cada columna de M. Para hacerlo de forma ordenada se dividen en tres grupos y se identifican por el nombre de la magnitud que registran y la columna de M que ocupan.

<u>DATOS GENERALES</u>

Anexo A, líneas de 166 a 254.

Las columnas 1 (nombre del ensayo) y 2 (velocidad ideal del ensayo) se extraen de los comentarios en formato texto que se incluyen en la matriz *textdata*.

Para obtener la columna 3 (velocidad de impacto) y 4 (distancia longitudinal en STOP), es importante detectar primero si se ha producido o no un impacto durante el ensayo. La forma más fiable de detectarlo es evaluando la posición del vehículo en relación al tiempo que nos aporta el GPS diferencial. Dado que el objetivo se considera el eje de coordenadas, toda posición anterior al objetivo es de valor positivo, mientras que toda posición posterior es negativa. Así pues, si el vehículo atraviesa el punto 0 del eje X de coordenadas, significa que ha entrado en contacto con el objetivo y por tanto se ha producido un choque.

Se evalúa entonces si el canal de posición contenido en *dataREF* sufre un cambio de signo en su trayectoria. Si la respuesta es **afirmativa, se ha producido un choque** y por tanto la distancia al target en frenada es automáticamente 0 y la velocidad de impacto se obtiene a partir de evaluar la velocidad en el punto en que se produce el cambio de signo. Si la respuesta es **negativa, no se ha producido un choque** y por tanto la velocidad de impacto es automáticamente 0 y la distancia al target en frenada se obtiene a partir de evaluar el punto en que la velocidad se estabiliza en 0 (<0.1 m/s).

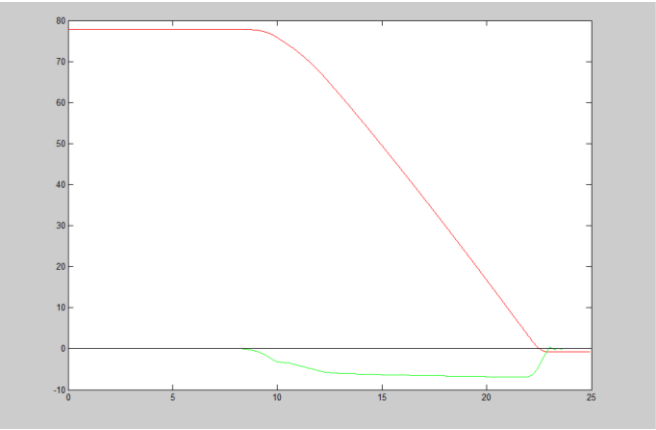


Figura 6.7. Ejemplo de posición [m] (rojo) y velocidad [m/s] (verde) de un ensayo real en el que se ha producido impacto.

MEDIDAS DE WARNING, TBR Y ZBR

Anexo A, líneas de 255 a 458.

Estas posiciones las ocupan medidas de los canales warning, TBR y ZBR. Dado que estos canales son similares, para cada uno de ellos debe utilizarse el mismo procedimiento. Por eso se agrupan las posiciones que realizan el mismo cálculo sobre los tres canales y se calculan en bucle los valores de cada canal.

Las columnas 5, 10 y 15 contienen el TTC (time to colision), que marca el tiempo que calcula el radar que falta para que el vehículo impacte, calculando en cada instante en función de su velocidad y trayectoria. Esta medida también es muestreada en el tiempo, pero no se captura ningún canal de TTC durante los ensayos y por eso, debe crearse a partir de los datos a los que se tiene acceso.

Para calcular los valores de este canal, el mejor procedimiento es aplicar la fórmula del Movimiento Rectilíneo Uniforme (MRU) sobre cada pareja de puntos de *Dx01* y *Vx01*.

$$X = X_0 + V_0 * t \qquad [1]$$

$$t = X - X_0/V_0 \qquad [2]$$

Entonces considerando X como 0 (eje de coordenadas) se obtiene para cada instante de muestreo el TTC. No se añade el valor de aceleración a este cálculo ya que este valor debe ser nulo en los instantes interesantes para el análisis y además dificulta el cálculo y añade gran variedad de anomalías al canal resultante sin modificar apreciablemente los resultados aprovechables.

Con el canal de TTC's en función del tiempo creado satisfactoriamente, se puede calcular el TTC de los instantes concretos de activación de warning, TBR y ZBR y escribirlos en la matriz M.

Las columnas 6, 11 y 16 miden la distancia a la que se encuentra el vehículo respecto al target en el momento de activación de una de las tres señales. Del mismo modo las columnas 8, 13 y 18, presentan la velocidad en el momento de activación según el GPS y 9, 14 y 19, la velocidad en el momento de activación según el vehículo.

Son distintas las columnas 7, 12 y 17, que presentan la duración de cada una de las señales warning, TBR y ZBR a partir de detectar el flanco de bajada de esas señales y restar los tiempos entre flanco de bajada y flanco de subida calculado anteriormente.

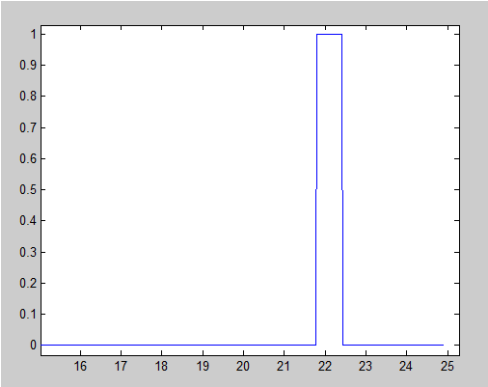


Figura 6.8. *Señal de warning con flanco de subida y flanco de bajada.*

MEDIDAS EN LOS PERFILES DE FRENADA

Anexo A, líneas de 459 a 511.

Los perfiles de frenada son aquellas variaciones de desaceleración que el sistema AEB calcula que se necesitan para detener el coche evitando así la colisión inminente, así como la desaceleración real que experimenta el coche intentando ajustarse a este perfil. Ambas tienen una forma similar y marcan la brusquedad y efectividad de la maniobra.

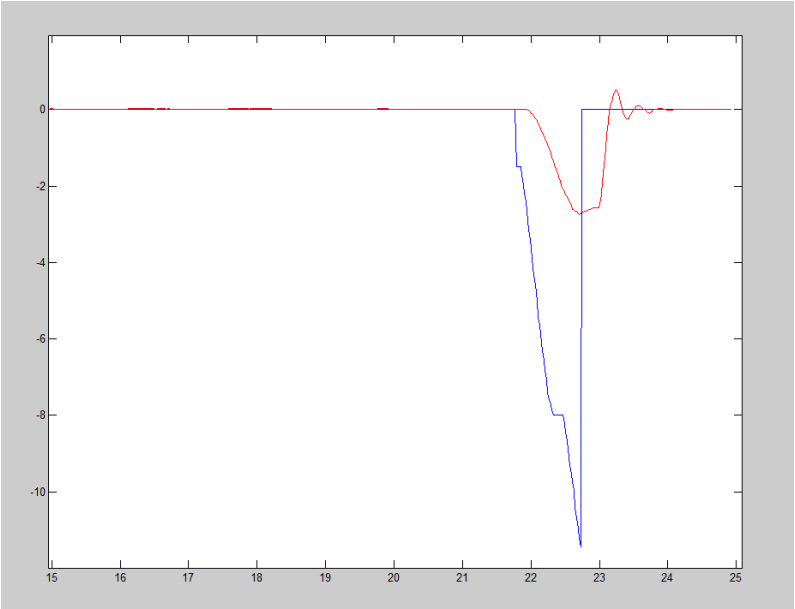


Figura 6.9. *Perfiles de frenada requerido [g] (rojo) y real [g] (azul) para un ensayo AEB.*

Se calcula, sobre estos perfiles, el valor máximo del perfil de frenada en g, siendo $1\,g = 9.81\ m/s^2$, el valor máximo del perfil de frenada en m/s^2 y el valor máximo del perfil de frenada real, los cuales se añaden a las columnas 20, 21 y 22. También en la columna 23 se calcula el retraso que se produce entre ambos máximos. Se resta el valor real del ideal ya que normalmente este se produce antes. En cualquier caso, un retraso negativo indica que ha sucedido de forma contraria.

La columna 24 (comentarios) se destina a contener los comentarios de texto que se tomaron durante el ensayo. Estos comentarios vienen escritos en una de las filas de la matriz *textdataREF*.

Anexo A, líneas de 512 a 530.

Finalmente se procede a la creación de la matriz MM que ha de guardar los vectores M de cada ensayo que se cargue, y también se crea el vector Mh, que contiene los títulos de cada una de las columnas de MM.

6.2.6. Predicciones y evaluación

Anexo A, líneas de 531 a 603.

Dentro de predicciones y evaluación se encuentran dos cálculos que adquieren sentido una vez se ha añadido más de un ensayo en el programa. Se eligen una serie de columnas en MM sobre las que se quieren realizar dos operaciones, calcular la media de

los valores entre ensayos (matriz *MMean*) y calcular la desviación típica de esos valores respecto a su media *(MDisp*). Los nombres de los canales elegidos se almacenan en *MMeanh* y hasta el momento son los siguientes.

- *Impact Speed [km/h]*
- *Dist.Long.(x) en STOP [m]*
- *Warinig TTC [s]*
- *Warning Distance [m]*
- *ATB TTC [s]*
- *ATB Distance [m]*
- *ZBR TTC [s]*
- *ZBR Distance [m]*

MMean y MDisp resultan de gran utilidad en la comparación de ensayos ya que muestran tendencias que se producen entre ellos, y también resaltan la aparición de valores atípicos que pueden llevar al usuario a detectar errores en los datos. Comparando las medias y desviaciones obtenidas con las que cabría esperar o los rangos que se esperan se obtiene información útil, pero estos valores de comparación son de carácter confidencial.

A continuación se genera el vector *Result* donde se guardan frases de texto con una apreciación de lo ocurrido durante el ensayo.

6.2.7. Secuencia de guardado

Anexo A, líneas de 604 a 657.

Después de extraer todos los datos necesarios del ensayo analizado se debe tener en cuenta que el programa ofrece la opción de añadir más de un ensayo dentro de la misma ejecución y visualización. Es por eso que se crea una matriz llamada *TName* donde se guardan los nombres de todos los ensayos cargados en el programa, en forma de vector columna. Además se añade al final de cada nombre la velocidad de ensayo, tal y como se apunta en M(1,2).

Para evitar que al compilar de nuevo el programa desde el principio para un nuevo ensayo se borren los datos del ensayo anterior y no puedan visualizarse, se almacenan todas las matrices intermedias con una terminación en función del número de ensayo que marca *Ntest.* Así, el espacio de trabajo contiene siempre los datos de todos los ensayos una vez se han realizado los cálculos pertinentes con ellos. También se incluyen las variables *timewarning* y *timeESP*, ya que el instante de activación de ambas señales es distinto en cada ensayo y debe representarse en la interface de visualización.

6.2.8. Ejecución de la interface GUIDE

Anexo A, líneas de 658 a 682.

Finalmente es el momento de ejecutar la segunda parte de la aplicación la interface de usuario en base GUIDE. Dado que ambos archivos se encuentran en la misma carpeta de trabajo solo es necesario llamar a la aplicación directamente por su nombre.

Con la interface activada, se crea un mensaje de texto con una pregunta para el usuario. Si se elige la opción "YES", el programa de cálculo se reinicia para añadir un nuevo ensayo, cuyos datos se acumularán a los existentes de ensayos anteriores, creando así un bucle que permite la inserción de múltiples ensayos en el programa. Cada vez que se reinicie el cálculo, la variable *Ntest* se incrementa en 1.

Si se elige la opción "NO", el programa da por terminada su función y deja paso solo a la interface. En caso de no elegir ninguna opción, la interface de usuario puede utilizarse normalmente y el programa queda en pausa hasta que se dé una respuesta.

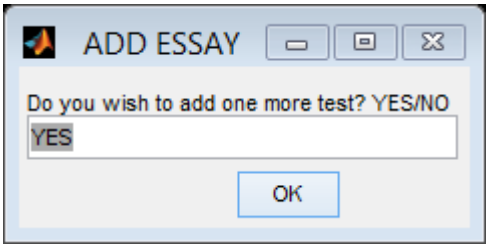


Figura 6.10. *Cartel de adición de ensayo.*

6.3. Arquitectura de Interface Usuario

La interface de usuario es la segunda parte del programa y consta de una pantalla desarrollada con la herramienta Matlab GUIDE, así como su correspondiente archivo de código. La programación de esta ventana se basa en un lenguaje de programación orientada a objetos, es decir que todos los elementos que se presentan en ella son una entidad propia con nombre y propiedades concretas y que reaccionan de forma distinta si se realizan acciones sobre ellas, ya sea interactuando con otras entidades o por acción del usuario.

En el caso de este proyecto, se presenta primero un índice de esta pantalla con todos los elementos que contiene y su nombre identificador.

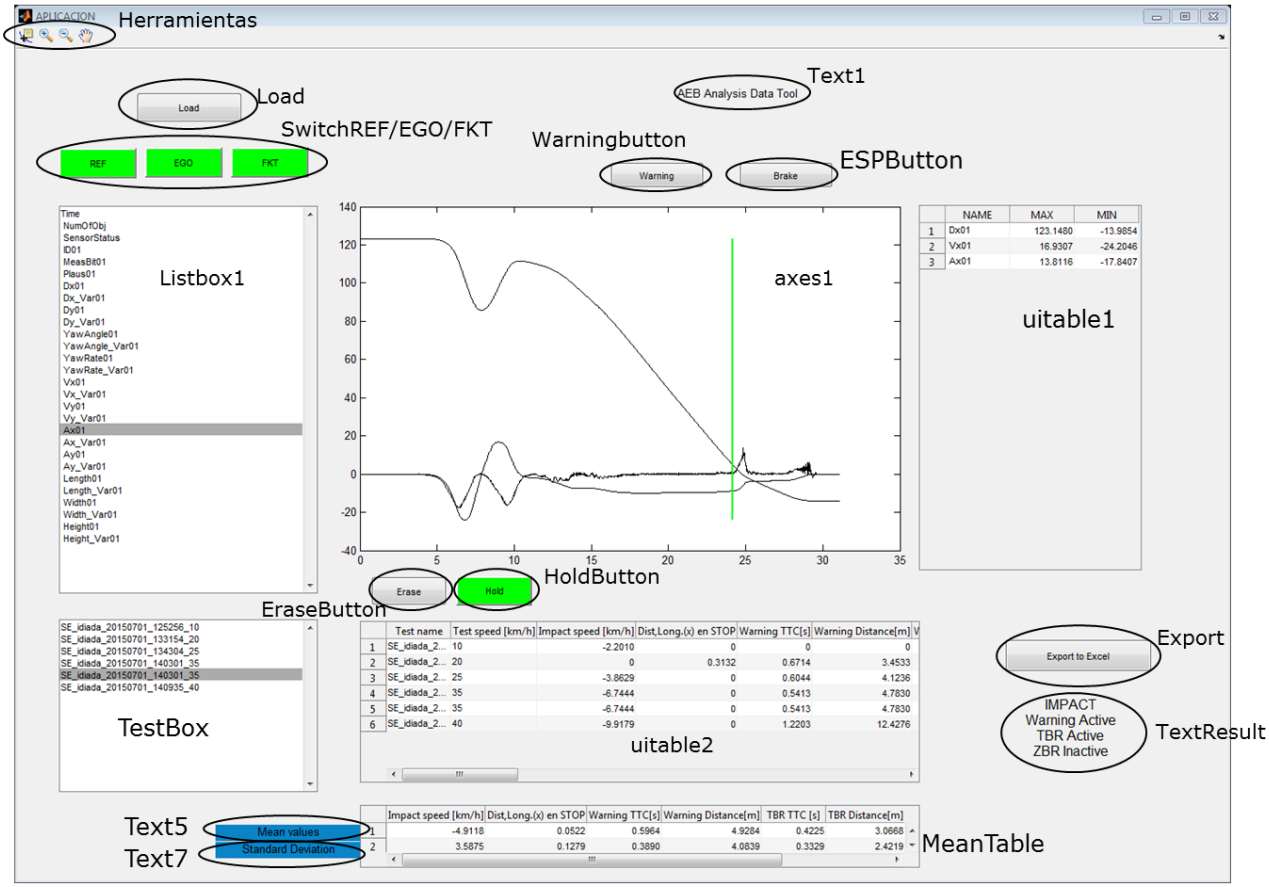


Figura 6.11. *Visión de la ventana de interface usuario.*

- Herramientas: Barra de herramientas que proporciona el desarrollador GUIDE y que se aplican sobre el gráfico dinámico. Son cursor para leer valores puntuales, acercar/alejar gráfico y una mano para desplazar el gráfico libremente.
- Load: Botón de carga de datos.
- SwitchREF, SwitchEGO y SwitchFKT: Botones para cambiar entre magnitudes de diferente origen.
- WarningButton: Botón para añadir el instante de inicio de warning al gráfico.
- ESPButton: Botón para añadir el instante de inicio de señal de frenado en el gráfico.
- Axes1: Espacio de grafico interactivo.
- Text1: Nombre del programa.
- Listbox1: Lista de canales a representar.
- TestBox: Lista de ensayos cargados en el programa.
- Text5: Indica donde se encuentran los valores medios.
- Text7: Indica donde se encuentran los valores de desviación estándar.
- Uitable1: Tabla con los valores máximo y mínimo de las señales representadas en el gráfico.
- Export: Botón que inicia la exportación de datos a formato Excel.
- TextResult: Espacio para las conclusiones obtenidas con el cálculo.
- Uitable2: Espacio para mostrar la matriz MM con todos los ensayos y los nombres de los datos que contiene.
- EraseButton: Botón para dejar el gráfico en blanco.
- HoldButton: Botón para mantener las señales representadas en el grafico y añadir nuevas.
- Meantable: Espacio para mostrar las medias y desviaciones típicas entre ensayos.

La programación de las interacciones de estos objetos y el programa de cálculo se estructura entonces en distintas funciones, la activación de las cuales depende de las acciones que el usuario realice sobre la ventana de interface. Con esta idea se explican una por una las funciones que componen la programación de la interface, considerando sus condiciones de activación y el resultado final de su aplicación.

Anexo B, líneas de 1 a 88.

Las primeras cuatro funciones se generan de forma automática por el editor GUIDE y por tanto no se analizan en este documento. Realizan tareas de creación y puesta a punto del programa.

Función Load

Anexo B, líneas de 232 a 330.

Para iniciar la función el usuario debe pulsar sobre el botón *Load*. Al iniciarse, el programa carga la variable *TestShow* desde el espacio de trabajo y crea todos los nombres con los que cargar los archivos correspondientes, desde en entorno de trabajo, para tener en la ventana interface todos los datos necesarios.

Función SwitchREF, Función SwitchEGO y Función SwitchFKT

Anexo B, líneas de 192 a 228.

Pulsando el botón *SwitchREF, SwitchEGO o SwitchFKT*, el usuario activa su función correspondiente, la cual cambia las variables que se presentan en *Listbox1* y que están disponibles para ser graficadas.

Función listbox1

Anexo B, líneas de 90 a 189.

Esta función se activa en el momento en que se pulsa sobre cualquiera de los nombres de las variables que se presentan en *Listbox1*. Al pulsar uno de estos nombres la función grafica el canal correspondiente sobre el gráfico *axes1*. Substituyendo las curvas existentes si *hold* no está activo, o añadiendo a las existentes en caso contrario.

Función EraseButton

Anexo B, líneas de 333 a 341.

Al pulsar sobre el botón *EraseButton* se ejecuta una rutina que permite dejar en blanco la representación que se está visualizando en *axes1*.

Función Warningbutton

Anexo B, líneas de 344 a 392.

La rutina que se inicializa al pulsar el botón *Warningbutton* se encarga de cargar desde el espacio de trabajo el punto calculado como inicio de la acción del warning en forma de flanco de subida. Este instante es muy importante para el análisis de las señales y por tanto resulta de gran interés poder representarlo sobrepuesto a cualquier señal que se esté visualizando en el grafico dinámico. La forma visual y clara que se ha elegido es una línea recta vertical de color rojo y grosor doble que atraviesa todas las señales que se estén visualizando en el instante temporal *timewarning_ (el número del ensayo seleccionado, que se indexa como siempre con la variable Testshow).*

Esta opción de visualización se puede activar o desactivar cada vez que se pulse sobre *Warningbutton,* por eso se evalúa la variable booleana previamente creada *warning* que indica con true o false si debe representarse o no la recta.

Función ESPButton

Anexo B, líneas de 416 a 464.

Con el mismo procedimiento que *Warningbutton, esta función representa el otro valor más importante,* el punto calculado como inicio de la acción de señal de frenado TBR. En esta ocasión el color de la línea recta vertical es verde.

Función Hold

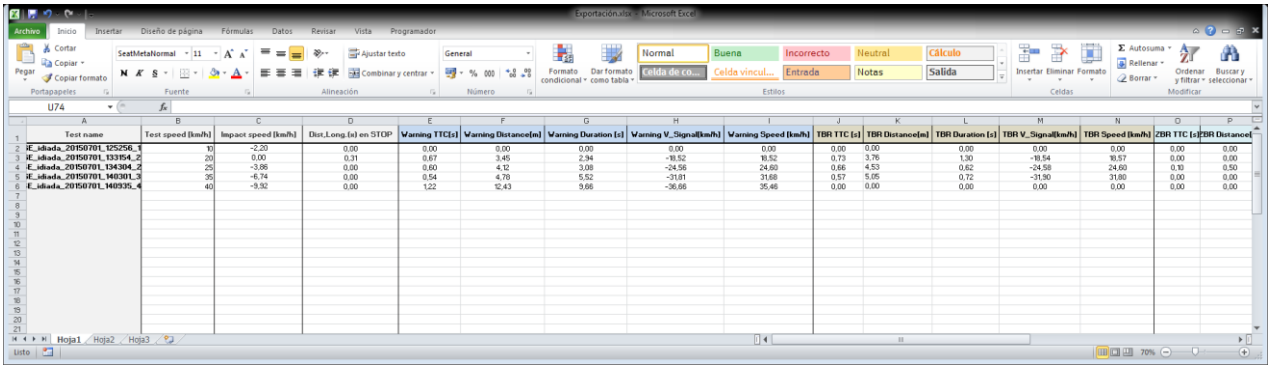
Anexo B, líneas de 395 a 413.

La función hold, de forma similar a la función *Warningbutton_Callback,* cambia el valor booleano de la variable *hold* y lo reescribe en el entorno de trabajo. Asi se puede decidir en la función *listbox1_Callback* si se debe representar añadiendo o substituyendo señales sobre el gráfico. También cambia el color del botón *HoldButton* a verde en estado activo y gris en estado inactivo.

Función Export

Anexo B, líneas de 468 a 481.

Utilizando el botón *Export* se ejecuta la rutina de exportación que permite copiar los datos contenidos en la matriz MM a un archivo Excel previamente editado para ser almacenados o visualizados por terceros con un formato agradable y cómodo. Al seleccionar la matriz MM con los datos y Mh con los nombres de estos.



Test name	Test speed [km/h]	Impact speed [km/h]	Dist_Long [s] en STOP	Warning TTC[s]	Warning Distance[m]	Warning Duration [s]	Warning V_Signal[km/h]	Warning Speed [km/h]	TBR TTC [s]	TBR Distance[m]	TBR Duration [s]	TBR V_Signal[km/h]	TBR Speed [km/h]	ZBR TTC [s]	ZBR Distance[
E_idiada_20150701_125256_1	10	-2,20	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
E_idiada_20150701_133154_2	20	0,00	0,31	0,67	3,45	2,94	-19,52	19,52	0,73	3,76	1,30	-19,54	19,57	0,00	0,00
E_idiada_20150701_134304_2	25	-3,86	0,00	0,60	4,12	3,08	-24,56	24,60	0,66	4,53	0,62	-24,58	24,60	0,10	0,50
E_idiada_20150701_140301_3	35	-6,74	0,00	0,54	4,70	5,52	-31,81	31,68	0,57	5,05	0,72	-31,90	31,80	0,00	0,00
E_idiada_20150701_140935_4	40	-9,92	0,00	1,22	12,43	9,66	-36,66	35,46	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Figura 6.12. *Ejemplo de archivo final exportado a formato .xlsx.*

Función TestBox

Anexo B, líneas de 485 a 508.

La última función es aquella que se activa con la selección del nombre de uno de los ensayos que se muestra en la caja de texto *TestBox*. Al hacerlo, el programa asigna el índice de ese ensayo a la variable TestShow que, como se ha visto anteriormente, determina de que ensayo lee el programa los datos antes de mostrarlos en la ventana interface. También deshabilita todos los botones *Switch* hasta que el usuario pulse el botón *Load*, cargando así los datos del ensayo pertinente.

6.4. Interconexiones y comunicación

Ambas arquitecturas deben comunicarse para el funcionamiento total del sistema. Si bien es cierto que en la mayoría de casos se ejecuta primero la arquitectura de cálculo y es después la arquitectura de interface usuario la que lee datos almacenados por la primera, en algunos casos las variables se leen y reescriben de forma continua mientras el usuario interactúa con la ventana interface. Para simplificar el análisis el siguiente esquema pretende resumir que variables se transfieren solo unidireccionalmente de cálculo a interface y que otras se transfieren bidireccionalmente entre ambas arquitectura.

Estas variables o matrices de datos se han nombrado y definido en los capítulos anteriores.

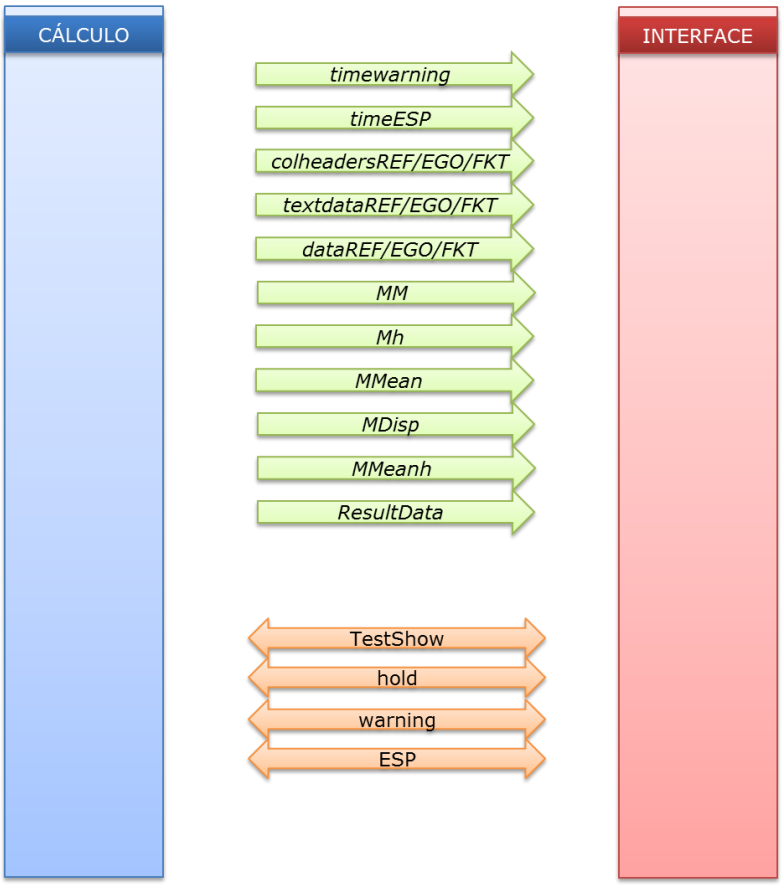


Figura 6.13. *Esquema de variables y matrices de datos cuando se comunican entre los dos bloques del programa.*

CAPÍTULO 7: PLANIFICACIÓN

Desde que Henry Gantt creara en 1915 una herramienta de control y planificación de actividades que recibió el nombre de "Diagrama de Gantt, se ha establecido como un modelo de referencia para muchas planificaciones modernas. Este diagrama permite generar una visión sencilla y precisa de las fechas y tiempos marcados para cada hito del proyecto. Por su utilidad y sencillez se utiliza el Diagrama de Gantt para la planificación de este proyecto.

7.1. Planificación temporal del desarrollo

Después de acotar los requerimientos y características del proyecto, como se especifican en el capítulo 2, se realizó un análisis de recursos y definición de objetivos y tiempo. Para cumplir dichos objetivos se confeccionó un Diagrama de Gantt con hitos semanales, coincidiendo con el funcionamiento logístico del cliente.

Por medio de esta planificación se dividió el total de tareas en fragmentos y se calculó el tiempo que idealmente debían requerir. Se estableció por tanto un orden de prioridades que ha servido de guía a la hora de evaluar el progreso del proyecto y encontrar variaciones que se hayan podido producir.

Cabe destacar que la actividad "Redacción de la memoria" ocupa gran parte de la planificación simultáneamente a otras tareas, dado que se trata de un trabajo de redacción constante durante todo el proyecto, y no una tarea que se pueda realizar solo en algún punto concreto. También se establece un periodo de margen antes de la entrega del proyecto en previsión de retrasos que puedan producirse respecto a la planificación original.

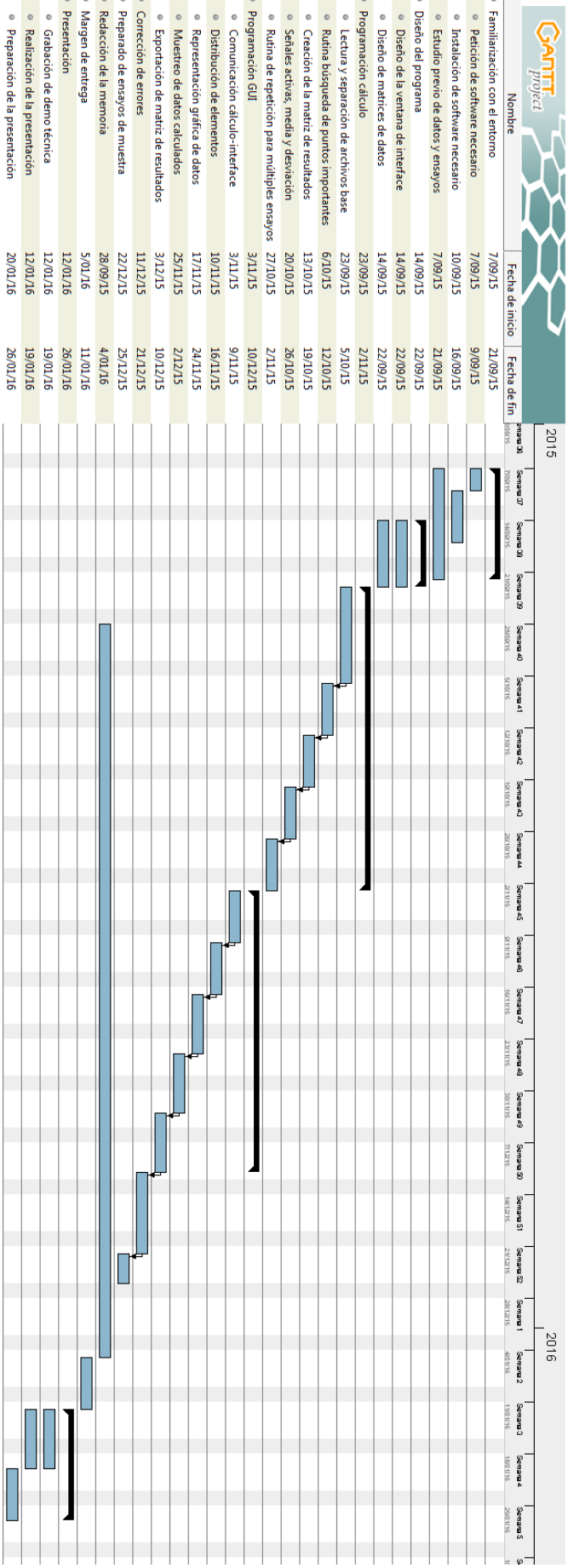


Figura 7.1. *Diagrama de Gantt.*

7.2. Diagrama temporal final

Después de la finalización del proyecto se observa de nuevo el Diagrama de Gannt original y se realiza un nuevo diagrama con los tiempos que realmente se han cumplido. A modo de comparación y evaluación se sobreponen estos dos diagramas y se observan dos puntos importantes. En primer lugar se ha producido un retraso en la instalación de software de una semana aproximadamente debido a que la adquisición de licencias MATLAB ha ocupado más tiempo del previsto. Esta circunstancia provoca que todas las actividades de programación se retrasen en el tiempo.

De igual modo, la detección y corrección de errores ha requerido más tiempo del previsto debido a la gran cantidad de líneas de programa y funciones que se implementan en la interface de usuario, así como algunas correcciones de carácter estético.

Los cambios han obligado a utilizar parte del tiempo de margen que se planificó al inicio del ensayo y gracias a eso se ha conseguido terminar el proyecto dentro del tiempo establecido a pesar de los imprevistos sufridos.

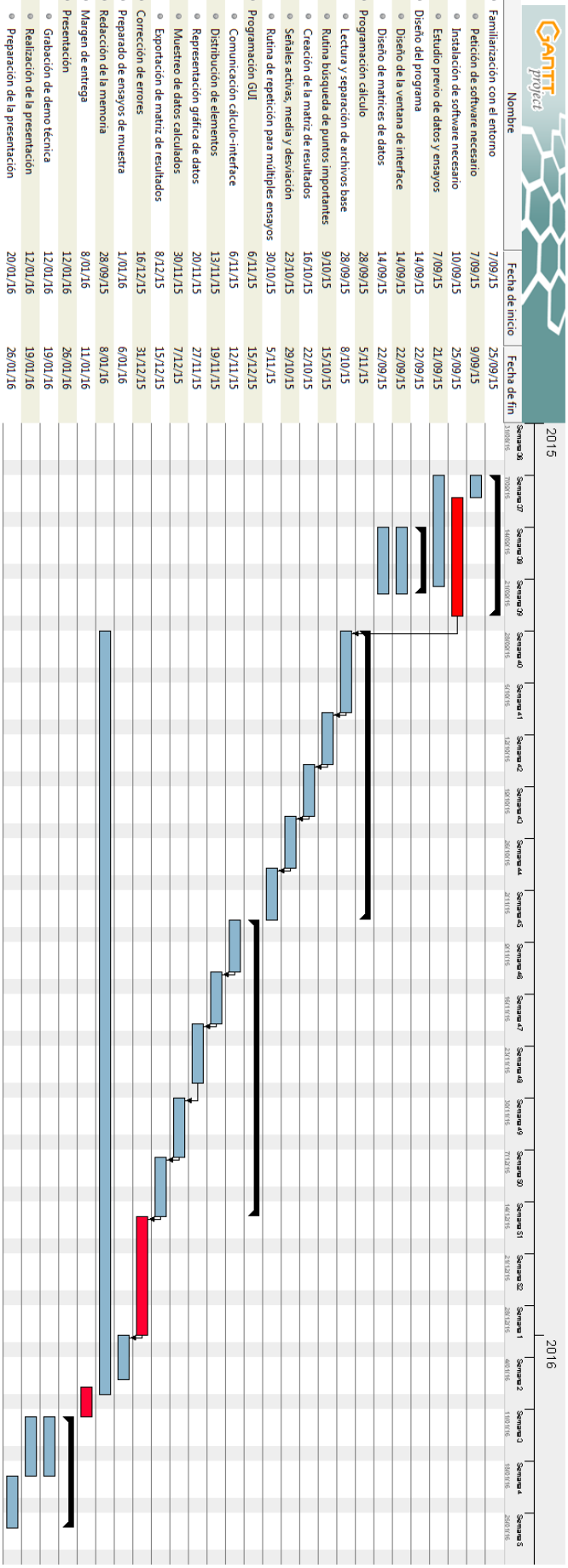


Figura 7.2. *Diagrama de Gantt final.*

CAPÍTULO 8: COSTES DE DESARROLLO

A la hora de evaluar los costes de desarrollo de este proyecto, la realidad es que no requiere de ningún material físico con un coste tangible, ya que se trata de una aplicación de software. Sin embargo pueden evaluarse los costes de desarrollo a nivel de horas y licencias necesarias para desarrollar el programa.

Por un lado, la herramienta de desarrollo es MATLAB, un software que tiene un precio para empresas elevado. Depende de las prestaciones contratadas y el tipo de uso pero en general el coste de una licencia de Matlab para la versión 2013 asciende a 6000 €. Para el programa base creado en este proyecto no se necesitan complementos adicionales, pero de cara a una aplicación como las comentadas en el apartado 8.2, el precio de cada complemento ronda a los 3000 € de media.

Por otro lado las horas invertidas en el desarrollo de este programa se han calculado a partir del Diagrama de Gantt final, a razón de 5 horas por día laborable (jornada reducida) durante el periodo de trabajo en la empresa. El resultado son 80 días trabajados y se cuantifica el precio de cada hora como la media de hora de ingeniería, incluyendo recursos utilizados y sueldo, con un total de 50 €/h.

Entonces, el precio final es el siguiente.

$$Coste = 6000 + 50 * 80 * 5 = 26000\ €$$

El coste final puede parecer elevado pero es necesario evaluar las ventajas que implica la creación de este software. Dado el creciente interés en el desarrollo de estas tecnologías, cada vez son más los ensayos que se realizan en esta área. Al mismo tiempo, por cada ensayo se generan una cantidad muy grande de datos que, absolutamente todos, deben

ser analizados. El análisis de datos se realizaba, hasta la fecha, con programas de visualización y muchas horas de ingeniero trasladando datos que, manualmente, buscaba y calculaba a partir de estas visualizaciones.

Así pues y desde ahora, la generación de tablas de valores, el acondicionamiento de las señales en una visualización clara y agradable, y el almacenamiento de las conclusiones extraídas se genera de forma automática y por tanto el ahorro de tiempo es más que significativo. Es muy difícil traducir este ahorro en valores monetarios pero una posible aproximación pasa por considerar que se generan 1 o 2 ensayos al día, los cuales presentan de media 30 casos de carga distintos y todos ellos deben ser analizados.

Una persona sola necesita una media de 20 minutos de análisis por cada caso de carga suponiendo que algunos no aportan mucha información pero otros deben analizarse en detalle para explicar errores y comportamientos atípicos. Además se requiere tiempo para generar un reporte con todos los datos extraídos escritos a mano. En total, y solo para un mes de trabajo, se puede deducir lo siguiente:

$$CosteAnalisis_{mes} = 1_{ensayo/día} 30_{casos/día} * 0{,}2_{h} * 22_{días} * \ 50_{€/h} = \ 6600\ €$$

Entonces, eliminar estos pasos supone un ahorro en horas de 6600 €/mes, cerca de unos 80000 €/año, con lo que la inversión inicial queda cubierta en aproximadamente 4 meses de uso a pleno rendimiento.

CAPÍTULO 9: CONCLUSIONES

9.1. Evaluación de objetivos

Finalizado el proyecto, es el momento de repasar los objetivos que, en un primer momento se establecieron para este y compararlos con el resultado final.

- **Automatización de análisis de ensayos AEB:** se ha conseguido el objetivo principal de cálculo automático de obtención de valores y extracción de conclusiones de ensayos AEB. Después de evaluar el programa con datos reales se han obtenido matrices de datos analizados con validez para la empresa y ahorrando una considerable cantidad de tiempo y trabajo manual.

  Los valores de estas tablas de resultados coinciden con los obtenidos de forma manual y por tanto se considera que los cálculos son fiables y robustos. Pese a que el programa no es capaz de explicar el porqué de valores anómalos sí que puede hacer al usuario prestar atención a estas señales que pueden aportar información sobre fallos críticos del sistema.

- **Interfaz de usuario:** La interfaz final permite al usuario la visualización de datos sin necesidad de acceder, en ningún momento, a la programación textual del programa. Los botones y tablas son intuitivas y sencillas de manejar, por lo que no requieren de conocimiento en el programa base Matlab para utilizarse.

- **Visualización previa de datos:** con la interface final el usuario puede representar las señales que desee, tanto una por una como superponiendo cualquiera de ellas, ya sean del mismo ensayo o de otro distinto. Además, el

usuario posee diversas herramientas de visualización, tales como alejar o ampliar el gráfico actual, leer valores concretos sobre las curvas por medio de un cursor interactivo, y mover libremente la visualización del gráfico en cualquier dirección del eje cartesiano.

Además se puede borrar y mantener/sobreponer las señales con solo pulsar un botón. Del mismo modo se representan los valores de interés warning y brake con una recta de colores distintos de forma visual y práctica.

- **Facilidad de ampliación:** en esta área el programa se ha estructurado en forma de capítulos o secciones con comentarios detallados sobre la función de cada parte. Por otro lado se procura guardar dentro del espacio de trabajo todas las variables y matrices de datos utilizadas para que pueda accederse a ellas posteriormente, por ejemplo, guardando los datos *data* de todos los ensayos con su identificador correspondiente para que no importe si la variable *data* sufre cambios. En los casos en los que sí se eliminan variables, se utiliza el comando *clear* en una sola línea y al final de su sección, para vaciar el espacio de trabajo de variables innecesarias al final de la compilación y aportar así claridad.

  A nivel de variables, se ha procurado dotar a cada una de un nombre que sugiera el valor contenido o el estado señalado.

  De ese modo se asegura que, en caso de una ampliación posterior, el programador sea capaz de localizar el punto en el cual desea continuar y tenga a su disposición todas las variables y matrices de datos necesarias. También facilita este desarrollo la programación orientada a objetos de la interface GUIDE ya que, al recoger el código de cada objeto dentro de una función propia, pueden añadirse funcionalidades nuevas sin necesidad de intervenir en el funcionamiento de otros objetos ya creados si no es necesario.

- **Adaptación a los formatos necesarios:** desde el programa de adquisición de datos se pueden obtener estos en distintos formatos. El que posee más compatibilidad con otros programas es el .csv, el cual se ha establecido como formato de entrada de los archivos de ensayos a cargar en el programa. Muchos programas de adqusisición y visualización tienen la opción de escribir datos en este formato y es el más adecuado en previsión de un cambio futuro en el programa de adquisición que se viene utilizando hasta la fecha.

  Además, el formato de salida de datos es .xlsx, un formato que puede transformase de forma trivial en .csv por medio de herramientas al alcance de todos los usuarios, como Microsoft Excel.

- **Herramienta centralizada:** no se debe olvidar que esta herramienta de cálculo es exclusiva del cliente y por tanto se debe prestar atención a la confidencialidad de los datos que contiene. Para evitar que entes externos puedan apropiarse del programa, se han establecido dentro de él rutinas de búsqueda de las matrices de datos a partir de los nombres de las variables que contienen. Estos nombres son propios de la empresa y por tanto, si no coinciden con los introducidos en los archivos de entrada, el programa no funciona correctamente. Además para

cambiar estas especificaciones sí debe poseerse el programa original y no ninguna copia "runtime" que pueda derivarse.

Por otro lado el programa está muy especializado en encontrar los valores que la empresa ha considerado que son importantes. No ofrece la posibilidad de localizar puntos distintos ni valores que puedan ser de interés para otras entidades ya que esta opción no es prioritaria para el cliente que encarga este proyecto. En caso de necesitar valores nuevos, la empresa puede contar con su creador para añadir estas nuevas funcionalidades. En definitiva, el programa no resulta útil a entes externos que no compartan los procedimientos internos con los utilizados en este proyecto.

En términos generales, el desarrollo de este proyecto ha sido largo y difícil dada la gran cantidad de prestaciones de las que se ha querido dotar al programa y el aprendizaje autónomo que ha debido realizar el autor, que si bien conocía las bases de la programación en lenguaje Matlab, ha necesitado ampliar estos en gran medida y aprender por completo el funcionamiento de la herramienta GUIDE para la creación de aplicaciones.

Sin embargo, el programa final se califica de éxito dados los resultados que se han obtenido con datos reales y que se han utilizado para aplicaciones útiles a la empresa. Además del aprendizaje que ha aportado a su autor, no solo en términos de lenguaje de programación sí no también en el campo del AEB y el funcionamiento interno dentro de una gran empresa.

9.2. Trabajos de futuro

Las posibilidades de ampliación del programa son muchas y diversas. Esto es consecuencia de la relativa juventud del campo, por la que aparecen nuevas situaciones de ensayo, requerimientos de sistema o datos de interés a cada momento. Tanto es así que se presentan a continuación gran variedad de posibles ampliaciones posteriores a este proyecto, algunas de ellas descartadas en el momento de definir los límites de este, y otras que han aparecido como idea y deberán evaluarse de cara al futuro. Hay que recordar que este proyecto tenía como objetivo sentar las bases de una aplicación útil en un campo cuyos requerimientos son cambiantes actualmente.

- *Mejoras en la visualización*

Las mejoras que podrían implementarse en la visualización de señales (ventana de la interface), son tantas como gustos tienen los usuarios potenciales de programa. Puede adaptarse la vista a cada usuario de forma infinita, por ejemplo, añadiendo la opción de cambiar los colores de las curvas en el gráfico (que inicialmente son solo de color negro), la elección de valores concretos de la tabla MM y no representar todos ellos, el recordé de la señal visualizada a un intervalo de tiempo concreto, la visualización simultanea de dos gráficos distintos, la eliminación de señales innecesarias, y un largo etcétera.

Es difícil determinar cuáles de todas estas medidas son prioritarias y cuáles no vale la pena implementar. Es necesario que se de uso al programa base creado en este proyecto para que surjan necesidades y se decida que debe implementarse y que puede esperar.

- *Detección de valores anómalos*

A partir de los valores de media y desviación estándar se barajó la idea en un principio de que el programa sea capaz de detectar, dentro de la matriz MM, que valores representan un resultado anómalo o están fuera de lo esperado, facilitando así al usuario que analice las señales más problemáticas e intente determinar que ha salido mal en el ensayo.

Esta es una aplicación muy útil que no se ha implementado por falta de tiempo y dificultad, ya que implica que se introduzcan en el programa toda una serie de parámetros y condiciones que determinen cuando un valor se considera anómalo o digno de análisis, teniendo en cuenta que las variables son de magnitudes y unidades diversas y que algunas varían considerablemente en función de la velocidad del ensayo. Con todo, sería necesario un estudio propio para determinar estas condiciones, hecho que escapa al objetivo del proyecto pero que podría realizarse en un futuro próximo.

- *Creación automática de reportes*

Otra opción que se desea implementar en un futuro es la de crear reportes en un formato de texto (por ejemplo .pdf) y que presenten un informe detallado de los ensayos analizados, conteniendo gráficos, tablas de valores, medidas de dispersión entre ensayos, comparativas y texto. Falta decidir que debe y que no debe aparecer en este reporte y con que herramienta se quiere implementar, pero una buena opción podría ser el complemento *Matlab Report Generator*, que tiene este mismo objetivo y se puede implementar en el propio programa base de desarrollo.

La compra de licencia de este complemento requiere una inversión por parte de la empresa, hecho que ha retrasado su implementación.

- *Uso del programa para usuarios sin Matlab*

Hasta el momento, todos los usuarios potenciales de este proyecto poseen en sus puestos de trabajo Matlab con licencia instalada. Sin embargo, y dada la creciente relevancia del campo de la seguridad activa, es muy probable que en un futuro otros usuarios necesiten tener acceso a este programa pero no puedan justificar la inversión que requiere una licencia de Matlab en su equipo. Por ese motivo, se considera la trasformación del programa en una versión ejecutable (.exe) que pueda compilarse desde la versión "run time" de Matlab, de licencia gratuita.

Sin embargo para crear esta versión ejecutable es necesario un complemento de Matlab llamado *Matlab Compiler* que de nuevo requiere una inversión por parte de la empresa.

- *Almacenamiento de entornos de trabajo*

Otra opción interesante de futuro es el almacenamiento de entornos de trabajo creados con el programa al añadir muchos ensayos y realizar los cálculos pertinentes. Hasta el momento solo los resultados obtenidos pueden guardarse como tabla y de hecho cumple el objetivo necesario. No obstante, es posible que en el futuro las cantidades de datos sean muy elevadas o se añadan funcionalidades al programa que necesiten guardar el entorno de trabajo para continuar con el análisis en otro momento sin necesidad de cargar de nuevo los ensayos o repitiendo acciones sobre ellos.

No es el caso del programa actual y por eso no se ha implementado esta opción, pero queda propuesta como posible ampliación.

REFERENCIAS BIBLIOGÁFICAS

1. "Support and help". Mahtworks. (Última consulta 22/11/2015).
   http://es.mathworks.com/help/matlab/

2. Group Euro NCAP. Euro NCAP. (Última consulta 15/10/2015).
   http://www.euroncap.com/es

3. *Matlab Aspectos Basicos*. *Escuela Universitaria de Ingeniería Técnica Industrial de Bilbao* (Última consulta 19/11/2015).
   http://www.sc.ehu.es/sbweb/energias-renovables/MATLAB/intro.html

4. *Matlab GUIS. Joseline Lega. Universidad de Arizona.* (Última consulta 5/12/2015).
   http://math.arizona.edu/~lega/MATLAB_GUIs/MATLAB_GUIs.html

5. *ECU Development & Test with CANoe.* Vector. *(Última consulta 10/12/2015).*
   http://vector.com/vi_canoe_en.html

6. *EB Assist ADTF.* Elektrobit. *(Última consulta 10/12/2015).*
   https://www.elektrobit.com/products/eb-assist/adtf/

7. *NI Diadem.* National Instruments. *(Última consulta 11/12/2015).*
   http://www.ni.com/diadem/esa/

8. *Excel Specifications and Limits*. Microsoft *(Última consulta 11/12/2015).*
   https://support.office.com/en-nz/article/

9. *European New Car Assessment Programme. Test Protocol AEB Systems.* Euro NCAP. Versión 1.1. Junio de 2015.

10. *European New Car Assessment Programme. Test Protocol AEB VRU Systems.* Euro NCAP. Versión 1.0.1 Junio de 2015.

Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Apéndices

APLICACIÓN PARA ANÁLISIS Y CLASIFICACIÓN DE DATOS EN ENSAYOS DE CHOQUE

TFG presentado para optar al título de GRADO en INGIENERÍA Electrónica Industrial y Automática
por **Sergi Delgado Rodríguez**

Barcelona, 11 de Enero de 2015

Tutor proyecto: Gerard Escudero Bakx
Departamento de LSI
Universitat Politècnica de Catalunya (UPC)

ÍNDICE APÉNDICES

Apéndice A: Código de cálculo 2
ApÉndice B: Código de interface 15

APÉNDICE A:
CÓDIGO DE CÁLCULO

A continuación se presenta el código perteneciente al bloque de cálculo del proyecto tal y como se encuentra en el archivo .m.

```

% CARTEL DE PRESENTACIÓN

if exist ('Ntest','var') == 0
   uiwait (msgbox({'AEB Analysis Data Tool ', 'Wellcome',' ',' ', 'Created by Sergi Delgado'}))
   uiwait (msgbox('Please select REF file'))
end


%% CARGAR ARCHIVOS .CSV DE ENTRADA
% Apartado donde se cargan los tres archivos que pertenecen a un mismo
% ensayo.

%Cargar archivo REF
%Elección del archivo en la carpeta de trabajo
FileName = uigetfile('*.csv');

newData1 = importdata(FileName);

% Creación de nuevas variables para contener los datos en el workspace.
vars = fieldnames(newData1);
for i = 1:length(vars)
    assignin('base', vars{i}, newData1.(vars{i}));
end

%Asignación de nombre para las matrices de datos
colheadersREF = colheaders;
textdataREF = textdata;
dataREF = data;
clear colheaders textdata data

% eliminar valores nan de data
ind=find(isnan(dataREF));
dataREF(ind)=0;



%Cargar archivo EGO

%Cambio de nombre para cargar automáticamente el mismo ensayo elegido en REF
token=strtok(FileName,'R');
FileName1 = strcat(token,'EGO','.csv');

newData1 = importdata(FileName1);

% Creación de nuevas variables para contener los datos en el workspace.
vars = fieldnames(newData1);
for i = 1:length(vars)
    assignin('base', vars{i}, newData1.(vars{i}));
end

%Asignación de nombre para las matrices de datos
colheadersEGO = colheaders;
textdataEGO = textdata;
```

```
dataEGO = data;

% eliminar valores nan de data
ind=find(isnan(dataEGO));
dataEGO(ind)=0;
clear colheaders textdata data



%Cargar archivo FKT
%uiwait (msgbox('Insertar archivo FKT'))
token=strtok(FileName,'R');
FileName2 = strcat(token,'FKT','.csv');

newData1 = importdata(FileName2);

% Create new variables in the base workspace from those fields.
vars = fieldnames(newData1);
for i = 1:length(vars)
    assignin('base', vars{i}, newData1.(vars{i}));
end

colheadersFKT = colheaders;
textdataFKT = textdata;
dataFKT = data;
%Eliminar valores nan de data
ind=find(isnan(dataFKT));
dataFKT(ind)=0;

%Limpieza de variables
clear colheaders textdata data


%% CÁLCULO DE PUNTOS DE INTERÉS
% Se calculan los puntos donde se produce el Warning, se inicia la acción
% del Teilbremsung y la acción del Zielbremsung

% Index es una variable que indica que columna de la matriz de datos
% (magnitud concreta) estamos utilizando


%Encontrar tiempo de valor maximo del canal Vorwarnung_Sensor
index = strmatch( ' Vorwarnung_Sensor', char(colheadersFKT));
dataFKT(1:500,index) = 0; %eliminamos los primeros +/- 5 segundos para evitar
valores de puesta a punto
%que muchas veces generan errores
maxwarning=max(dataFKT(:,index));
timewarning_index=find(dataFKT(:,index)==maxwarning,1,'first');
time=dataFKT(:,1);
timewarning=time(timewarning_index);
%imponemos que sea 0 para valores bajos evitando anomalías de valores
%pequeños que no representan un flanco de subida real.
if timewarning<0.01
    timewarning = 0;
    WarningActive = false;
```

```
else
    WarningActive = true;
end
clear maxwarning time

%Encontrar tiempo de valor maximo del canal  TBR_Freigabe_Sensor
index = strmatch( ' ATB_Freigabe_Sensor', char(colheadersFKT));
dataFKT(1:500,index) = 0; %eliminamos los primeros +/- 5 segundos para evitar valores de puesta a punto
%que muchas veces generan errores
maxESP=max(dataFKT(:,index));
timeESP_index=find(dataFKT(:,index)==maxESP,1,'first');
time=dataFKT(:,1);
timeESP=time(timeESP_index);
%imponemos que sea 0 para valores bajos evitando anomalías de valores
%pequeños
if timeESP<0.01
    timeESP = 0;
    TBRActive = false;
else
   TBRActive = true;
end
clear maxESP time

%Encontrar tiempo de valor máximo del canal  ZBR_Freigabe_Sensor
index = strmatch( ' ZBR_Freigabe_Sensor', char(colheadersFKT));
dataFKT(1:500,index) = 0; %eliminamos los primeros +/- 5 segundos para evitar valores de puesta a punto
%que muchas veces generan errores
maxZBR=max(dataFKT(:,index));
timeZBR_index=find(dataFKT(:,index)==maxZBR,1,'first');
time=dataFKT(:,1);
timeZBR=time(timeZBR_index);
%imponemos que sea 0 para valores bajos evitando anomalías de valores
%pequeños
if timeZBR<0.05
    timeZBR = 0;
    ZBRActive = false;
else
   ZBRActive = true;
end

clear maxZBR time

%% DEFINICIÓN DE VARIABLES BOOLEANAS
% Se definen variables booleanas necesarias para la comunicación con la
% interface usuario o útiles en el propio programa


%Para la interface usuario
hold = false;
warning = false;
ESP = false;
TestShow = 1; %Variable necesaria para la interface que nos indicará que test se está visualizando
```

```

%Para el programa de cálculo
impacto = false;


%% CREACIÓN DE LA MATRIZ DE RESULTADOS


%La matriz de resultados es la más importante y muestra de manera ordenada
%todos los parámetros que deben encontrarse y analizarse a partir de los
%datos iniciales. Cada fila de esta matriz representa un ensayo diferente.
%por eso se debe poder guardar las filas en el espacio de trabajo y cargar
%nuevos ensayos para insertar nuevas filas.

%creación de la estructura de vector de resultados (M) con números y letras

M = {'name','0',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,'Comments',' '};

%DATOS GENERALES

%Columna 2, velocidad del ensayo. Separa el string para dejar solo la
%velocidad
str= textdataREF(8,1);
text= strtok(str, 'record.long_description=GRA=');
Token= strtok(text, ' /');

M(1,2)=Token;

%Columna 1, nombre del ensayo
M(1,1)=strcat(token, ' ',M(1,2));

%Columna 3, Velocidad de impacto. Se necesita determinar si hay impacto.
%Para eso se evalúa primero la distancia mínima relativa entre el coche y
%el target, valor que se representa en la columna 4, para determinar si se
%ha producido un choque, en el momento en que la velocidad se estabiliza en 0.

index = strmatch( ' Dx01' , char(colheadersREF));
Y = dataREF(:,index);

%Rutina de localización de cambio de signo en la posición del vehículo en X
a=[];
b=[];
ta=0;
tb=0;

for i=2:length(Y)
    if Y(i-1)>0&Y(i)<0
        ta=ta+1;
        a(ta,1)=i-1;
        a(ta,2)=i;
    else
        if Y(i-1)<0 & Y(i)>0
            tb=tb+1;
            b(tb,1)=i-1;
            b(tb,2)=i;
```

```
        else
        end
    end
end

% a es entonces el punto en el que se ha producido impacto o su valor es 0
% si no se ha producido

if length(a)==0 %NO hay impacto

    %Se busca la distancia a la que se mantiene el coche una vez
    %estabilizada la velocidad a 0 (en realidad se considera 0 a un valor
    %inferior a 0.1)
    index = strmatch( ' Vx01' , char(colheadersREF));
    Y = dataREF(:,index);
    v0=find(Y<0.1,1,'last');

    index = strmatch( ' Dx01' , char(colheadersREF));
    Y = dataREF(:,index);
    Distance = Y(v0);
    Impactspeed = 0;
    impact = false;

else %Si hay impacto
    Distance = 0;

    index = strmatch( ' Vx01' , char(colheadersREF));
    Y = dataREF(:,index);
    Impactspeed = Y(a(1,1));

    impact = true;
end

M(1,3) = {Impactspeed};

%Columna 4, distancia longitudinal en STOP

M(1,4) = {Distance};

%-----------------------------------------------------------------------

%MEDIDAS DEL WARNING/TBR/ZBR
VectorSignals = {' Vorwarnung_Sensor',' ATB_Freigabe_Sensor', '↙
ZBR_Freigabe_Sensor'};
VectorPoints = [timewarning,timeESP,timeZBR];

%Columna 5/10/15, TTC. Es necesario crear un canal nuevo de TTC para poder
%buscar después los 3 que se necesitan.

%acotar el número de valores al tiempo de REF
time=dataREF(:,1);
aux = size (time);
maxtime = aux(1);
%Crear un canal nuevo TTC del tamaño de time
TTC1 = zeros(maxtime,1);
```

```
TTC2 = zeros(maxtime,1);

%Aplicar la ecuación del MRUA a cada punto capturado en el tiempo
for i=1:maxtime

    %ECUACIÓN DE SEGUNDO GRADO
    %Obtención de variables

    index = strmatch( ' Dx01' , char(colheadersREF));
    Y = dataREF(:,index);
    X= Y(i);
    index = strmatch( ' Vx01' , char(colheadersREF));
    Y = dataREF(:,index);
    V0=Y(i);
    index = strmatch( ' Ax01' , char(colheadersREF));
    Y = dataREF(:,index);
    A=Y(i);
    %Solucionar la ecuación y tomar el valor positivo
    C=[A/2, V0, X];
    value =max(roots(C));
    %aplicar al canal TTC el valor obtenido
    TTC1(i) = value;

end
%ECUACIÓN DE PRIMER GRADO
index = strmatch( ' Dx01' , char(colheadersREF));
X = dataREF(:,index);
index = strmatch( ' Vx01' , char(colheadersREF));
Vx = dataREF(:,index);
TTC2 = rdivide(abs(X),abs(Vx));
%clear A C X X0 V0 Vx


for i=1:3

    time=dataREF(:,1);
    val = VectorPoints(i); %value to find
    if val == 0
        break
    else
        tmp = abs(time-val);
        [idx idx] = min(tmp); %index of closest value
        closest = time(idx); %closest value
Y=TTC2;

%Cargamos el resultado en la posición de M correspondiente segun i
            if i == 1
                TTCWarning= Y(idx);
                M(1,5) = {TTCWarning};
            end
            if i== 2
                TTCTBR= Y(idx);
                M(1,10) = {TTCTBR};
            end
            if i== 3
```

```
                TTCZBR= Y(idx);
                M(1,15) = {TTCZBR};
            end
    end
end



%Columna 6/11/16, Distancia

for i=1:3
   time=dataREF(:,1);
    val = VectorPoints(i); %value to find
    if val == 0
        break
    else
        tmp = abs(time-val);
        [idx idx] = min(tmp); %index of closest value
        closest = time(idx); %closest value

        index = strmatch( ' Dx01' , char(colheadersREF));
        Y = dataREF(:,index);

        %Cargamos el resultado en la posición de M correspondiente según i
            if i == 1
                XWarningREF=Y(idx);
                M(1,6) = {XWarningREF};
            end
            if i== 2
                XTBRREF=Y(idx);
                M(1,11) = {XTBRREF};
            end
            if i== 3
                XZBRREF=Y(idx);
                M(1,16) = {XZBRREF};
            end
    clear tmp idx
    end
end

%Columnas 7/12/17, Duración

%Tenemos calculado el instante en que se produce el flanco de subida. Se
%calcula el flanco de bajada y se resta el tiempo

for i=1:3

    index = strmatch( VectorSignals(i), char(colheadersFKT));
    maxwarning=max(dataFKT(:,index));
    %imponemos que sea 0 para valores bajos evitando anomalías de valores
    %pequeños
    if maxwarning ~= 0
        timewarningEnd_index=find(dataFKT(:,index)==maxwarning,1,'last');
        time=dataFKT(:,1);
        timewarningEnd=time(timewarningEnd_index);
```

```
        clear maxwarning time timewarningEnd_index
        %Cargamos el resultado en la posición de M correspondiente según i
        if i == 1
            DuracionWarning = timewarningEnd - timewarning;
            M(1,7) = {DuracionWarning};
        end
        if i== 2
           DuracionTBR = timewarningEnd - timeESP;
           M(1,12) = {DuracionTBR};
        end
        if i== 3
           DuracionZBR = timewarningEnd - timeZBR;
           M(1,17) = {DuracionZBR};
        end
    end
end

%Columna 8/13/18, velocidad en el momento de activación según vehículo
% utilizamos la variable timewarning/timeAEB/timeZBR que hemos calculado para la
% representación gráfica

%pasar a la escala de tiempo de REF. idx es la posición y closest el valor de
%tiempo más próximo encontrado

for i=1:3

    time=dataREF(:,1);
    val = VectorPoints(i); %value to find
    if val == 0
        break
    else
        tmp = abs(time-val);
        [idx idx] = min(tmp); %index of closest value
        closest = time(idx); %closest value

        index = strmatch( ' Vx01' , char(colheadersREF));
        Y = dataREF(:,index);

        %Cargamos el resultado en la posición de M correspondiente según i
            if i == 1
                VWarningREF=Y(idx);
                M(1,8) = {VWarningREF*3.6}; %importante pasar a km/h
            end
            if i== 2
                VTBRREF=Y(idx);
                M(1,13) = {VTBRREF*3.6}; %importante pasar a km/h
            end
            if i== 3
                VZBRREF=Y(idx);
                M(1,18) = {VZBRREF*3.6}; %importante pasar a km/h
            end
    clear tmp idx
    end
end
%Columna 9/14/19, velocidad en el momento de activación según GPS
```

```

%En este caso no es necesaria la transformación de escala de tiempos ya que
%la V_ESP_FZG ya se encuentra en el entorno SKT

for i=1:3

index = strmatch( ' v_ESP_Fzg' , char(colheadersFKT));
Y = dataFKT(:,index);

%Cargamos el resultado en la posición de M correspondiente segun i
            if i == 1
                VWarningFKT= Y(timewarning_index);
                M(1,9) = {VWarningFKT*3.6}; %importante pasar a km/h
            end
            if i== 2
                VTBRFKT= Y(timeESP_index);
                M(1,14) = {VTBRFKT*3.6}; %importante pasar a km/h
            end
            if i== 3
                VZBRFKT= Y(timeZBR_index);
                M(1,19) = {VZBRFKT*3.6}; %importante pasar a km/h
            end
end

%MEDIDAS DE LOS PERFILES DE FRENADA


%Columna 20, Max value perfil de frenada en g

index = strmatch( ' Sollverzoegerung_Sensor' , char(colheadersFKT));
Y = dataFKT(:,index);
M(1,20) = {min(Y)/9.81};

%Columna 21, Max value perfil de frenada en m/s^2

index = strmatch( ' Sollverzoegerung_Sensor' , char(colheadersFKT));
Y = dataFKT(:,index);
M(1,21) = {max(abs(Y))};

%Columna 22, Max value perfil de frenada real

index = strmatch( ' v_ref_ESP_Fzg' , char(colheadersFKT));
Y =(dataFKT(:,index));
M(1,22) = {max(Y)/9.81};

%Columna 23, Delay entre 7 y 8. Encuentra el tiempo de los valores mínimos y
%los copara

time=dataFKT(:,1);

%min perfil de frenada
index = strmatch( ' Sollverzoegerung_Sensor', char(colheadersFKT));
eval1 = cell2mat(M(1,20));
eval1 = eval1*9.81;
eval1index = find(dataFKT(:,index)==eval1,1);
```

```
eval1t=time(eval1index);

%min perfil de frenada real
index = strmatch( ' v_ref_ESP_Fzg', char(colheadersFKT));
eval2 = cell2mat(M(1,22));
eval2 = eval2*9.81;
eval2index = find(dataFKT(:,index)==eval2,1);
eval2t=time(eval2index);

%Resta para saber que retraso tiene un máximo respecto al otro
delay = eval2t-eval1t;
M(1,23) = {delay};

clear eval1 eval2 eval1index eval2index eval1t eval2t

%Columna 24, Comments

str= textdataREF(8,1);
[Token remain]= strtok(str, '=');
[Token remain]= strtok(remain, '=');
M(1,24)=Token;

%CONCADENACIÓN DE RESULTADOS

%Vector de headlines para las columnas de M

Mh = {'Test name','Test speed [km/h]','Impact speed [km/h]','Dist,Long.(x) en STOP',...
    'Warning TTC[s]','Warning Distance[m]','Warning Duration [s]',...
    'Warning V_Signal[km/h]','Warning Speed [km/h]','TBR TTC [s]','TBR Distance [m]',...
    'TBR Duration [s]','TBR V_Signal[km/h]','TBR Speed [km/h]','ZBR TTC [s]',...
    'ZBR Distance[m]','ZBR Duration [s]','ZBR V_Signal[km/h]','ZBR Speed [km/h]',...
    'Braking Profile ANB MAX [g]','Braking Profile ANB MAX [m/s^2]',...
    'Braking Profile TBR MAX [g]','Delay [ms]','Comments',' ',};
%Acumula las filas en MM de cada ensayo. Si es el primero crea MM
if exist ('MM','var') == 0
   MM = M;
else
    MM= [MM; M];
end


%% PREDICIONES Y EVALUACIÓN
%Encontrar el valor medio de los valores obtenidos en MM para las
%magnitudes que nos interesan.


pointer = [3,4,5,6,10,11,15,16];
MMeanh = {'Impact speed [km/h]','Dist,Long.(x) en STOP','Warning TTC[s]','Warning Distance[m]',...
    'TBR TTC [s]','TBR Distance[m]','ZBR TTC [s]','ZBR Distance[m]'};

%Evaluar la media de cada valor entre todos los ensayos
```

```
for i=pointer

    aux=cell2mat(MM(:,i));
    aux = mean (aux);
    if exist ('MMean','var') == 0
        MMean = [aux];
    else

        MMean = [MMean, aux];
    end
end
clear aux

%Evaluar la dispersión de cada valor entre todos los ensayos
for i=pointer

    aux=cell2mat(MM(:,i));
    aux = std (aux);
    if exist ('MDisp','var') == 0
        MDisp = [aux];
    else

        MDisp = [MDisp, aux];
    end
end
clear aux

%Escritura de resumen de resultados del ensayo

Result = {0;0;0;0};
if impact
    Result {1} = 'IMPACT';
else
    Result {1} = 'NO IMPACT';
end
if WarningActive
    Result {2} = 'Warning Active';
else
    Result {2} = 'Warning Inactive';
end
if TBRActive
    Result {3} = 'TBR Active';
else
    Result {3} = 'TBR Inactive';
end
if ZBRActive
    Result {4} = 'ZBR Active';
else

    Result {4} = 'ZBR Inactive';
end

% Se guarda para cada ensayo

if exist ('ResultData','var') == 0
```

```
    ResultData = Result;
else
    ResultData = [ResultData, Result];
end




%% SECUÉNCIA DE GUARDADO Y CARGA DE NUEVO ENSAYO

%Matriz TName que registra los nombres de los ensayos cargados
if exist ('TName','var') == 0
   TName = strcat(token, ' ',M(1,2));
else
    TName= [TName; strcat(token, ' ',M(1,2))];
end


%AÑADIR NUEVO ENSAYO
%Para poder encadenar vectores M de datos analizados en los ensayos se crea
%un contador de ensayos (Ntest) y una matriz que engloba los títulos de M
%(Mh) y todos los vectores M, la matriz MM.

%Pregunta al usuario si quiere añadir un nuevo ensayo. SI lo añade, NO
%cierra el diálogo y permite operar normalmente. No contestar permite
%seguir operando con la interface GUIDE

%Para cada nuevo ensayo se debe calcular los valores de la matriz M y se
%pueden borrar una vez guardados en MM, pero los entornos de datos deben
%guardarse idexados para poder ser visualizados en la aplicación.

if exist ('Ntest','var') == 0
   Ntest = 1;
end



%Almacenamiento de datos
Name2Save = strcat('colheadersREF', '_', num2str(Ntest));
assignin('base',Name2Save,colheadersREF);
Name2Save = strcat('colheadersEGO', '_', num2str(Ntest));
assignin('base',Name2Save,colheadersEGO);
Name2Save = strcat('colheadersFKT', '_', num2str(Ntest));
assignin('base',Name2Save,colheadersFKT);
Name2Save = strcat('textdataREF', '_', num2str(Ntest));
assignin('base',Name2Save,textdataREF);
Name2Save = strcat('textdataEGO', '_', num2str(Ntest));
assignin('base',Name2Save,textdataEGO);
Name2Save = strcat('textdataFKT', '_', num2str(Ntest));
assignin('base',Name2Save,textdataFKT);
Name2Save = strcat('dataREF', '_', num2str(Ntest));
assignin('base',Name2Save,dataREF);
Name2Save = strcat('dataEGO', '_', num2str(Ntest));
assignin('base',Name2Save,dataEGO);
Name2Save = strcat('dataFKT', '_', num2str(Ntest));
```

```
assignin('base',Name2Save,dataFKT);

Name2Save = strcat('timewarning', '_', num2str(Ntest));
assignin('base',Name2Save,timewarning);
Name2Save = strcat('timeESP', '_', num2str(Ntest));
assignin('base',Name2Save,timeESP);

%% EJECUCIÓN DE LA INTERFACE GUIDE

APLICACION;


prompt={'Do you wish to add one more essay? YES/NO'};
name='ADD ESSAY';
numlines=1;
defaultanswer={'YES'};

options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';

answer=inputdlg(prompt,name,numlines,defaultanswer,options);

answerIndex= ['YES','NO'];
index = strmatch( answer, answerIndex);

if index == 1
    Ntest = Ntest + 1;
    clear MMean;
    clear MDisp;
    Main
end

```

APÉNDICE B:
CÓDIGO DE INTERFACE

A continuación se presenta el código perteneciente al bloque de interface usuario del proyecto tal y como se encuentra en el archivo .m.

```
function varargout = APLICACION(varargin)
% APLICACION M-file for APLICACION.fig
%      APLICACION, by itself, creates a new APLICACION or raises the existing
%      singleton*.
%
%      H = APLICACION returns the handle to a new APLICACION or the handle to
%      the existing singleton*.
%
%      APLICACION('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in APLICACION.M with the given input arguments.
%
%      APLICACION('Property','Value',...) creates a new APLICACION or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before APLICACION_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to APLICACION_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help APLICACION

% Last Modified by GUIDE v2.5 03-Dec-2015 10:02:57

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @APLICACION_OpeningFcn, ...
                   'gui_OutputFcn',  @APLICACION_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before APLICACION is made visible.
function APLICACION_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to APLICACION (see VARARGIN)

% Choose default command line output for APLICACION
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);



% UIWAIT makes APLICACION wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = APLICACION_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


%--- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)

%Carga de variables
global Datatable;
guidata(hObject,handles)
Workgroup=handles.Workgroup;
guidata(hObject,handles)
Namegroup=handles.Namegroup;
guidata(hObject,handles)

%Selección de canal
seleccionado=get(handles.listbox1,'Value');
%Dibujo de señal
handles.P=plot(Workgroup(:,1),Workgroup(:,seleccionado));
guidata(hObject,handles)
set(handles.P,'Color','k')

%Tabla de valores
```

```
hold = evalin('base','hold');
newrow =  [Namegroup(seleccionado), max(Workgroup(:,seleccionado)),min(Workgroup(:,seleccionado))];
if hold == 1
    Datatable= [Datatable; newrow];
else
    Datatable = newrow;
end

set(handles.uitable1, 'Data', Datatable);

%Introducción línea de warning
warning = evalin('base','warning');
global W

if warning==1
    %carga del tiempo de warning
    Workgroup=handles.Workgroup;
    guidata(hObject,handles)

    seleccionado=get(handles.listbox1,'Value');

    TestShow = evalin('base','TestShow');
    handles.TestShow=TestShow;
    guidata(hObject,handles)

    Name2Load = strcat('timewarning', '_', num2str(TestShow));
    timewarning = evalin('base',Name2Load);
    handles.timewarning=timewarning;
    guidata(hObject,handles)



    %Evalúa si los rangos deben ampliarse debido a la escala del grafico
    %resultante


    DatatableMatMin=min(cell2mat(Datatable(:,3)));
    DatatableMatMax=max( cell2mat(Datatable(:,2)));
    W=line([timewarning timewarning],[DatatableMatMin DatatableMatMax],'color','r','linewidth',2);
else
    W=line([0 0],[0 0],'color','r','linewidth',2);
end

%Introducción línea ESP
ESP = evalin('base','ESP');
global E


if ESP==1
    %carga del tiempo de warning
    Workgroup=handles.Workgroup;
    guidata(hObject,handles)
```

```
    seleccionado=get(handles.listbox1,'Value');



    TestShow = evalin('base','TestShow');
    handles.TestShow=TestShow;
    guidata(hObject,handles)

    Name2Load = strcat('timeESP', '_', num2str(TestShow));
    timeESP = evalin('base',Name2Load);
    handles.timeESP=timeESP;
    guidata(hObject,handles)

    %Evalúa si los rangos deben ampliarse debido a la escala del grafico
    %resultante


    DatatableMatMin=min(cell2mat(Datatable(:,3)));
    DatatableMatMax=max( cell2mat(Datatable(:,2)));

    if (DatatableMatMin==0 && DatatableMatMax == 0)
        E=line([timeESP timeESP],[-1 1],'color','g','linewidth',2);

    else

        E=line([timeESP timeESP],[DatatableMatMin↙
DatatableMatMax],'color','g','linewidth',2);
    end
end


% --- Executes on button press in SwitchREF.
function SwitchREF_Callback(hObject, eventdata, handles)
% hObject    handle to SwitchREF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.listbox1,'String', handles.colheadersREF)
%Establecer el workspace de REF
handles.Workgroup = handles.dataREF;
guidata(hObject,handles)
handles.Namegroup = handles.colheadersREF;
guidata(hObject,handles)


% --- Executes on button press in SwitchEGO.
function SwitchEGO_Callback(hObject, eventdata, handles)
% hObject    handle to SwitchEGO (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.listbox1,'String', handles.colheadersEGO)
%Establecer el workspace de EGO
handles.Workgroup = handles.dataEGO;
guidata(hObject,handles)
handles.Namegroup = handles.colheadersEGO;
```

```
guidata(hObject,handles)


% --- Executes on button press in SwitchFKT.
function SwitchFKT_Callback(hObject, eventdata, handles)
% hObject    handle to SwitchFKT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.listbox1,'String', handles.colheadersFKT)
%Establecer el workspace de FKT
handles.Workgroup = handles.dataFKT;
guidata(hObject,handles)
handles.Namegroup = handles.colheadersFKT;
guidata(hObject,handles)


% --- Executes on button press in Load.
function Load_Callback(hObject, eventdata, handles)
% hObject    handle to Load (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%introducir archivo REF


TestShow = evalin('base','TestShow');
handles.TestShow=TestShow;
guidata(hObject,handles)

Name2Load = strcat('colheadersREF', '_', num2str(TestShow));
colheadersREF = evalin('base',Name2Load);
handles.colheadersREF=colheadersREF;
guidata(hObject,handles)

Name2Load = strcat('textdataREF', '_', num2str(TestShow));
textdataREF = evalin('base',Name2Load);
handles.textdataREF=textdataREF;
guidata(hObject,handles)

Name2Load = strcat('dataREF', '_', num2str(TestShow));
dataREF = evalin ('base',Name2Load);
handles.dataREF=dataREF;
guidata(hObject,handles)

set(handles.SwitchREF,'BackgroundColor','g');
set(handles.SwitchREF,'Enable', 'on');

%introducir archivo EGO
TestShow = evalin('base','TestShow');
handles.TestShow=TestShow;
guidata(hObject,handles)

Name2Load = strcat('colheadersEGO', '_', num2str(TestShow));
colheadersEGO = evalin('base',Name2Load);
handles.colheadersEGO=colheadersEGO;
guidata(hObject,handles)
```

```
Name2Load = strcat('textdataEGO', '_', num2str(TestShow));
textdataEGO = evalin('base',Name2Load);
handles.textdataEGO=textdataEGO;
guidata(hObject,handles)

Name2Load = strcat('dataEGO', '_', num2str(TestShow));
dataEGO = evalin ('base',Name2Load);
handles.dataEGO=dataEGO;
guidata(hObject,handles)

set(handles.SwitchEGO,'BackgroundColor','g');
set(handles.SwitchEGO,'Enable', 'on');

%introducir archivo FKT

TestShow = evalin('base','TestShow');
handles.TestShow=TestShow;
guidata(hObject,handles)

Name2Load = strcat('colheadersFKT', '_', num2str(TestShow));
colheadersFKT = evalin('base',Name2Load);
handles.colheadersFKT=colheadersFKT;
guidata(hObject,handles)

Name2Load = strcat('textdataFKT', '_', num2str(TestShow));
textdataFKT = evalin('base',Name2Load);
handles.textdataFKT=textdataFKT;
guidata(hObject,handles)

Name2Load = strcat('dataFKT', '_', num2str(TestShow));
dataFKT = evalin ('base',Name2Load);
handles.dataFKT=dataFKT;
guidata(hObject,handles)

set(handles.SwitchFKT,'BackgroundColor','g');
set(handles.SwitchFKT,'Enable', 'on');


%Cargar todas las matrices de datos en su correspondiente figura

TName = evalin('base','TName');
set (handles.TestsBox,'String',TName);

MM = evalin('base','MM');
set (handles.uitable2,'Data',MM)
Mh = evalin('base','Mh');
set (handles.uitable2,'ColumnName',Mh)

MMean = evalin('base','MMean');
MDisp = evalin('base','MDisp');
ADD = [MMean;MDisp];
set (handles.MeanTable,'Data',ADD);
MMeanh = evalin('base','MMeanh');
set (handles.MeanTable,'ColumnName',MMeanh);
```

```

%Cargar resultados
TestShow = evalin('base','TestShow');
guidata(hObject,handles)
ResultData = evalin('base','ResultData');
set (handles.TextResult,'String',ResultData(:,TestShow));


% --- Executes on button press in EraseButton.
function EraseButton_Callback(hObject, eventdata, handles)
% hObject    handle to EraseButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

 delete(allchild(handles.axes1));
 set(handles.uitable1,'Data',[]);
 clear global Datatable


% --- Executes on button press in Warningbutton.
function Warningbutton_Callback(hObject, eventdata, handles)
% hObject    handle to Warningbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Esta rutina busca el punto en el que se produce el warning (calculado al
%inicio del programa) y ajusta una línea vertical recta al grafico actual
%indicando ese punto entre el máximo y el mínimo de la escara mayor
%representada
warning = evalin('base','warning');
warning = not(warning);
assignin('base', 'warning', warning)

global W minwarning maxwarning
global Datatable

if warning==1
    %carga del tiempo de warning
    Workgroup=handles.Workgroup;
    guidata(hObject,handles)

    seleccionado=get(handles.listbox1,'Value');

    TestShow = evalin('base','TestShow');
    handles.TestShow=TestShow;
    guidata(hObject,handles)

    Name2Load = strcat('timewarning', '_', num2str(TestShow));
    timewarning = evalin('base',Name2Load);
    handles.timewarning=timewarning;
    guidata(hObject,handles)

    %Evalúa si los rangos deben ampliarse debido a la escala del grafico
    %resultante

```

```
    DatatableMatMin=min(cell2mat(Datatable(:,3)));
    DatatableMatMax=max( cell2mat(Datatable(:,2)));
        if (DatatableMatMin==0 && DatatableMatMax == 0)
        W=line([timewarning timewarning],[-1 1],'color','r','linewidth',2);

        else

        W=line([timewarning timewarning],[DatatableMatMin↙
DatatableMatMax],'color','r','linewidth',2);
        end
else
     listbox1_Callback(hObject, eventdata, handles);
     clear W;
end


% --- Executes on button press in Hold.
function Hold_Callback(hObject, eventdata, handles)
% hObject    handle to Hold (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Cambio del indicador hold y el modo de visualización del axes
hold = evalin('base','hold');
hold = not(hold);

if hold == 1
    set(handles.axes1,'NextPlot', 'add');
    set(handles.Hold,'BackgroundColor','g');
else
    set(handles.axes1,'NextPlot', 'replace');
    set(handles.Hold,'BackgroundColor',[0.4,0.4,0.4]);
end

assignin('base', 'hold', hold)


% --- Executes on button press in ESPButton.
function ESPButton_Callback(hObject, eventdata, handles)
% hObject    handle to ESPButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ESP = evalin('base','ESP');
ESP = not(ESP);
assignin('base', 'ESP', ESP)

global E minESP maxESP
global Datatable

if ESP==1
    %carga del tiempo de warning
    Workgroup=handles.Workgroup;
    guidata(hObject,handles)

    seleccionado=get(handles.listbox1,'Value');
```

```
    TestShow = evalin('base','TestShow');
    handles.TestShow=TestShow;
    guidata(hObject,handles)

    Name2Load = strcat('timeESP', '_', num2str(TestShow));
    timeESP = evalin('base',Name2Load);
    handles.timeESP=timeESP;
    guidata(hObject,handles)


    %Evalúa si los rangos deben ampliarse debido a la escala del grafico
    %resultante


    DatatableMatMin=min(cell2mat(Datatable(:,3)));
    DatatableMatMax=max( cell2mat(Datatable(:,2)));

        if (DatatableMatMin==0 && DatatableMatMax == 0)
        E=line([timeESP timeESP],[-1 1],'color','g','linewidth',2);

        else

        E=line([timeESP timeESP],[DatatableMatMin↙
DatatableMatMax],'color','g','linewidth',2);
        end

else
    listbox1_Callback(hObject, eventdata, handles);
    clear E;

end



% --- Executes on button press in Export.
function Export_Callback(hObject, eventdata, handles)
% hObject    handle to Export (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

 MM = evalin('base','MM');
 Mh = evalin('base','Mh');
 filename = 'Exportación.xlsx';

 xlswrite(filename,Mh,1,'A1')
 xlswrite(filename,MM,1,'A2')

 winopen(filename)



% --- Executes on selection change in TestsBox.
function TestsBox_Callback(hObject, eventdata, handles)
% hObject    handle to TestsBox (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Cambia el ensayo actual mostrado
TestShow = evalin('base','TestShow');
handles.TestShow=TestShow;
guidata(hObject,handles)

TestShow=get(handles.TestsBox,'Value');

assignin ('base', 'TestShow', TestShow);

%Deshabilita todos los botones hasta que se cargue el ensayo
set(handles.SwitchEGO,'BackgroundColor','r');
set(handles.SwitchEGO,'Enable', 'off');
set(handles.SwitchREF,'BackgroundColor','r');
set(handles.SwitchREF,'Enable', 'off');
set(handles.SwitchFKT,'BackgroundColor','r');
set(handles.SwitchFKT,'Enable', 'off');

set(handles.listbox1,'String', ' ')
```