Implementación de un sistema autónomo para el control de un vehículo eléctrico comercial.

Javier Vidal, Pedro J. Navarro, Carlos Fernández, Pedro Sánchez Palma
E-mail: javi.vidal@upct.es; pedroj.navarro@upct.es; carlos.fernandez@upct.es; pedro.sanchez@upct.es

Resumen *– El presente artículo presenta el desarrollo de un sistema autónomo para el control de un vehículo eléctrico comercial basado en una plataforma CompactRIO y programado en LabVIEW. El sistema autónomo incorpora los algoritmos: evitación de obstáculos, procesado de información GPS, procesado de imágenes, generación de trayectorias sobre mapas vectoriales, etc. El trabajo aquí expuesto ha sido desarrollado íntegramente por la UPCT y ha permitido validar diferentes algoritmos y arquitecturas hardware/software de manera eficiente. Además resulta una plataforma docente de gran interés para el desarrollo de proyectos académicos*

1. Introducción

Los vehículos autónomos han despertado un gran interés en la actualidad. La idea de un vehículo capaz de trazar rutas evitando los obstáculos que vayan surgiendo en el camino, es muy tentadora a la vez que complicada. La Universidad de Virginia Tech ha desarrollado diferentes proyectos (*Odin* y *Blinddriver)* relacionados con vehículos autónomos en el que ha participado profesores y alumnos. El proyecto Odin [1] logró un tercer puesto en el prestigioso concurso Urban Challenge [2] organizado por la *Defense Advanced Research Projects Agency* (DARPA). El proyecto *Blinddriver* desarrolló un vehículo semiautomático para discapacitados visuales, en el que un chaleco vibro-táctil informaba al conductor de parámetros como la velocidad y el grado de frenado del vehículo. Ambos proyectos se caracterizan por la utilización de controladores RT embebidos (compactRIO) y el software de programación LabVIEW de NI [3].

El grupo de investigación DSIE (División de Sistemas en Ingeniería Electrónica) dispone en la actualidad de diversas plataformas móviles de carácter docente e industrial para el desarrollo e investigación en el área de la robótica móvil. En este trabajo se presenta la implementación hardware/software para el desarrollo de un vehículo eléctrico autónomo (ver figura 1).

Fig. 1: Vehículo eléctrico autónomo.

2. Arquitectura del sistema.

La figura 2 muestra la arquitectura desarrollada para el control del vehículo autónomo. Esta está compuesta principalmente por cuatro elementos: unidad de control, unidad de gobierno, unidad de comunicaciones y unidad sensorial. A continuación se procederá a describir en detalle cada una de ellas:

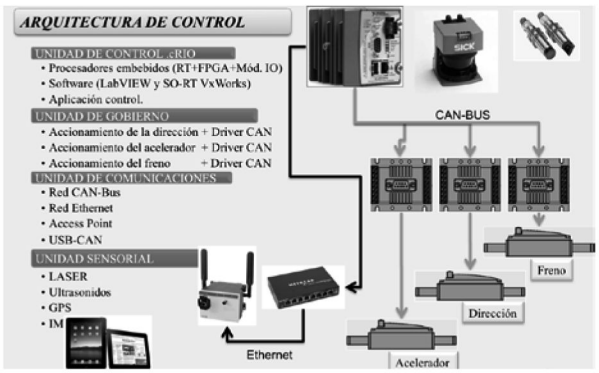

Fig. 2: Arquitectura del sistema de control del vehículo.

2.1 Unidad de control

La unidad de control está basada un controlador programable automatizado (PAC) CompactRIO de National Instruments [4]. Este controlador ha sido diseñado para aplicaciones que requieren alto rendimiento y fiabilidad. El hardware del CompactRIO está constituido por tres componentes:

1. Un procesador Real-Time. El sistema embebido CompactRIO tiene un procesador industrial Freescale MPC5200 de 400 MHz que ejecuta de manera determinista aplicaciones escritas en LabVIEW. Opera en el sistema operativo Wind River VxWorks en tiempo real. LabVIEW tiene funciones integradas que permiten la comunicación entre la FPGA y el procesador en tiempo real.

2. Un chasis gobernado por una FPGA (Xilinx) La FPGA gobierna un chasis al que se conectan los módulos de E/S. El procesador RT a través de la FPGA tiene acceso al procesamiento de las

señales locales con objeto de realizar las toma de decisiones del sistema.

3. Módulos de E/S. Los módulos de E/S abarcan todo tipo de configuraciones, topolgías y protocolos (digitales, analógicas, CAN, RS485, Wireless, PROFIBUS, GPS, etc.).

2.2 Unidad de gobierno

La unidad de gobierno permite a la unidad de control actuar mecánicamente sobre los accionamientos del vehículo (acelerador, freno y dirección). Para desarrollar la unidad de gobierno ha sido necesario modificar mecánicamente el vehículo eléctrico objeto de la automatización. La figura 3 muestra los motores instalados en el vehículo para la robotización del acelerador, freno y dirección.

Cada uno de los motores instalados posee un driver con salida CAN conectado a la unidad de comunicaciones del vehículo.

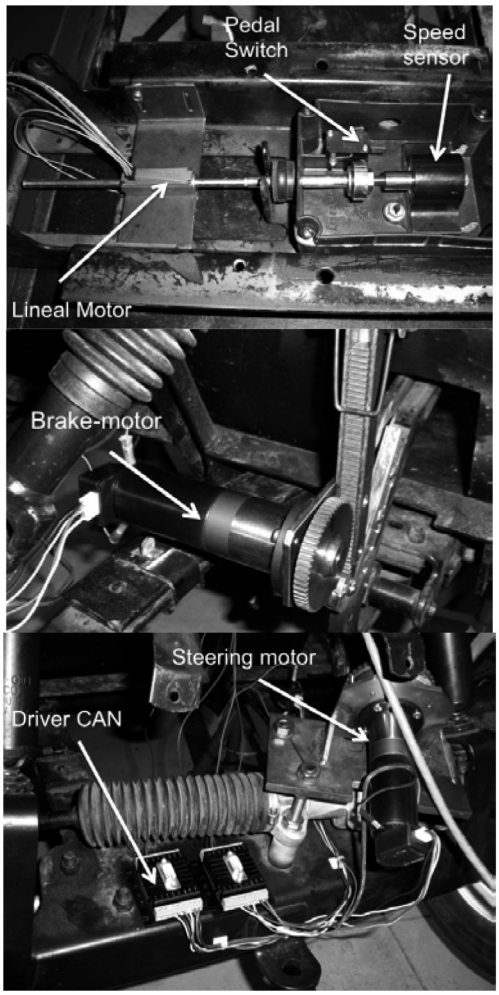

Fig. 3: Modificaciones mecánicas en los accionamientos del vehículo.

2.3 Unidad de comunicaciones

Para la comunicación del cRIO con el resto del vehículo se han utilizado varios estándares de comunicación.

CANOpen. CANOpen [5] es un protocolo de comunicación industrial que implementa la capa de aplicación para el bus CAN. En CANOpen cada mensaje tiene un identificador que define su prioridad y al nodo al que va dirigido. Los mensajes con un identificador más bajo tienen más prioridad. El protocolo se encarga de resolver los conflictos y de distribuir los mensajes entre los nodos. Este protocolo se ha utilizado para enviar las órdenes de control a los motores. Se han definido 3 nodos, uno para cada accionamientos: acelerador, freno y volante.

TCP/IP. El protocolo TCP/IP se ha utilizado para el intercambio de información con el sistema de telemetría láser (LIDAR) instalado en el vehículo, para recibir las órdenes de movimiento en el modo de manejo teleoperado (vía Wi-Fi), y como comunicación con la estación de programación del vehículo.

RS232. El protocolo serie RS232 se utiliza para la recepción de las tramas enviadas por el GPS instalado en el vehículo.

2.4 Unidad sensorial.

El vehículo consta de los siguientes sensores instalados abordo:

LASER (LIDAR). Permite crear un mapa del entorno por el que se desplaza el vehículo. El LASER instalador posee un rango de medición de 20 metros en 270º con precisión 0.25º y 1mm.

Anillo de ultrasonidos. Los 8 sensores ultrasónicos permiten al vehículo detectar elementos a una distancia menor a 2 metros del vehículo.

GPS y giróscopo. El GPS y el giróscopo instalados en el vehículo permite conocer la posición global y local del mismo en cada instante.

Sistema de visión. El sistema de visión permite detectar y clasificar los objetos que se encuentren frente al vehículo (personas, señales de tráfico, etc.)

3. Software

La implementación de los algoritmos para el control del vehículo han sido realizados en LabView. Para ello, se ha creado un programa para la FPGA que recoge los datos de los puertos de E/S y otro para el procesador RT que toma las decisiones de control.

En el procesador RT se han programado varias librerías que realizan diversas funciones para llevar a cabo el control del vehículo.

Una de estas librerías desarrolladas se encarga de la comunicación con los motores vía CANOpen. La estructura de dicha librería puede observarse en la figura 4.

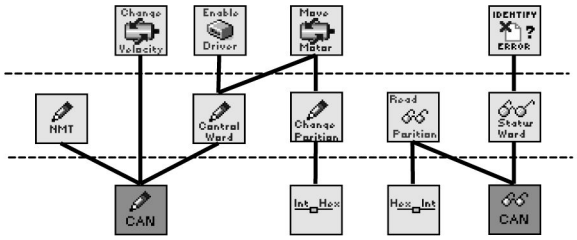

Fig. 4. Jerarquía de la librería creada para el control de motores

Otra de las librerías (figura 5) establece la comunicación y realiza el tratamiento de los datos recibidos del LIDAR. Esta librería recibe continuamente los datos del LIDAR y los interpreta para construir un mapa en 2D del entorno. Este mapa se muestra posteriormente en la interfaz del usuario.

Para la navegación del vehículo, se ha implementado un algoritmo de evitación de obstáculos conocido como Vector Field Histogram (VFH) [6]. Este algoritmo toma la información del láser y la representa en un gráfico XY. La coordenada X representa el ángulo de visión y la coordenada Y la distancia al objeto en dicho ángulo. En la figura 6 puede observarse un ejemplo del resultado de la implantación del algoritmo VFH en la interfaz de usuario. Como puede observarse, buscando un mínimo en el histograma es posible hallar una ruta a través de los obstáculos.

a) b) c)

Fig. 5. a) Fotografía tomada desde el vehículo, b) Mapa obtenido de los datos del LIDAR, c) Histograma construido a partir de dichos datos.

Para la realización de misiones sencillas en ambiente semi-estructurados se ha incorporado a la interfaz un planificador de ruta. El planificador, una vez fijada la misión sobre el entorno, divide esta en submisiones de menor complejidad. Estas submisiones son suministradas al VFH para el cálculo de las trayectorias del vehículo en cada instante. Los mapas de las misiones son cargados en forma dos archivos, uno correspondiente a la imagen del entorno en formato JPG y otro con datos de configuración. Estos datos deben incluir: posición GPS del centro de la imagen, tamaño en píxeles de la imagen y zoom con la que ha sido tomada la imagen.

La planificación de la ruta es fijada por el usuario marcando sobre la imagen precargada del entorno, el punto inicial y final de la misión (ver figura 7). A continuación, el planificador selecciona la ruta óptima mediante el algoritmo A* [7].

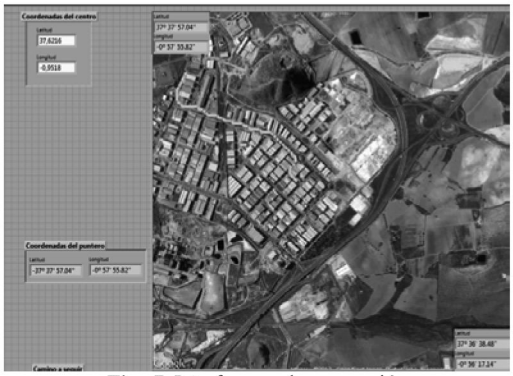

Fig. 7. Interfaz para la navegación

4. Conclusiones

El presenta trabajo muestra el desarrollo de un sistema autónomo implementado sobre un vehículo eléctrico comercial. El sistema desarrollado ha permitido validar diferentes algoritmos y arquitecturas hardware/software de manera eficiente dicha plataforma. Además este sistema resulta un entorno docente de gran interés para el desarrollo de proyectos académico en diversa disciplinas de la ingeniería.

5. Agradecimientos

Este trabajo ha sido llevado a cabo como parte de los proyectos EXPLORE (CICYT-TIN2009-08572) y MISSION-SICUVA (SENECA ref. 15374/PI/10).

6. Referencias

[1] National Instrument. Odin. http://sine.ni.com/cs/app/doc/p/id/cs-11323

[2] DARPA. http://www.darpa.mil/

[3] NI LabVIEW Robotics for Autonomous Vehicle Applications. http://zone.ni.com/wv/app/doc/p/id/wv-1131

[4] http://sine.ni.com/nips/cds/view/p/lang/es/nid/208044

[5] "*CANopen: high level protocol for CAN-bus*".*http://www.nikhef.nl/pub/departments/ct/po/doc/CANopen30.pdf*

[6] Borenstein, J.; Koren, Y. (1991). *"The vector field histogram-fast obstacle avoidance for mobilerobots"*. Robotics and Automation, IEEE Transactions on 7 (3): 278–288. doi:10.1109/70.88137

[7] "A* Path finding tutorial" http://www.policyalmanac.org/games/aStarTutorial.htm