ISBN 978-84-15914-12-9

SOBRE LA VULNERABILIDAD DE LOS ROBOTS MÓVILES FRENTE A LOS ATAQUES HARDWARE

F. Gomez-Bravo, Raúl Jiménez Naharro, J. Medina García, J. A. Gómez Galán, M. Sánchez Raya

Depto. Ingeniería Electrónica, Sistemas Informáticos y Automática.
Escuela Técnica Superior de Ingeniería,
Universidad de Huelva. Carretera de Palos-La Rábida s/n. 21071 Huelva.
{fernando.gomez; naharro; jonathan.medina; jgalan; msraya }@diesia.uhu.es

Resumen

Este trabajo presenta un estudio sobre distintos tipos de ataques a la seguridad de un sistema robótico. El trabajo presta especial atención al estudio de ataques hardware, y en concreto analiza las características vulnerables del protocolo I2C. Se presenta un análisis de dicha vulnerabilidad cuando los ataques se realizan sobre un robot móvil de conducción diferencial que pretende realizar un seguimiento de trayectorias. El trabajo demuestra cómo, con ataques de este tipo, es posible alterar el comportamiento del robot manteniendo la integridad del mismo y sin que la acción deje ningún rastro.

Palabras Clave: Ataques Hardware, Robótica Móvil, Seguridad y Fiabilidad en Robots.

1 INTRODUCCIÓN

En la actualidad, las arquitecturas de control para robots se basan en el uso de microprocesadores. La manera más común de atacar este tipo de sistemas se fundamenta en el uso de virus informáticos, que tradicionalmente afectan a la ejecución del software que corre en el procesador. La solución obvia contra este ataque es utilizar un software antivirus actualizado. Sin embargo, hasta ahora, la mayoría de los robots no incluían mecanismos específicos para mejorar su seguridad frente a estas amenazas informáticas. Esta tendencia está cambiando debido al elevado uso de sistemas robóticos en aplicaciones con información sensible [9, 11] (como bancos, entornos militares o sistemas de vigilancia). Es por esto que, los atacantes de sistemas informáticos (tradicionalmente conocidos como piratas informáticos o "*hackers*") están considerando la posibilidad aplicar la tecnología de ataque a sistemas robóticos en general. Teniendo en cuenta que los virus aparecen en primer lugar, y que con posterioridad se desarrolla el sistema antivirus, existe una alta probabilidad de que algún ataque aparezca en el tiempo de adaptación del software antivirus. Dependiendo del sistema a proteger, este tiempo de vulnerabilidad puede ser intolerable. Por lo tanto, una posible solución se encuentra en la implementación de un sistema totalmente hardware que no incluya la ejecución de ningún software. Obviamente, esta solución evita la posible ejecución de cualquier ataque software. Sin embargo, los sistemas totalmente hardware pueden ser también atacados por piratas informáticos, utilizando los llamados ataques hardware.

El principal objetivo de este trabajo se encuentra en el estudio de la influencia de los ataques de hardware en sistemas robóticos.

Un ataque hardware es un ataque utilizando las vulnerabilidades de una implementación hardware. Los principales objetivos de estos ataques pueden ser muy diferentes, pero todos se pueden resumir en el mismo efecto: el comportamiento final del sistema es diferente al deseado por los usuarios. Algunos ejemplos pueden ser:

- Transmitir información sensible a personas no autorizadas.
- El aumento de la vulnerabilidad de un algoritmo de cifrado o en general de un proceso de comunicación.
- Obtención de información sensible en el interior del sistema.

Existen muchos mecanismos para implementar un ataque hardware: el conocido como hardware troyano (que consiste en incluir un bloque de hardware malicioso en el sistema con el fin de alterar su comportamiento global) [12]; ataques de replay (que consisten en suplantar la identidad de un bloque autorizado por un bloque no autorizado en un proceso de comunicación) [2]; ataques de injección de fallos (que consisten en inyectar un fallo en el sistema para aumentar su vulnerabilidad) [1, 7]; la ingeniería de inversa (que consiste en obtener información del comportamiento interno del sistema a partir de los valores de determinadas señales) [6].

ISBN 978-84-15914-12-9

Concretamente, en este trabajo se muestra la vulnerabilidad respecto a algunos de estos ataques sobre el protocolo I2C, ampliamente utilizado en los sistemas robóticos. De hecho, el uso de controladores I2C de bajo nivel en robótica móvil se ha extendido ampliamente en los últimos años. Aún más, la fiabilidad de los robots móviles es un tema abierto y recientemente discutido por la comunidad científica. Como consecuencia, la robótica móvil es un contexto natural para el estudio de la influencia de los errores en la seguridad del bus I2C en la fiabilidad de dichos sistemas.

La aplicación que se presenta en este artículo está relacionada con la navegación de robots móviles a lo largo de una trayectoria definida previamente. Este escenario representa una situación típica en muchas de las aplicaciones actuales de robótica (industria, agricultura, servicios, etc.), realizadas ya sea en aplicaciones en exteriores o en escenarios interiores [3, 4]. Se supone que las intenciones del usuario implican la visita de ciertas áreas del escenario por parte del robot. Por lo tanto, se supone también que un algoritmo de planificación ha proporcionado un camino, y que un algoritmo de seguimiento de ruta se aplica de manera que el robot sigue esta trayectoria con suficiente precisión.

En consecuencia, aquellos que quisieran interferir en las intenciones del usuario, podrían tratar de modificar el curso del robot, de tal manera que el usuario no se diera cuenta de la existencia de una interferencia externa (únicamente detectaría un cambio en el comportamiento del robot). Es decir, el controlador de alto nivel mantendría invariable la ruta planificada, mientras que la interferencia externa se aplicaría al actuador de bajo nivel durante un corto período de tiempo. Esta pequeña duración impediría el descubrimiento de una interferencia externa. El cambio puntual de comportamiento del robot puede ser repetir unos determinados movimientos o evitar su paso por algún lugar. Fuera de estos cambios puntuales, el robot mantendría la ruta planificada.

2 ARQUITECTURA DE CONTROL PARA ROBOTS MÓVILES Y EL PROTOCOLO I2C

2.1 ARQUITECTURA DE CONTROL PARA ROBOTS MÓVILES

De acuerdo con la arquitectura tradicional de una plataforma robótica móvil [3, 4, 10], (ver Fig. 1) los controladores de alto nivel son los responsables de tomar decisiones sobre el movimiento del robot, así como establecer las acciones correspondientes para ser ejecutadas por los actuadores. En el caso de tener motores como actuadores, y dependiendo de la

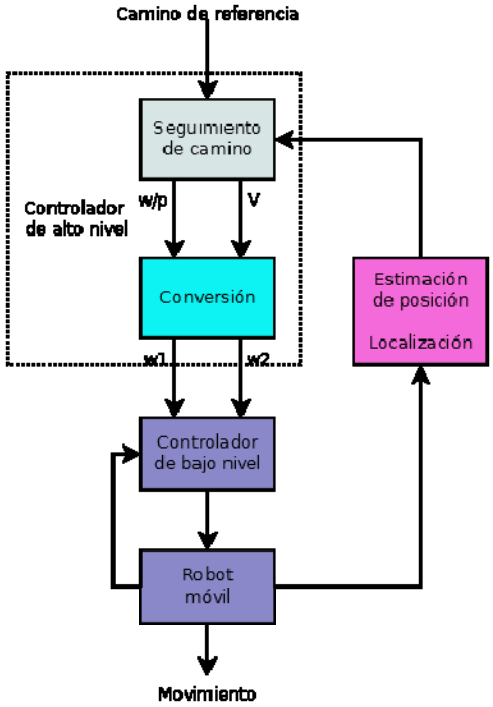


Figura 1: Arquitectura tradicional de control de un robot móvil.

aplicación robótica, el controlador de alto nivel define la velocidad del motor y/o la posición angular del eje del mismo. Estos valores se transfieren al controlador de bajo nivel, de modo que los motores hacen que el robot se mueva de una manera correcta. Normalmente, ambas funcionalidades, control de alto y bajo nivel se implementan en CPUs diferentes.

Concretamente, en el mercado se pueden encontrar dispositivos electrónicos específicos para el control de bajo nivel aplicado a motores [8]. En estas circunstancias, debe existir un procedimiento de comunicación entre una y otra CPU para la transferencia de estos datos, y un canal de comunicación que soporte este proceso. En este contexto, la protección de esta comunicación es una cuestión relevante que tiene que ser tomada en cuenta para el desarrollo de robots fiables y seguros. Hay numerosos ejemplos robóticos donde se aplican controladores I2C de bajo nivel para establecer las acciones de los actuadores. En este trabajo, los conductores I2C se utilizan para controlar la velocidad de las ruedas de un robot móvil. Sobre éste, se realizan ataques temporales contra el proceso de comunicación entre el algoritmo de seguimiento de caminos y el controlador I2C, con el fin de cambiar el curso final del robot. El ataque será ejecutado de tal manera que no se dañe ni la estructura del robot, ni en el hardware de control. El usuario debe percibir el ataque como un comportamiento extraño del robot pero nunca como el resultado de una interferencia externa.

ISBN 978-84-15914-12-9

2.2 PROTOCOLO I2C

El protocolo I2C se basa en dos señales, generalmente llamadas SDA (línea de datos) y SCL (línea de reloj) y eventualmente una señal de tierra (todos los módulos deben tener la misma referencia de voltaje). Todos los módulos están conectados a las mismas líneas del bus, como se muestra en la Fig. 2a, y en general, cualquier módulo permiten la conexión de un nuevo módulo al bus. Existen dos tipos de módulos: un módulo principal que genera la señal SCL y controla las transmisiones; y uno o varios módulos esclavos que serán la fuente (operaciones de lectura) o destino (operaciones de escritura) de la información. La señal SDA es generada tanto por el maestro como por los módulos esclavos cuando se produce la comunicación. Los valores lógicos en el protocolo I2C son: tierra para el nivel bajo nivel y alta impedancia para el nivel alto. La utilización de alta impedancia implica que los diferentes módulos no tienen que utilizar la misma fuente de polarización y, por lo tanto, los módulos que necesitan alta tensión se pueden conectar al mismo bus que los módulos necesarios de baja tensión.

Las características iniciales de protocolo I2C (un protocolo abierto y el uso de alta impedancia para el nivel alto) representan una alta vulnerabilidad en el sistema. En primer lugar, un protocolo abierto permite la conexión de cualquier módulo (autorizado o no autorizado) al bus. En segundo lugar, la utilización de alta impedancia como nivel lógico permite sobrescribir la información usando niveles bajos sin ninguna consecuencia en la señalización del protocolo.

El comportamiento de una operación de escritura usando un protocolo I2C se ilustra en la Fig. 2b) y

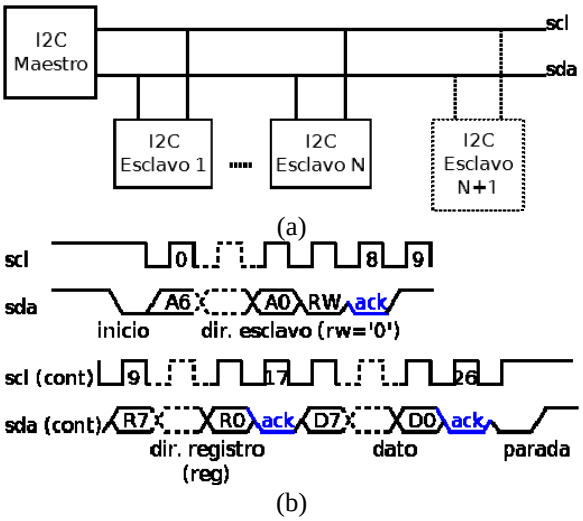


Figura 2: (a) Arquitectura de un sistema basado en el protocolo I2C; (b) señales del procedimiento de escritura en el protocolo I2C.

está ampliamente descrito en la literatura científica y técnica [5].

3 ATAQUES HARDWARE

3.1 ATAQUES EN EL PROTOCOLO I2C

En la aplicación considerada, el *hacker* debe realizar varias acciones con el fin de garantizar la acción correcta del ataque. En primer lugar, el pirata informático debe elegir el momento de inicio del ataque. Este momento se puede obtener utilizando la técnica de ingeniería inversa (para determinar el comportamiento del sistema). En segundo lugar, el pirata informático debe evitar que las órdenes del módulo maestro lleguen al módulo esclavo. Esta acción implica el uso de hardware troyano con el fin de poner en práctica el ataque. El ataque consiste en inyectar un fallo que modifique la frecuencia en la línea del reloj (señal SCL). Por último, el hardware troyano enviará el acuse de recibo al módulo maestro como si lo hubiese hecho el módulo esclavo, es decir, se trata de un ataque de replay.

Según lo descrito, el ataque no puede ser realizado directamente por el *hacker*, debido a la movilidad del sistema considerado. Por lo tanto, el ataque debe ser implementado en un módulo como un módulo troyano. Generalmente, un troyano se incluye en la fase de fabricación por un empleado (con autorización, o sin autorización). Sin embargo, el sentido abierto del protocolo I2C permite incluir cualquier módulo troyano después de la fase de fabricación, es decir, durante la fase de utilización. Una vez que el módulo troyano (módulo de ataque a partir de ahora) está instalado en el bus, el *hacker* puede controlar el ataque sin necesidad de acceder físicamente al sistema. Obviamente, este módulo no tiene una dirección asociada, ya que es desconocido para el resto del sistema.

El módulo de ataque tiene tres funciones básicas para garantizar la pérdida puntual de comunicaciones con un determinado esclavo. En primer lugar, el módulo controlará las transmisiones como un esclavo común. Esta función de supervisión tiene como fin detectar la comunicación con el esclavo a atacar. La dirección correcta y el momento del ataque se pueden determinar usando técnicas de ingeniería inversa porque el módulo de ataque tiene acceso a la transmisión de datos. El módulo de ataque puede enviar esta información a un *hacker* por un canal inalámbrico para procesarlo. A continuación, el *hacker* puede enviar al módulo de ataque la información necesaria (dirección y el momento de ataque). En nuestro caso, el módulo de ataque ya tiene la dirección del esclavo a atacar (el controlador de motor) y la orden de ataque llegará en tiempo real.

ISBN 978-84-15914-12-9

En segundo lugar, en el caso de un ataque, el módulo debe impedir la comunicación entre el maestro y el módulo esclavo. La comunicación se impide evitando los cambios en la señal de SCL haciendo que esta señal tome un nivel bajo. Esto es posible debido que el nivel lógico alto es la situación de alta impedancia. Con esta acción, el primer paquete responsable del direccionamiento no se habrá recibido y no habrá ningún esclavo que envíe el mensaje de acuse de recibo. Esta acción puede ser vista como un ataque mediante inyección de fallo variando la frecuencia de la señal de reloj. En tercer lugar, el módulo de ataque debe generar la señal de acuse de recibo. En esta situación, el maestro creerá que el proceso de comunicación se está realizando de forma correcta, y no volverá a repetir la orden. Por último, el módulo de ataque debe permitir la comunicación de la condición de detención con el fin de que todos los esclavos se preparen para un nuevo proceso de comunicación.

3.2 ATACANDO EL ROBOT EN MOVIMIENTO.

En este trabajo, el robot ha sido controlado mediante el uso de un algoritmo de seguimiento de la trayectoria muy bien conocido, cuya eficacia ha sido demostrada ampliamente: persecución pura (*pure pursuit*) [10]. Este algoritmo tiene en cuenta la configuración actual del robot, y determina la acción de bajo nivel correspondiente que hace que el robot siga una trayectoria deseada.

La configuración del robot es tal que utiliza dos motores como actuadores (conectados al mismo controlador), de tal forma que el algoritmo de control determina las velocidades de ambos. El controlador de bajo nivel (ver Fig. 1) recibirá estos valores y es el responsable de generar y enviar los comandos de velocidad a los motores. Por tanto, se utilizan dos dispositivos I2C (un maestro y un esclavo) con el fin de soportar la transmisión de comandos a los motores.

En esta etapa (la comunicación entre los controladores de alto y bajo nivel), los ataques hardware selectivos pueden ser implementados inyectando fallos en la señal del reloj del bus I2C. Concretamente, el objetivo de esta investigación es el estudio de los efectos que tiene poner en práctica ataques selectivos de hardware para evitar la comunicación entre el controlador y el motor de la rueda derecha, el de la rueda izquierda o de ambos motores. Como se muestra abajo, si estos ataques se ejecutan correctamente, el curso de robot puede ser modificado convenientemente. La principal limitación para el diseño de los ataques es que, ni el usuario, ni el controlador deben detectar que un ataque hardware se ha realizado, con la excepción del cambio del curso robot (identificándolo como un fallo puntual de la trayectoria). Como consecuencia, las siguientes condiciones deben ser tomadas en cuenta para la implementación del ataque:

1) Durante y después del ataque, el robot debe seguir navegando acorde a las instrucciones del controlador de alto nivel.

2) La consecuencia del ataque es una de estas opciones:

a) El robot no visitará un lugar específico.

b) El robot repetirá la visita de un área determinada, aumentando el tiempo en recorrer la trayectoria.

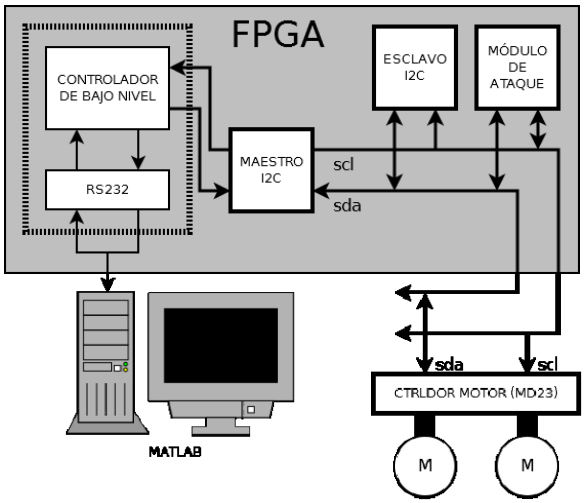
4 PLATAFORMA EXPERIMENTAL

El esquema de la plataforma experimental se muestra en la Fig. 3 a). La plataforma se divide en tres zonas diferentes: un PC que ejecuta Matlab para implementar el algoritmo de seguimiento de caminos; un dispositivo FPGA para implementar módulos de hardware (tales como el módulo de ataque) y un dispositivo I2C esclavo estándar (más concretamente el MD23 controlador de motor [8]). El dispositivo FPGA tiene la misión de implementar los módulos de hardware para obtener un mejor control de las señales. Los módulos implementados en el dispositivo FPGA son una UART basado en el protocolo RS232 (para comunicarse con Matlab en PC), un controlador de trayectoria (para adaptar las órdenes de Matlab al resto de los módulos), un maestro I2C (para controlar la comunicación a través del protocolo I2C), un esclavo I2C (para probar el sistema dentro de la FPGA) y el módulo de ataque (para poner en práctica los ataques a las comunicaciones I2C).

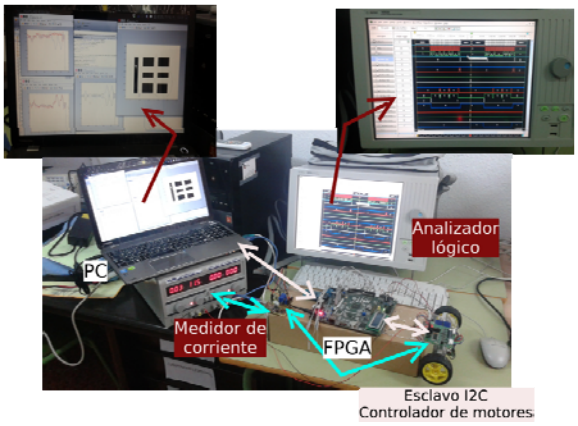
En la Figura 3 b) se muestra la plataforma real sobre la que se han realizado los experimentos. En ella se distinguen las tres partes básicas de la plataforma. Un PC conectado vía RS232 al dispositivo FPGA, el cual está conectado al controlador de motores (esclavo I2C) mediante las líneas del bus I2C. También se han incluido los elementos utilizados para realizar las mediciones. Estos elementos son el propio PC, un medidor de corriente para garantizar las medidas que devuelven los encoders, y un analizador lógico para monitorizar el bus I2C y las principales señales generadas en el dispositivo FPGA.

La operatividad para realizar los experimentos ha sido la siguiente. A partir de un escenario particular se ha definido una trayectoria que el robot ha de seguir con precisión. Dicha trayectoria es entregada al programa de control de alto nivel que corre en el entorno de Matlab. Este programa se ejecuta dentro de un bucle iterativo, en el que realiza la lectura de los encoders de los motores, aplica técnicas de

ISBN 978-84-15914-12-9



(a)



(b)

Figura 3: (a) Esquema de la plataforma experimental. (b) Fotografía de la plataforma experimental.

odometría para estimar la posición del robot y aplica el algoritmo *pure pursuit* para determinar las velocidades deseadas para las ruedas.

Esta información es enviada a la FPGA. El controlador de trayectorias le da el formato correcto (según los comandos de los motores) para que el maestro I2C pueda transmitirla en el bus. En el momento seleccionado, cuando el maestro trata de comunicar con el motor que se desea atacar, el módulo de ataque actúa sobre la señal de reloj, impidiendo tal comunicación. Posteriormente envía la señal de acuse de recibo. Debido a esto, el ataque consigue dos efectos:

- El módulo maestro piensa que no ha habido ningún problema y que el motor lo está obedeciendo.
- El módulo esclavo no percibe nada y mantiene el valor de velocidad que anteriormente le ha sido ordenado.

El posicionamiento del robot se ha obtenido únicamente mediante técnicas de odometría. No obstante, la inclusión de otros tipos de técnicas de posicionamiento (como GPS) no alteraría el comportamiento del sistema porque la lectura de los encoders (responsable de la odometría) no ha sido atacada. Luego, el controlador de alto nivel identifica que el robot no sigue la trayectoria correcta. Debido a que esta situación anómala es de carácter transitorio (no se produce en todos los puntos de la trayectoria ni en todas las trayectorias), el usuario no detectará una interferencia externa. El controlador de alto nivel siempre trata de corregir este comportamiento anómalo. No obstante, estas correcciones nunca llegan a los motores porque el canal de comunicación está intervenido. Un posible intento de corrección por parte del controlador de motores está descartado porque dicho controlador no identifica que se quieren comunicar con él.

Adicionalmente, la utilización de la técnica de GPS para obtener un posicionamiento más exacto del robot será considerado cuando se hagan las pruebas con el robot en un movimiento real. Actualmente, el robot no se mueve (aunque las ruedas sí lo hagan), debido a la cadena de instrumentación necesaria para obtener los resultados experimentales, como se aprecia en la Figura 3 b).

5 RESULTADOS EXPERIMENTALES

Diversos y diferentes experimentos se han realizado en varios escenarios. La idea ha sido encontrar el tipo de caminos y escenarios en los que los ataques de hardware temporales cumplen las condiciones definidas en el apartado 3. Debido a la falta de espacio, se presentarán sólo dos experimentos, los más significativos. El resultado más interesante fue encontrado en el escenario del experimento representado en la Fig 4.

En esta situación, el robot debe evolucionar

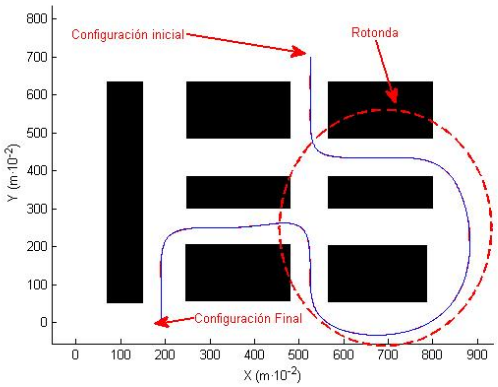


Figura 4: Experimento sin ataque, escenario y trayectoria del robot.

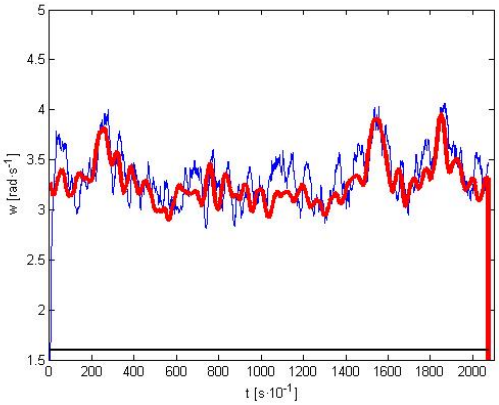
ISBN 978-84-15914-12-9

describiendo lo que puede ser llamado una trayectoria rotonda, en la que el robot se mueve alrededor de un área específica. En este experimento, la navegación se ha implementado sin ningún ataque. El algoritmo de seguimiento de caminos consigue controlar con éxito el movimiento del robot muy cerca de la trayectoria prevista. La trayectoria descrita por el robot se ha calculado mediante la aplicación de una metodología de estimación de posición, que utiliza la señal proporcionada por los encoders de los motores y aplica un Filtro de Kalman Extendido clásico (EFK).

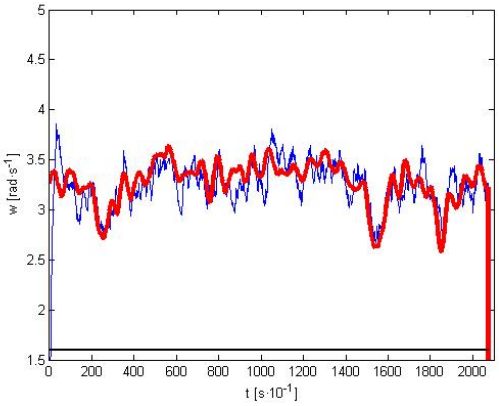
Las Fig. 5 a) y b) ilustran en rojo (con un trazo más grueso) la referencia proporcionada por el controlador de alto nivel y en azul (con un trazo más fino) la señal reconstruida mediante EFK a partir de la señal proporcionada por los encoders de los motores.

Obsérvese, cómo al no existir ningún ataque, la señal de los encoders sigue con bastante precisión la señal generada por el controlador de alto nivel.

En los siguientes experimentos, el ataque se llevó a cabo en los dos motores. En cada uno de ellos, se eligió un momento diferente para el ataque.



(a)



(b)

Figura 5: Experimento sin ataque: a) velocidad del motor derecho; b) velocidad del motor izquierdo.

En el primer experimento, el ataque del reloj tuvo lugar antes de la parte rotonda de la ruta, y desapareció en un momento en el que el robot estaba más cerca del final de la rotonda. Después de la finalización del ataque, el algoritmo de seguimiento hace que el robot siga el camino hasta el final de la rotonda (punto de la trayectoria más cercano), evitando que éste navegue a lo largo de la rotonda.

La Figura 6 a) muestra la ruta ejecutada por el robot, y las Figuras 6 b) y c) presentan respectivamente las referencias de velocidad y las señales reconstruidas de los encoders de ambos motores. Tenga en cuenta que, durante el ataque, ninguno de los motores siguen la señal del controlador de alto nivel.

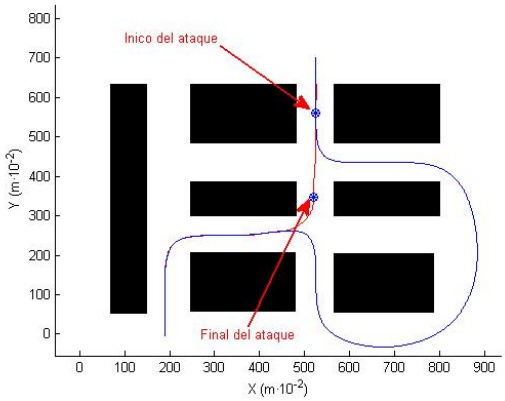
En el segundo experimento, el ataque tuvo lugar cerca del extremo de la parte rotonda de la ruta y desapareció en un momento en el que el robot está más cerca del comienzo de la misma. De esta manera, el algoritmo de seguimiento hace que el robot siga otra vez la rotonda, visitando esta zona de nuevo. La Fig. 7 ilustra la trayectoria del robot, las referencias de velocidad y las señales reconstruidas de los encoders de los motores.

6 CONCLUSIONES

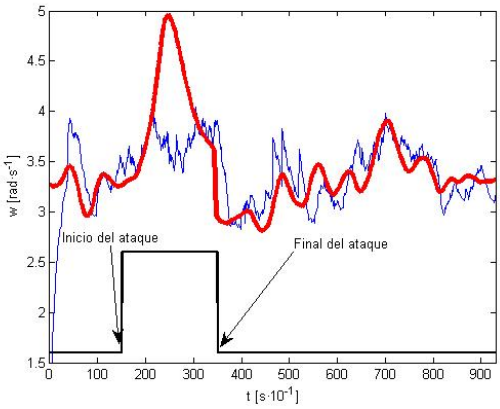
En este artículo se han descrito distintas formas de realizar ataques hardware sobre plataformas robóticas. Se ha prestado especial atención a describir el ataque por inyección de fallo en la señal de reloj del bus I2C. Este tipo de ataque ha sido utilizado en una plataforma experimental que ha permitido a los autores estudiar la influencia de dichos ataques en la navegación de un robot móvil que realiza una tarea de seguimiento de caminos. Concretamente, se han estudiado y caracterizado distintas circunstancias en las que el robot sigue navegando de forma segura, pero se le impide visitar una zona en concreto, o por el contrario se le obliga a visitar dos veces una misma zona. De esta forma, se comprueba como la vulnerabilidad del bus I2C deja abierta la puerta a la posible manipulación de la trayectoria seguida por el robot, sin que haya ninguna apariencia de intervención externa.

En este momento los autores están trabajando una metodología que permite la detección de ataques de reloj, lo que podría permitir establecer una reacción segura por parte del robot a los intento de ataques.

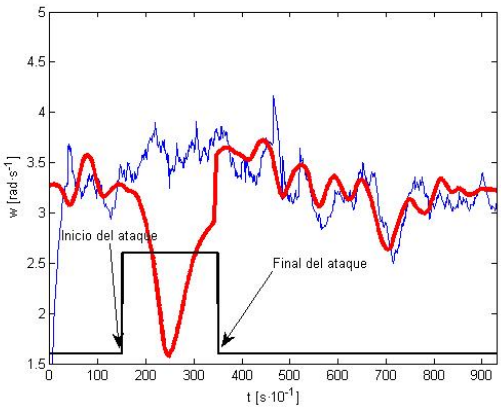
ISBN 978-84-15914-12-9



(a)

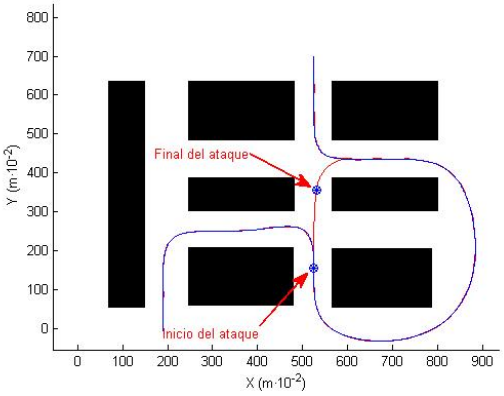


(b)

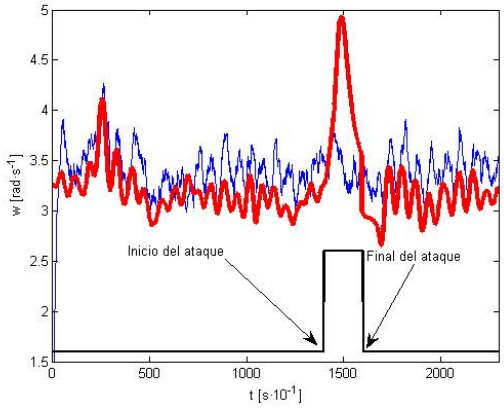


(c)

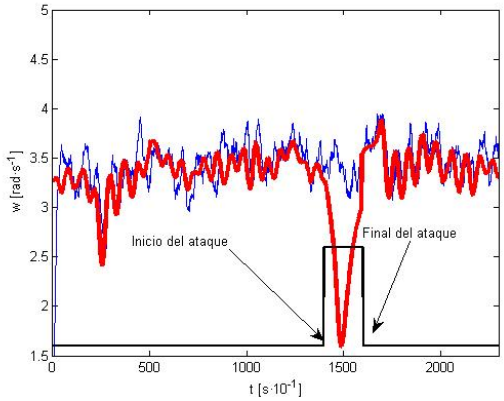
Figura 6: Experimento con ataque evitando la rotonda: a) trayectoria del robot; b) velocidad del motor derecho; c) velocidad del motor izquierdo.



(a)



(b)



(c)

Figura 7: Experimento con ataque repitiendo el trazado de la rotonda: a) trayectoria del robot; b) velocidad del motor derecho; c) velocidad del motor izquierdo.

Referencias

[1] Anderson, R., Kuhn, M., (1996) "Tamper Resistence – a Cautionary Note", 2nd USENIX Workshop on Electronic Commerce Proceeding, 1-11

[2] Bruschi, D., Cavallaro, L., Lanzi, A., (2005) "Replay Attack in TCG Specification and Solution", Proceedings of the 21st Annual Computer Security Applications Conference,, IEEE Computer Sociecity, 127–137

[3] Cuesta, F., Gómez-Bravo, F., & Ollero, A. (2004). Parking maneuvers of industrial-like electrical vehicles with and without trailer.

ISBN 978-84-15914-12-9

Industrial Electronics, IEEE Transactions on, 51(2), 257-269.

[4] Gómez-Bravo, F., Cuesta, F., & Ollero, A. (2001). Parallel and diagonal parking in non-holonomic autonomous vehicles. Engineering applications of artificial intelligence, 14(4), 419-434.

[5] Hamblen, J.O., van Bekkum, G.M.E., (2013) “An Embedded Systems Laboratory to Support Rapid Prototyping of Robotics and the Internet of Things”, Education, IEEE Transactions on, 56 (1), 121-128

[6] Huang, A., (2003) "Hacking the Xbox: An Introduction to Reverse Engineering," No Starch Press.

[7] Karaklajic, D, Verbauwhede, I., (2013) “Hardware Designer’s Guide to Fault Attacks”, IEEE Transactions on Very Large Scale Integration Systems, 21, 2295-2306

[8] Motor Controller MD23, (2014), http://www.robot-electronics.co.uk/htm/md23tech.htm

[9] Nobile, C., (2012) “Robots Vulnerable to Hacking”, http://www.roboticsbusinessreview.com/article/robots_vulnerable_to_hacking/

[10] Ollero, A., & Heredia, G. (1995, August). Stability analysis of mobile robot path tracking. In Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on (Vol. 3, pp. 461-466). IEEE.

[11] Sheppard, B., Thompson, T., (2014) “Cyber Security for Robots: Scenarios for 2030”, http://www.roboticsbusinessreview.com/article/cyber_security_for_robots_scenarios_for_2030

[12] Tehranipoor, M., Koushanfaar, F., (2010) “A Survey of Hardware Trojan Taxonomy and Detection”, IEEE Design and Test of Computers, 27(1), 10-25.