CEA Comité Español de Automática

XLV JORNADAS AUTOMÁTICA MÁLAGA 2024

UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Estudio de vulnerabilidades en el protocolo de comunicaciones Dynamixel aplicado en plataformas robóticas

Jiménez Naharro, R..[a,*], Garcia Jurado-Centurión, A.[b], Gómez Bravo, F.[a], López de Ahumada Gutiérrez, R.[a]

[a]*Dpto. de Ingeniería Electrónica, de Sistemas Informáticos y Automática, Universidad de Huelva, Avda. Fuerzas Armadas, s/n, 21007, Huelva, España.*
[b]*Dpto. de Tecnologías de la Información, Universidad de Huelva, Avda. Fuerzas Armadas, s/n, 21007, Huelva, España.*

To cite this article: Jiménez Naharro, R., Garcia Jurado-Centurión, A., Gómez Bravo, F., López de Ahumada Gutiérrez, R. 2024. Study of vulnerabilities in the Dynamixel communications protocol applied in robotic platformsl. Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10881

Resumen

La seguridad en las plataformas robóticas es una necesidad detectada por distintas instancias de alto nivel, como el Instituto Nacional de Ciberseguridad. En este contexto, el artículo presenta un estudio sobre las vulnerabilidades de un protocolo de comunicaciones utilizado en plataformas robóticas, el protocolo Dynamixel que soporta la transmisión de comandos de control utilizada por los actuadores digitales de esta marca. Dicho estudio ha revelado que el protocolo es vulnerable frente a ataques del tipo ”hombre de en medio”. Con el fin de abordar el estudio de esta vulnerabilidad, se han implementado transmisores de paquetes y módulos de ataque desarrollados mediante lenguaje VHDL, para su implantación en plataformas basadas en FPGA. Seguidamente, se han puesto de manifiesto las vulnerabilidades utilizadas por el atacante para llevar a cabo tal ataque, así como posibles estrategias de defensa que pueden ser implementadas en la arquitectura de la plataforma robótica.

Palabras clave: Implementación Digital, Protocolos de comunicación industrial, Fallos en sensores y actuadores, Seguridad en plataformas robóticas.

Study of vulnerabilities in the Dynamixel communications protocol applied to robotic platforms

Abstract

Security in robotic platforms is a need detected by high-level entities such as the National Cybersecurity Institute. In this context, the article presents a study on the vulnerabilities of a communication protocol used in robotic platforms, the Dynamixel protocol, that is used by digital actuators developed by this company. This study has revealed that the protocol is vulnerable to man-in-the-middle attacks. To this end, packet transmitters and attack modules implemented in VHDL have been developed for implementation on FPGA-based platforms. Next, the vulnerabilities used by the attacker to carry out such an attack have been revealed, as well as possible defense strategies that can be implemented in the robotic architecture.

Keywords: Digital implementation, Industrial communication protocols, Sensor and actuator faults, Robotic platform Security.

1. Introducción

En la actualidad, la utilización de plataformas robóticas está muy extendida en cualquier campo de aplicación, incluyendo áreas en las que la seguridad juega un papel primordial en el diseño del sistema. No obstante, en el mundo de la robótica, la seguridad no suele ser un factor incluido en la fase de diseño, sino como un añadido. En (INCIBE, 2020), el Instituto Nacional de Ciberseguridad, muestra los desafíos a los que se tiene que enfrentar la robótica industrial, indicando los siguientes campos:

- Gobierno
- Seguridad del software y del producto
- Identidad y acceso digital

*Autor para correspondencia: fernando.gomez@diesia.uhu.es

- Protección e identificación de los datos
- Operaciones de seguridad
- Gestión de vulnerabilidades
- Seguridad en las comunicaciones

Entre las actuaciones, dentro del campo de Operaciones de seguridad se encuentra: "*Implementar un hardware en los autómatas, que funcione como un cortafuegos para analizar constantemente las órdenes que el dispositivo recibe, admitiéndolas o denegándolas tras consultarlo con un tercer equipo que recoge todas reglas del autómata*". Por lo tanto, ya se cree en la necesidad de incluir defensas a bajo nivel, y no únicamente en el software de control utilizado.

La arquitectura básica de control de la plataforma robótica sobre la que se estudiarán las vulnerabilidades es la mostrada en la Figura 1 . En ella se identifican los bloques principales: controlador de alto nivel, controlador de bajo nivel, estimador de posición y robot. El controlador de alto nivel será el encargado de generar las órdenes de movimiento mediante algún algoritmo de planificación dependiendo de las órdenes de referencia y de la estimación de la posición real en la que se encuentre el robot en cada instante. En la mayoría de los casos, la implementación de este controlador suele ser software debido a la complejidad de los algoritmos utilizados. El controlador de bajo nivel será la pasarela que conecta las órdenes con los actuadores, y su misión principal consistirá en la traducción de dichas órdenes en los comandos de los actuadores utilizados en el robot. Por lo tanto, la implementación de este bloque dependerá tanto del formato de las órdenes de movimiento como de los actuadores utilizados. El bloque de estimación de posición recibirá la información de la sensorización del robot (como pueden ser los encoders de los motores) para poder estimar su posición actual.

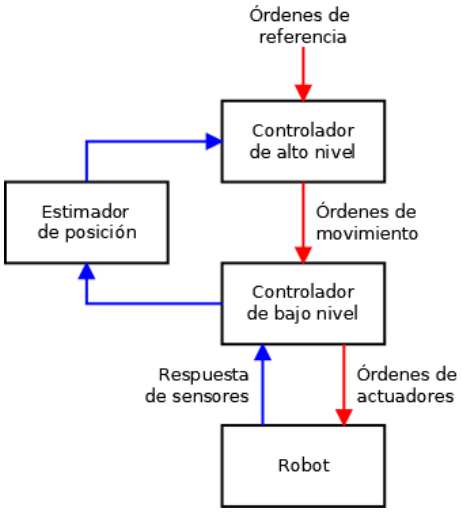

Figura 1: Arquitectura básica de control de una plataforma robótica.

La arquitectura de la Figura 1 también muestra los caminos de un sistema de control estándar. En rojo, se encuentra marcado el camino directo que va desde el controlador de alto nivel a los actuadores de la plataforma; mientras que en azul, se encuentra marcado el camino de realimentación que va desde la sensorización de la plataforma hasta el controlador de alto nivel. De este modo se encuentra el lazo cerrado de cualquier estrategia de control estándar con autocorrección.

Al tratar el tema de seguridad, hay que tener en cuenta que todas las partes del sistema pueden ser vulnerables, desde los bloques individuales hasta la comunicación entre ellos. En (INCIBE, 2020) se hace especial énfasis en la parte software y a los accesos a las sesiones. No obstante, existen otro tipo de vulnerabilidades que pueden ser explotadas mediante técnicas hardware, dichas técnicas reciben el nombre de ataques hardware. Estos ataques pueden ser clasificados en tres grandes grupos:

- Ataques físicos, que implican una manipulación directa sobre el sistema que se va a atacar. Un ejemplo de ataque físico serían los ataques por sondas cuya misión es determinar la estructura del sistema. Un posible ejemplo de objetivo de dicho ataque es la realización de un proceso de ingeniería inversa que permita acelerar las tareas de diseño del atacante.
- Ataques colatelares, que utilizan información colateral como fuente de vulnerabilidad a explotar (como pueden ser la radiación electromagnética, el consumo de potencia o la latencia del sistema). Un posible ejemplo de objetivo de dichos ataques es la adquisición de información sensible como claves.
- Ataques por inserción de fallos, que utilizan las propias especificaciones de los elementos del sistema como fuente de vulnerabilidad (como por ejemplo, la velocidad de operación, al hacer funcionar el sistema a una velocidad mayor que la indicada por el fabricante). Un posible ejemplo de objetivo de dichos ataques es la ejecución de determinadas operaciones a las que el perfil del usuario no debería tener acceso.

Ya existen estudios en la literatura sobre ataques a diferentes protocolos de comunicación usuales en la comunicación entre el controlador de bajo nivel y los actuadores. En (Jiménez-Naharro et al., 2017) se realiza el diseño de una defensa contra ataques por inserción de fallos en la señal de reloj del protocolo de comunicaciones I2C. En dicho estudio, el objetivo del ataque era forzar a un robot móvil a no cumplir determinadas órdenes, y por lo tanto, no seguir la trayectoria planificada. Para ello, se desactivaba la señal de reloj del protocolo de comunicaciones para aquellas órdenes que no debían ser realizadas.

En (Khelif et al., 2020) se plantea un ataque al bus de comunicaciones PCI Express para obtener información sensible y órdenes que se comunican a través de dicho bus cuando está conectado a una CPU, que será emulada mediante un dispositivo FPGPA.

Si se consideran los estudios que utilizan servomotores Dynamixel como plataformas de estudio, en (Potts and Ismail, 2024), se estudia la forma de detectar paquetes maliciosos generados por una orden maliciosa ejecuta sobre el PLC de control del sistema robótico.

En (Patel et al., 2024) se realiza un estudio sobre las vulnerabilidades de la plataforma TurtleBot3 de tal forma que el atacante se posiciona entre el controlador de alto nivel y la

plataforma robótica. Cabe destacar que el ataque estudiado se puede clasificar dentro de la categoría de hombre de en medio.

En (HOUCHOUAS et al., 2017) se estudia el efecto de la radiación electromagnética sobre un servomotor Dynamixel. En concreto, la radiación utilizada es un tren de pulsos de radio frecuencia, provocando un comportamiento erróneo e impredecible del servomotor.

2. Metodología

Las vulnerabilidades serán estudiadas sobre el entorno mostrado en la Figura 2. El ataque propuesto es del tipo hombre de en medio (*man in the middle*) (Cekerevac et al., 2017), en el que el módulo de ataque se coloca en medio del canal de comunicación. De esta forma, cualquier comunicación entre el emisor (el controlador de bajo nivel) y el receptor (los actuadores del robot) pueden ser monitorizados por el módulo de ataque. Los objetivos estándares de este tipo de ataques son la captura de información sensible que se transmite a través del bus de comunicaciones, y/o la intrusión de órdenes diferentes a las deseadas en el bus de comunicaciones.

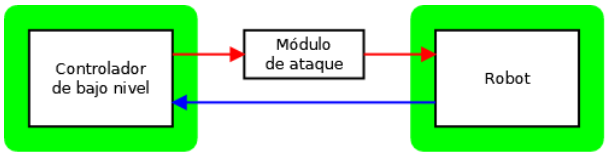

Figura 2: Arquitectura del ataque realizado sobre la plataforma robótica.

La arquitectura mostrada en la Figura 2 va a ser implementada teniendo en cuenta las siguientes consideraciones:

- Los actuadores pertenecerán a la familia Dynamixel. Se ha elegido dicha familia porque es de uso común en platafomas robóticas estándares; aunque no será determinante para el estudio.
- El controlador de bajo nivel tendrá dos casos de estudio. En primer lugar, se considerará que dicho controlador sea implementado por una plataforma Arduino UNO. En segundo lugar, se considerará que dicho controlador sea implementado por una plataforma basada en FPGA.
- El módulo de ataque será implementado en una plataforma basada en FPGA, para facilitar el prototipado. La utilización de este tipo de dispositivos (dispositivos lógicos de bajo nivel) viene motivado por su comportamiento no estándar. Esta situación impide el uso de plataformas estándares, pero no limita al uso de dispositivos FPGA. Su ubicación estará entre el controlador de bajo nivel y el primero de los servomotores, de tal forma que pueda modificar/cortar la comunicación cuando lo precise.

2.1. *Protocolo de comunicaciones Dynamixel*

El actuador utilizado es el servomotor AX-12A (Robotis, 2006). Debido a sus comandos puede ser utilizado en manipuladores (ya que permite el posicionado en un ángulo determinado) o en robótica móvil (ya que permite el movimiento del motor a una determinada velocidad); por lo que el estudio puede ser válido para cualquier plataforma robótica que utilice este tipo de actuadores.

El conexionado de dichos actuadores se muestra en la Figura 3. Como se puede apreciar, los actuadores están preparados para ser conectados en serie, siendo necesario un único controlador de bajo nivel para gobernar a varios actuadores. El conexionado está formado por tres hilos: una señal de dato, una línea de polarización y una línea de tierra. La capa física de la señal de dato es una conexión UART half duplex, permitiendo la comunicación de todos los elementos mediante un único cable. Dicha capa implica que la línea de dato se encuentra en alta impedancia mientras no se esté realizando ninguna comunicación.

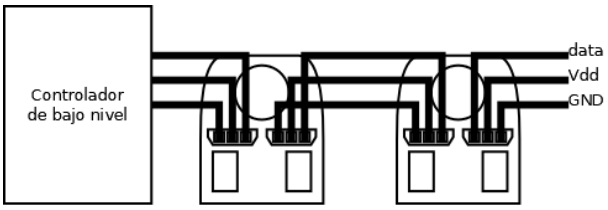

Figura 3: Conexionado de los servomotores AX-12A.

El protocolo de comunicaciones implica que el controlador manda un paquete de instrucción; y el actuador, incluido en la comunicación, envía un paquete de estado o respuesta después de un tiempo estipulado por lo que el controlador dispone de ese tiempo para liberar el bus de comunicaciones permitiendo la respuesta del actuador. La composición del paquete de instrucción, en su versión 1.0, se muestra en la Tabla 1, mientras que el paquete de respuesta sólo se diferencia en que el byte de instrucción es sustituido por un paquete de error para indicar el estado de la ejecución de la orden recibida.

Tabla 1: Estructura del paquete de instrucción en el protocolo Dynamixel (versión 1.0).

Byte	Comentario
Sinc	Sincronización (0xFF)
Sinc	Sincronización (0xFF)
Instr	Instrucción indicando el comando a realizar
Long	N. de bytes excepto los bytes anteriores
Par 0 - Par N	Parámetros relativos a la instrucción
Check sum	Tratamiento de errores en la comunicación

2.2. *Controlador de bajo nivel*

El controlador de bajo nivel va a ser implementado utilizando dos tecnologías diferentes. En primer lugar se utilizará una plataforma basada en Arduino UNO, como solución estándar utilizada ampliamente; además de sevir como comparador para la implementación en FPGA. En segundo lugar, se utilizará una plataforma basada en FPGA para poder realizar un diseño completamente a medida de la funcionalidad requerida.

La implementación sobre una plataforma Arduino UNO se muestra en la Figura 4. Cabe destacar que ha sido necesaria la utilización de un buffer triestado externo (74LS241) para implementar la conexión UART half-duplex. La comunicación entre la plataforma Arduino y el servomotor se ha realizado a

través de las librerías AX12A-servo-librery (Librería, 2024a) y DynamixelSerial (Librería, 2024b).

Para la implementación sobre una platafoma FPGA, se ha utilizado la placa de desarrollo Basys3 (Basys3, 2024). Para ello, se ha diseñado un sistema en VHDL formado por un generador de paquetes Dynamixel conectado a un UART half-duplex. El generador recibe la codificación de los comandos a transmitir, y genera el paquete para ser enviado por la UART. Cabe destacar que el dispositivo FPGA permite disponer de las salidas en alta impedancia, por lo que no es necesario la utilización del buffer triestado de la implementación en Arduino UNO. Dicha implementación se muestra en la Figura 5.

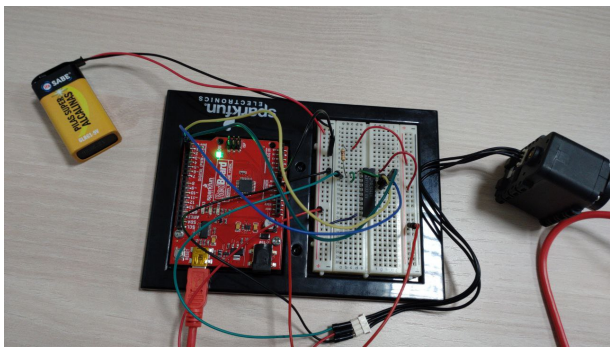

Figura 4: Conexionado de los servomotores AX-12A utilizando Arduino como controlador de bajo nivel.

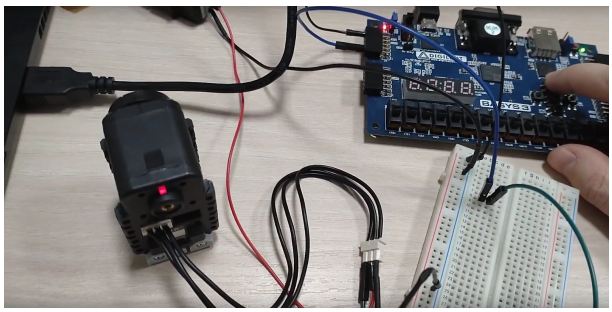

Figura 5: Conexionado de los servomotores AX-12A utilizando una placa de desarollo Basys3 como controlador de bajo nivel.

En la Figura 6 se muestran las formas de onda de la simulación correspondiente al controlador de bajo nivel implementado en la plataforma FPGA. El funcionamiento es el siguiente. Mediante las señales *select_com* (selector de instrucción), *on_off* (estado del led), *angle* (ángulo de giro) y *speed* (velocidad del motor), se construye el paquete que se desea transmitir. El comienzo de la transmisión se realiza con la activación de la señal *start*, y finaliza con la activación de la señal *lectura_completa*. En la forma de onda se han destacado los diferentes bytes que componen el paquete transmitido.

Por último, se han comparado los paquetes generados tanto por la versión basada en Arduino UNO como por la versión basada en la plataforma FPGA. Para ello, se han generado las mismas órdenes con las dos versiones, observándose las mismas formas de onda en el analizador lógico. Un ejemplo de ello se muestra en la Figura 7.

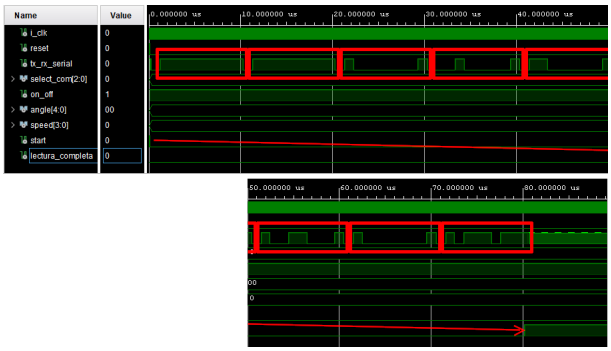

Figura 6: Formas de onda correspondiente al controlador de bajo nivel implementado en la placa de desarrollo Basys3.

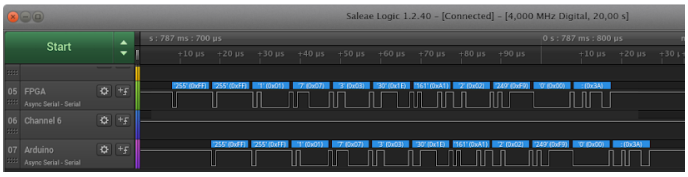

Figura 7: Comparación entre las formas de onda correspondiente al paquete generado por la versión basada en Arduino y la versión basada en la plataforma FPGA.

Se puede ver que los dos paquetes son iguales, siendo la única diferencia el instante en el que comienza la transmisión. Esto es debido a la plataforma basada en FPGA y la basada en Arduino necesitan tiempos diferentes para montar el paquete a transmitir. En concreto, la implementación en la plataforma basada en FPGA requiere menos tiempo para formar el paquete.

2.3. *Módulo de ataque*

La estructura de componentes del módulo de ataque se muestra en la Figura 8. Dicho módulo está formado por tres bloques, todos ellos diseñados en VHDL para que puedan ser implementados en una plataforma FPGA. En primer lugar, se dispone de una sniffer que monitoriza las comunicaciones transmitidas por el bus, de tal forma que las decodifica permitiendo conocer el servomotor al que va dirigida, la instrucción a realizar y los parámetros de dicha instrucción. Dicha información es enviada a un bloque de toma de decisión, cuya misión es identificar la situación que se desea atacar y la instrucción sustituta. En el caso de ataque, el bloque transmisor envía la instrucción indicada por el bloque de toma de decisión. En el caso de que no se realice el ataque, el bloque transmisor envía la instrucción proveniente del controlador de bajo nivel. A pesar de que únicamente se ha considerado el ataque en la dirección desde el controlador al servomotor, cabe destacar que con una pequeña modificación, se puede incluir la dirección desde el servomotor al controlador.

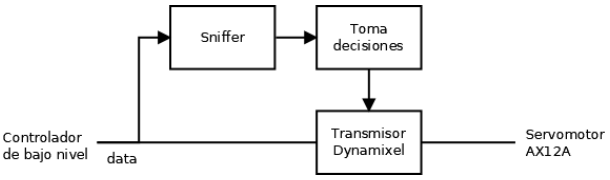

Figura 8: Estructura de componentes del módulo de ataque.

3. Casos de estudio

En la Figura 9 se muestra una fotografía de la plataforma de ataque utilizada en los casos de estudio. En dicha figura se puede distinguir la placa de desarrollo Basys3, que implementa el controlador de bajo nivel; la placa de desarrollo Arty7 (Arty7, 2024), que implementa el módulo de ataque; y el servomotor AX-12A, que será el objetivo del ataque. En el caso de que el controlador de bajo nivel sea implementado en una placa de Arduino, la figura sería la misma sustituyendo la placa Basys3 por la placa Arduino UNO.

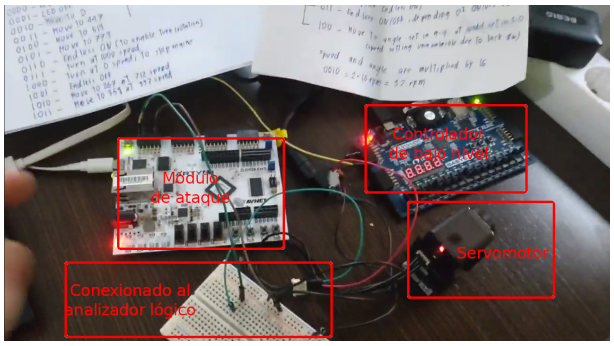

Figura 9: Fotografía de la plataforma de ataque considerando el controlador de bajo nivel implementado en una plataforma FPGA.

Cabe destacar que debido a la conexión modular de los servomotores (que se considera una ventaja para poder utilizar un control centralizado para todos los servomotores), la inclusión del módulo de ataque en el sistema ha sido muy simple.

La Figura 10 muestra la acción del ataque. implementado con la plataforma mostrada en la Figura 9. En primer lugar, el controlador de bajo nivel transmite un paquete de orden al servomotor (resaltado en color rojo). Al llegar al módulo de ataque, se decodifica el paquete, y se decide si se lleva a cabo el ataque. Después de la toma de decisión, el módulo de ataque transmite el nuevo paquete generado (resaltado en color azul). Una vez recibido por el servomotor, después de esperar el tiempo estipulado para responder, el servomotor envía el paquete de respuesta (resaltado en color verde).

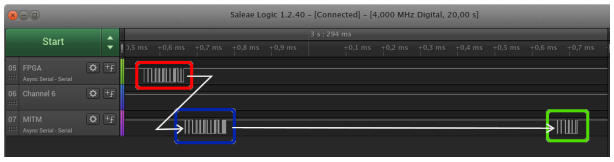

Figura 10: Formas de onda obtenidas del analizador lógico correspondiente al escenario 1.

A continuación se van a considerar tres escenarios utilizando la plataforma de ataque. El primer escenario implicará que el módulo de ataque sea transparente, es decir, el paquete transmitido por el módulo de ataque será el mismo que el recibido por el controlador de bajo nivel.

El segundo escenario implicará que el módulo de ataque modifique completamente la orden a ejecutar, independientemente de la orden transmitida por el controlador de bajo nivel. Cabe destacar que el paquete de respuesta indica que la orden se ha ejecutado de forma correcta, por lo que el controlador de bajo nivel no tiene ningún motivo para suponer que la orden ejecutada ha sido diferente a la enviada originalmente.

El tercer escenario implicará que el módulo de ataque modifique únicamente los parámetros de la orden a ejecutar. En la Figura 11 se muestra un zoom sobre el paquete original a ser transmitido y el paquete modificado por el módulo de ataque y finalmente transmitido. El byte original que se desea transmitir se ha destacado en rojo, mientras que el byte modificado por el módulo de ataque se ha destacado en azul. Evidentemente, también se ha modificado el último byte correspondiente al checksum del paquete a transmitir.

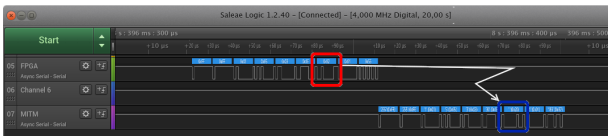

Figura 11: Zoom sobre el paquete original y atacado de una transmisión correspondiente al escenario 3.

En la Figura 12 se muestra el zoom sobre el paquete de respuesta del servomotor. Como se puede ver en el byte de error (cuarto byte), el servomotor no ha detectado nada anormal por lo que le indica al controlador de bajo nivel que todo ha salido de forma correcta. Por lo tanto, ni el controlador de bajo nivel ni el servomotor se han percatado de la existencia de ningún ataque.

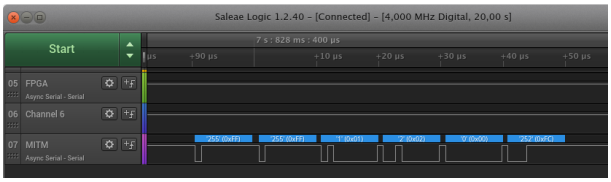

Figura 12: Zoom sobre el paquete de respuesta de una transmisión correspondiente al escenario 3.

4. Estudio de soluciones estándares y vulnerabilidades

Una vez demostrado que dicho ataque es factible, el siguiente paso será identificar las posibles vulnerabilidades detectadas, y realizar una discusión sobre la viabilidad de las estrategias de defensa estándares a este tipo de ataques. Para ello, hay que puntualizar que el objetivo de la defensa no es evitar el ataque, sino que el atacante no logre su objetivo.

En primer lugar, se va a considerar la modularidad del sistema, es decir, la facilidad para añadir un nuevo elemento más al bus de comunicaciones Dynamixel. Esta situación es tratada como una ventaja por permitir una ampliación sencilla de la plataforma, así como el intercambio sencillo de elementos defectuosos. No obstante, el atacante ha aprovechado esta ventaja para incluir el módulo de ataque, y de esta forma realizar los ataques indicados en el texto. Por lo tanto, dicha ventaja se ha convertido en vulnerabilidad que tiene difícil solución sin modificar la filosofía del protocolo de comunicaciones.

Adicionalmente, con un único módulo de ataque se pueden vulnerar todos los servomotores conectados en la misma línea debido al control centralizado. Frente a esta vulnerabilidad se puede actuar moviendo desde un control centralizado a un control descentralizado o semicentralizado, que obligue

a disponer de tantos módulos de ataque como líneas diferentes de comunicación. Dicho control descentralizado o semicentralizado podría tener la ventaja adicional del paralelismo de varios controladores de bajo nivel acelerando la velocidad del sistema siempre y cuando se mantenga un sincronización aceptable.

Con respecto a las estrategias de defensa estándar, la estrategia básica frente a cualquier ataque a las comunicaciones, y por tanto, al ataque del hombre de en medio, es la encriptación de la información. De esta forma se oculta la información sensible que es transmitida por el canal expuesto. En nuestro caso particular, no existe información sensible propiamente dicha, puesto que las órdenes que puede recibir el servomotor son públicas. El único caso en el que tendría sentido la encriptación de la orden transmitida sería dificultar la tarea decodificar la orden para poder modificarla. Esta situación sólo tendría ventaja en el escenario número 3, único escenario en el que la acción del módulo de ataque está relacionada con la orden original transmitida. Como los servomotores no están preparados para trabajar con la información encriptada, se debería añadir un módulo adicional para incluir la desencriptación. No obstante, la inclusión de este nuevo módulo puede provocar un retraso adicional que debe ser tenido en cuenta. En algunas aplicaciones, sobre todo en aplicaciones de tiempo real, este retraso añadido puede llegar a ser prohibitivo.

También cabe destacar que la encriptación de la información no realiza ningún proceso de detección de ataque, por lo que si se consigue desencriptar la información, dicha encriptación no produce ninguna utilidad, y el controlador de bajo nivel no es avisado de tal circunstancia.

Otra estrategia de defensa estándar frente al ataque del hombre de en medio es evitar la repetición de transmisiones. De esta forma, el receptor (en nuestro caso puede ser tanto el servomotor como el controlador de bajo nivel en función de la dirección de la comunicación) podría saber que la instrucción recibida no es válida. Como en el caso anterior, la aplicación de esta estrategia implicaría añadir una capa al servomotor porque no sería extraño que el servomotor recibiera órdenes idénticas a otras órdenes recibidas con anterioridad. Al igual que en el caso anterior, la viabilidad de incluir un nivel adicional debe ser estudiada para la aplicación considerada.

5. Conclusiones

En el artículo se ha estudiado la viabilidad de realizar un ataque del hombre de en medio a una plataforma robótica que utilice servomotores digitales Dynamixel (más concretamente el modelo AX-12A), considerando tres estrategias diferentes. Se ha demostrado que este tipo de ataques es viable, pudiendo llegar a ser peligrosos en el caso de que no sean detectados.

A continuación se han estudiados las vulnerabilidades analizadas en el estudio, estando basadas en el control centralizado de todos los servomotores, y en su capacidad de ser configurada como una plataforma modular (con el fin de añadir y/o sustituir elementos). Dicha vulnerabilidad puede ser utilizada por un atacante para incluir un módulo de ataque desde el que se produzcan los mismos. Por lo tanto, sería conveniente el estudio de soluciones descentralizadas o semicentralizadas para minimizar los riesgos.

Finalmente, se han analizado las estrategias de defensa estándares para este tipo de ataques con el fin de estudiar su aplicación a la plataforma considerada. De ese análisis se se ha concluido que dichas estrategias no garantizan la defensa, y que en su caso, requieren la inclusión de módulos adicionales para su implementación. La inclusión de dichos módulos podrían ser prohibitivos en determinadas aplicaciones debido a la existencia de retrasos adicionales. Cabe destacar que dichas estrategias sólo monitorizan un sentido de la comunicación, por lo que la detección (en caso de realizarse) sólo se lleva a cabo en el receptor.

Como trabajo futuro, se debe plantear el estudio y diseño de los módulos que eviten la repetición de paquetes. Otra línea a plantear es el estudio de estrategias de defensa en las que la detección también sea realizada en el emisor, y no sólo en el receptor.

Referencias

Arty7, 2024. Arty a7 reference manual.
URL: https://digilent.com/reference/programmable-logic/arty-a7/reference-manual

Basys3, 2024. Basys 3 reference manual.
URL: https://digilent.com/reference/programmable-logic/basys-3/reference-manual

Cekerevac, Z., Dvorak, Z., Prigoda, L., Cekerevac, P., 2017. Internet of things and the man-in-the-middle attacks–security and economic risks. MEST Journal 5 (2), 15–25.
DOI: 10.12709/mest.05.05.02.03

HOUCHOUAS, V., ESTEVES, J. L., COTTAIS, E., KASMI, C., ARMSTRONG, K., 2017. Immunity assessment of a servomotor exposed to an intentional train of rf pulses. 2017 International Symposium on Electromagnetic Compatibilit.

INCIBE, 2020. Los ciberdesafíos de la seguridad en la robótica industrial.
URL: https://www.incibe.es/incibe-cert/blog/los-ciberdesafios-seguridad-robotica-industrial

Jiménez-Naharro, R., Gómez-Bravo, F., Medina-García, J., Sánchez-Raya, M., Gómez-Galán, J., 2017. A smart sensor for defending against clock glitching attacks on the i2c protocol in robotic applications. Sensors 677, 1–17.
DOI: 10.3390/s17040677

Khelif, M. A., Lorandel, J., Romain, O., Regnery, M., Baheux, D., Barbu, G., 2020. Toward a hardware man-in-the-middle attack on pcie bus. Microprocessors and Microsystems 77, 103198.
DOI: 10.1016/j.micpro.2020.103198

Librería, 2024a. Ax-12a-servo-library.
URL: https://github.com/jumejume1/AX-12A-servo-library

Librería, 2024b. Dynamixelserial-library.
URL: https://github.com/strawlab/laser_pan_tilt/blob/master/arduino/DynamixelSerial/DynamixelSerial.h

Patel, Y., H, P., Rughani, Maiti, T. K., 2024. An examination of the security architecture and vulnerability exploitation of the turtlebot3 robotic system. International Journal of Computing and Digital Systems.

Potts, J., Ismail, M., 2024. Hybrid cyber-physical intrusion detection system for smart manufacturing. The International FLAIRS Conference Proceedings.

Robotis, 2006. User's manual dynamixel ax-12.
URL: https://www.generationrobots.com/media/Dynamixel-AX-12-user-manual.pdf