MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

PRUEBAS DE INTRUSIÓN EN AUTOMÓVILES MEDIANTE ATAQUES DE RADIOFRECUENCIA. ANÁLISIS DE VULNERABILIDADES

Autor: Roberto Gesteira Miñarro

Director: Rafael Palacios Hielscher

Director: Gregorio López López

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

PRUEBAS DE INTRUSIÓN EN AUTOMÓVILES MEDIANTE ATAQUES DE RADIOFRECUENCIA. ANÁLISIS DE VULNERABILIDADES

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2021/22 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

GESTEIRA MIÑARRO ROBERTO - 48228840W

Firmado digitalmente por GESTEIRA MIÑARRO ROBERTO - 48228840W

Fdo.: Roberto Gesteira Miñarro Fecha: 11/07/2020

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

PALACIOS HIELSCHER RAFAEL - 00817103M

Digitally signed by PALACIOS HIELSCHER RAFAEL - 00817103M
Date: 2022.07.10 20:53:50 +02'00'

Fdo.: Rafael Palacios Hielscher Fecha: 11/07/2020

LOPEZ LOPEZ GREGORIO IGNACIO - 75887558R

Digitally signed by LOPEZ LOPEZ GREGORIO IGNACIO - 75887558R
Date: 2022.07.10 14:23:11 +02'00'

Fdo.: Gregorio López López Fecha: 11/07/2020

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

PRUEBAS DE INTRUSIÓN EN AUTOMÓVILES MEDIANTE ATAQUES DE RADIOFRECUENCIA. ANÁLISIS DE VULNERABILIDADES

Autor: Roberto Gesteira Miñarro

Director: Rafael Palacios Hielscher

Director: Gregorio López López

Madrid

Agradecimientos

A mi novia y a mis familiares, amigos y profesores, por ayudarme a ser como soy.

PRUEBAS DE INTRUSIÓN EN AUTOMÓVILES MEDIANTE ATAQUES DE RADIOFRECUENCIA. ANÁLISIS DE VULNERABILIDADES

Autor: Gesteira Miñarro, Roberto.
Director: Palacios Hielscher, Rafael.
Director: López López, Gregorio.
Entidad Colaboradora: CESVIMAP – MAPFRE.

RESUMEN DEL PROYECTO

En este proyecto se presenta una metodología de análisis de comunicaciones entre vehículos y llaves mediante canales de radiofrecuencia que permita evaluar el nivel de exposición y las vulnerabilidades de estos sistemas. Para poner en práctica esta metodología, se ha desarrollado una herramienta para capturar señales del mando de un cierto vehículo y extraer su contenido binario para posterior análisis.

Palabras clave: Radiofrecuencia, ciberseguridad, vehículos.

1. Introducción

Los automóviles son sistemas complejos cuyo funcionamiento está basado en millones de líneas de código y emplean una gran cantidad de tecnologías de comunicación, así como dispositivos y aplicaciones de terceros. Por tanto, la ciberseguridad es un tema especialmente relevante en el sector del automóvil; y lo será todavía más en el futuro, debido a la regulación que entrará en vigor [1].

2. Definición del proyecto

Hoy en día, las compañías de seguros aparecen como un actor clave en el sector del automóvil, ya que necesitan estimar los riesgos de ciberseguridad de los vehículos para reflejarlo en las pólizas. El proyecto se ha realizado en colaboración con CESVIMAP, una empresa de MAPFRE dedicada a I+D+i.

En el ámbito de la ciberseguridad se pretende definir una metodología que ofrezca la posibilidad de evaluar y clasificar los vehículos dependiendo de su nivel de exposición y protección frente a ataques que permitan: (i) robar el vehículo; (ii) robar datos contenidos en el vehículo; (iii) secuestrar el vehículo; o (iv) controlarlo de manera remota.

En este TFM se realiza un análisis del estado del arte, documentando los ataques de RF existentes en mandos tradicionales y llaves *keyless* [2], [3]; se define una metodología de análisis; y se desarrolla una herramienta de ataque semi-automática.

3. Descripción del sistema

La metodología desarrollada permite realizar capturas de señales del mando de un coche en cuestión mediante HackRF One y GNU Radio Companion y realizar un análisis exhaustivo mediante `inspectrum` para obtener el contenido binario de la señal. Una vez se tienen varias muestras de estas señales, es posible identificar la estructura de la trama y configurar la herramienta desarrollada: `rf_attack.py`.

La herramienta desarrollada en este proyecto está escrita en Python y utiliza el dispositivo YARD Stick One y `rfcat`. Se trata de un programa CLI que permite recibir y transmitir señales a partir de una configuración de parámetros de radiofrecuencia. Con `rf_attack.py` es posible realizar múltiples capturas de un mando y analizar su contenido en binario, descomponiendo la trama en los distintos campos identificados (por ejemplo: secuencia de sincronismo, comando, identificador de llave o código evolutivo). Además, es posible realizar pruebas de concepto de ataques documentados, pero con la diferencia de que el ataque se realiza en digital, a partir de un contenido binario determinado.

4. Resultados

La metodología descrita anteriormente se ha puesto en práctica con mandos que utilizan un sistema de códigos evolutivos (*rolling codes*) en distintos vehículos. Un resumen del análisis de cada señal se muestra en la Tabla 1.

Tabla 1. Características de señales capturadas

Fabricante	Modulación	Longitud
Marca A	2FSK	Una trama de 168 bits
Marca B	2FSK	3 tramas de 120 bits
Marca C	ASK/OOK	Una trama de 312 bits
Marca D	ASK/OOK	6 tramas de 312 bits y 27 tramas de 32 bits

Los fabricantes analizados utilizan modulaciones digitales sencillas como 2FSK (Fig. 1) o ASK/OOK (Fig. 2) para transmitir la información. Además, en todos los casos analizados se utiliza codificación Manchester, en la que la información se codifica por flanco (flanco de subida: 0 lógico y flanco de bajada: 1 lógico, o viceversa según el convenio), proporcionando así sincronismo a nivel de símbolo.

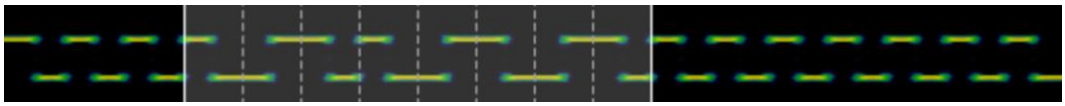

Fig. 1. Muestra de señal capturada para un vehículo de la Marca A

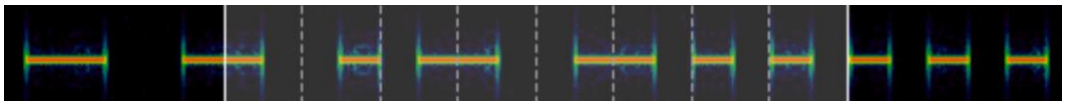

Fig. 2. Muestra de señal capturada para un vehículo de la Marca D

Todas las señales cuentan con una secuencia de sincronismo al comienzo de la trama (Fig. 3) que consiste en una sucesión de 1 lógicos hasta llegar a un 0 lógico, que indica el comienzo de los datos transmitidos.

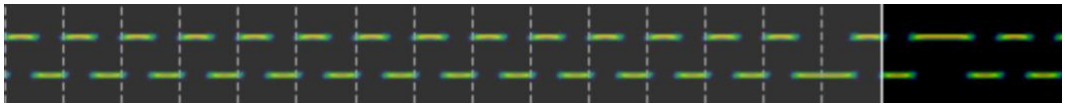

Fig. 3. Secuencia de sincronismo de señal capturada para un vehículo de la Marca A

Usando la metodología descrita y la herramienta desarrollada, fue posible realizar un análisis exhaustivo sobre la comunicación entre un vehículo de la Marca A y el mando correspondiente. Estos son algunos ejemplos de la información transmitida por el mando del vehículo cuando se pulsa el botón de apertura (en hexadecimal):

- `ffffffffffffe7b5fee66491d0592c722769755eb`.
- `ffffffffffffe7b9fee66491db9af74d04ba1ae4e`.
- `ffffffffffffe7b1fee66491d34a91610d4848854`.

Hay varios dígitos que coinciden en las muestras anteriores, lo cual indica que la información está dividida en diferentes secciones binarias. Tomando la primera muestra, se pueden identificar los siguientes campos:

- Secuencia de sincronismo: `ffffffffffffe`.
- Acción: `7b`.
- Valor cambiante: `5f`.
- Valor fijo: `ee66491d`.
- Código evolutivo: `0592c722769755eb`.

La acción es `7b` para abrir el vehículo, `3b` para cerrarlo y `5d` para abrir el maletero. Se desconoce la función del valor cambiante, aunque podría tratarse de algún método de detección de errores. El valor fijo se corresponde con un identificador de la llave. Así, solo las llaves configuradas para un vehículo funcionarán. Por último, el código evolutivo es un número pseudo-aleatorio de 64 bits.

Cabe mencionar que se ha realizado un análisis de vehículos con sistema *keyless*, tanto a nivel de comunicaciones de radiofrecuencia como a nivel de *hardware*.

5. Conclusiones

Una vez finalizado el proyecto, y en base a los resultados obtenidos, se ha conseguido definir una metodología de análisis para comunicaciones de RF y se ha puesto en práctica para analizar la seguridad del protocolo usado por un cierto vehículo de Marca A, resaltando la importancia de trabajar con las señales en digital, comprendiendo la información transmitida.

Adicionalmente, parte del proyecto se publicó en un *paper* para las VII Jornadas Nacionales de Investigación en Ciberseguridad (JNIC), accesible en [4].

6. Referencias

[1] U. Mezcua, «Los coches deberán tener certificado de ciberseguridad para venderse a partir de 2022,» 12 de ago. de 2020. [En línea]. Disponible en: https://www.abc.es/motor/reportajes/abci-coches-deberan-tener-certificado-ciberseguridad-para-venderse-partir-2022-202008120103_noticia.html (Accedido: 31-05-2022).

[2] O. Ibrahim, A. Hussain, G. Oligeri y R. Pietro, «Key is in the Air: Hacking Remote Keyless Entry Systems,» en. 10 de ene. de 2019, págs. 125-132, ISBN: 978-3-030-16874-2. DOI: 10.1007/978-3-030-16874-2_9.

[3] L. Wouters, B. Gierlichs y B. Preneel, «My other car is your car: compromising the Tesla Model X keyless entry system,» *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, nº 4, págs. 149-172, ago. de 2021. DOI: 10.46586/tches.v2021.i4.149-172. [En línea]. Disponible en: https://tches.iacr.org/index.php/TCHES/article/view/9063 (Accedido: 28-05-2022).

[4] R. G. Miñarro *et al.*, «Metodología y herramientas para análisis y evaluación de seguridad frente a ataques de radiofrecuencia en vehículos,» *Investigación en Ciberseguridad. Actas de las VII Jornadas Nacionales (JNIC 2022)*, págs. 167-171, jun. de 2022. Disponible en: https://2022.jnic.es/Actas_JNIC_2022_v11.pdf (Accedido: 30-06-2022).

PENETRATION TESTING IN CARS USING RADIOFREQUENCY ATTACKS.
VULNERABILITY ANALYSIS

Author: Gesteira Miñarro, Roberto.
Supervisor: Palacios Hielscher, Rafael.
Supervisor: López López, Gregorio.
Colaborating Entity: CESVIMAP – MAPFRE.

ABSTRACT

This project presents a methodology for analyzing communications between vehicles and keys through radiofrequency channels that allow evaluating the level of exposure and vulnerabilities of these systems. To put this methodology into practice, a tool has been developed for capturing remote signals from a certain vehicle to extract its binary content for later analysis.

Keywords: Radiofrequency, cybersecurity, vehicles.

1. Introduction

Cars are complex systems whose operation is based on millions of lines of code and employ many communication technologies, as well as third-party devices and applications. Therefore, cybersecurity is a particularly relevant issue in the automotive sector; and it will be even more so in the future, due to the regulation that will come into force [1].

2. Project definition

Nowadays, insurance companies appear as a key player in the automobile sector since they need to estimate the cybersecurity risks of vehicles to reflect it in policies. The project has been carried out in collaboration with CESVIMAP, a MAPFRE company dedicated to R&D&i.

In the field of cybersecurity, the aim is to define a methodology that offers the possibility of evaluating and classifying vehicles depending on their level of exposure and protection against attacks that allow: (i) theft of the vehicle; (ii) steal data contained in the vehicle; (iii) hijack the vehicle; or (iv) control it remotely.

In this Master's Thesis an analysis of the state of the art is carried out, documenting the existing RF attacks on traditional remote controls and keyless systems [2], [3]; an analysis methodology is defined; and a semi-automatic attack tool is developed.

3. System description

The developed methodology allows to capture signals from a certain car remote control using HackRF One and GNU Radio Companion and to perform an exhaustive analysis using `inspectrum` to obtain the binary content of the signal. Once several samples of these signals have been analyzed, it is possible to identify the structure of the frame and configure the developed tool: `rf_attack.py`.

The tool developed in this project is written in Python and uses a YARD Stick One device and `rfcat`. It is a CLI program that allows receiving and transmitting signals

given a configuration of radiofrequency parameters. With `rf_attack.py` it is possible to make multiple captures of a remote and analyze their content in binary, breaking down the frame into the different identified fields (for instance: synchronism sequence, command, key identifier or rolling code). In addition, it is possible to carry out proofs of concept of documented attacks, but with the difference that the attack is performed in digital format, from a specific binary content.

4. Results

The methodology described above has been put into practice with controls that use a system of rolling codes in different vehicles. A summary of the analysis of each signal is shown in Table 1.

Table 1. Characteristics of captured signals

Manufacturer	Modulation	Length
Brand A	2FSK	Single 168-bit frame
Brand B	2FSK	3 frames of 120 bits
Brand C	ASK/OOK	Single 312-bit frame
Brand D	ASK/OOK	6 frames of 312 bits and 27 frames of 32 bits

The analyzed manufacturers use simple digital modulations such as 2FSK (Fig. 1) or ASK/OOK (Fig. 2) to transmit the information. In addition, Manchester encoding is used in all the cases analyzed, in which the information is encoded by edge (rising edge: logic 0 and falling edge: logic 1, or vice versa according to the convention), thus providing synchronism at symbol level.

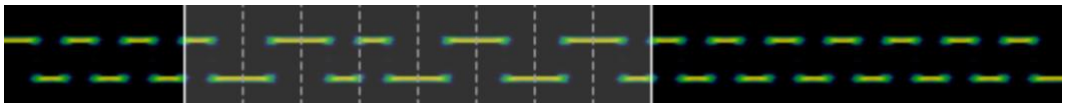

Fig. 1. Sample of a captured signal for a Brand A vehicle

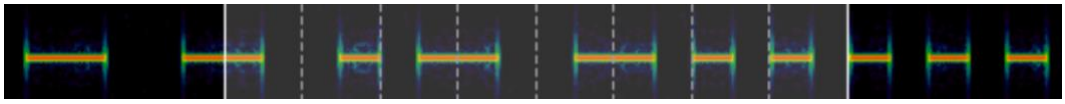

Fig. 2. Sample of a captured signal for a Brand D vehicle

All signals have a synchronism sequence at the beginning of the frame (Fig. 3) consisting of a series of logical 1's until reaching a logical 0, which indicates the beginning of the transmitted data.

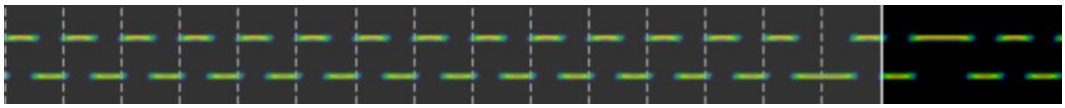

Fig. 3. Synchronism sequence of a captured signal for a Brand A vehicle

Using the described methodology and the developed tool, it was possible to perform an exhaustive analysis on the communication between a Brand A vehicle and the corresponding remote control. Here are some examples of the information transmitted by the vehicle's remote when the unlock button is pressed (in hexadecimal):

- `ffffffffffffe7b5fee66491d0592c722769755eb`.
- `ffffffffffffe7b9fee66491db9af74d04ba1ae4e`.
- `ffffffffffffe7b1fee66491d34a91610d4848854`.

There are several matching digits in the above samples, indicating that the information is structured into different binary sections. Taking the first sample, the following fields can be identified:

- Synchronism sequence: `ffffffffffffe`.
- Command: `7b`.
- Variable value: `5f`.
- Fix value: `ee66491d`.
- Rolling code: `0592c722769755eb`.

The action is `7b` to open the vehicle, `3b` to close it and `5d` to open the trunk. The function of the variable value is unknown, although it might be some method of error detection. The fix value corresponds to a key identifier. Thus, only the keys configured for a vehicle work. Finally, the rolling code is a 64-bit pseudo-random number.

It is worth mentioning that an analysis of vehicles with keyless system has been carried out, both at the level of radiofrequency communications and hardware level.

5. Conclusions

Once the project is finished, and based on the results obtained, an analysis methodology for RF communications has been defined and has been put into practice to analyze the security of the protocol used by a certain Brand A vehicle, highlighting the importance of working with digital signals, understanding the transmitted information.

Additionally, part of the project was published in a paper for *VII Jornadas Nacionales de Investigación en Ciberseguridad* (JNIC), accessible at [4].

6. References

[1] U. Mezcua, "Los coches deberán tener certificado de ciberseguridad para venderse a partir de 2022," Aug. 12, 2020. [Online]. Available: https://www.abc.es/motor/reportajes/abci-coches-deberan-tener-certificado-ciberseguridad-para-venderse-partir-2022-202008120103_noticia.html (Visited on 31-05-2020).

[2] O. Ibrahim, A. Hussain, G. Oligeri and R. Pietro, "Key is in the Air: Hacking Remote Keyless Entry Systems," in. Jan. 10, 2019, pp. 125-132, ISBN: 978-3-030-16874-2. DOI: 10.1007/978-3-030-16874-2_9.

[3] L. Wouters, B. Gierlichs and B. Preneel, "My other car is your car: compromising the Tesla Model X keyless entry system," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 4, pp. 149-172, Aug. 2021. DOI: 10.46586/tches.v2021.i4.149-172. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9063 (Visited on 28-05-2022).

[4] R. G. Miñarro *et al.*, "Metodología y herramientas para análisis y evaluación de seguridad frente a ataques de radiofrecuencia en vehículos," *Investigación en Ciberseguridad. Actas de las VII Jornadas Nacionales (JNIC 2022)*, pp. 167–171, Jun. 2022. [Online]. Available: https://2022.jnic.es/Actas_JNIC_2022_v11.pdf (Visited on 30-06-2022).

Índice de la memoria

Índice de figuras

Índice de tablas

Índice de códigos

Capítulo 1

Introducción

Los automóviles son sistemas complejos cuyo funcionamiento está basado en millones de líneas de código y emplean una gran cantidad de tecnologías de comunicación, así como dispositivos y aplicaciones de terceros. Por tanto, la ciberseguridad es un tema especialmente relevante a día de hoy en el sector del automóvil; y lo será todavía más en los próximos años, debido a la regulación que entrará en vigor al respecto.

Ante esta situación, es necesario disponer de metodologías y herramientas que permitan evaluar y clasificar los vehículos dependiendo de su nivel de exposición y protección frente a ataques de ciberseguridad. En este Trabajo Fin de Máster se presenta una metodología y un conjunto de herramientas que permiten analizar, evaluar y comparar los mensajes intercambiados entre mandos y vehículos por medio de canales de radiofrecuencia, además de poder realizar pruebas de concepto para replicar ataques documentados en la literatura.

La ciberseguridad representa actualmente un tema significativo en la industria del automóvil que ha comenzado a regularse recientemente (especialmente, desde 2019). Teniendo en cuenta que una regulación de ciberseguridad similar en otros sectores como la banca ha llevado más de una década, esto representa un desafío para la industria del automóvil.

Dado que en todos los países desarrollados es obligatorio disponer de un seguro asociado a cada vehículo, las compañías de seguros aparecen como un actor clave en este contexto, ya que necesitan estimar los riesgos de ciberseguridad de los vehículos para reflejarlo adecuadamente en las pólizas.

CAPÍTULO 1. INTRODUCCIÓN

El presente Trabajo Fin de Máster se ha realizado en colaboración con CESVIMAP (empresa del grupo MAPFRE dedicada a I+D+i en el sector del automóvil) en el marco del proyecto de investigación TIBET (inTegración de cIberseguridad y Biomecánica en vEhículos y Tráfico).

Concretamente, es necesario establecer una serie de metodologías rigurosas que permitan clasificar y evaluar los vehículos dependiendo de su nivel de exposición y protección frente a ataques que permitan: (i) robar el vehículo; (ii) robar datos contenidos en el vehículo; (iii) secuestrar el vehículo; o (iv) conseguir controlarlo de manera remota.

Este Trabajo Fin de Máster se centra en ataques de radiofrecuencia (RF). La mayoría de trabajos que se han llevado a cabo anteriormente en este campo se han centrado en intentar abrir o robar el vehículo, mediante señales en formato analógico. El objetivo de este TFM, en cambio, es definir una metodología que permita conocer y evaluar el funcionamiento de los mecanismos de apertura y arranque por RF de diferentes marcas de vehículos desde el punto de vista de la ciberseguridad. Para conseguir dicho objetivo, es fundamental trabajar con estas señales en formato digital, por lo que se necesita desarrollar una serie de herramientas que permitan obtener, analizar y reproducir dichas señales digitales.

En este documento, se desarrolla en primer lugar el estado del arte en relación con ataques y metodologías documentadas. Posteriormente, se explican la definición del trabajo realizado y los recursos y herramientas utilizadas. Finalmente, se explica en detalle el sistema desarrollado junto con un análisis de los resultados obtenidos, conclusiones y posibles líneas de investigación adicionales.

Como añadido, se incluyen una guía de instalación y un manual de usuario como anexos. Además, se incluye un tercer anexo con una breve reflexión sobre la importancia de los Objetivos de Desarrollo Sostenible (ODS) y su relación con el proyecto de investigación realizado.

Capítulo 2

Estado del arte

Actualmente, la mayoría de los automóviles en circulación utilizan canales de radiofrecuencia para comunicarse con otros dispositivos. Estos canales pueden presentar problemas de seguridad, ya que la comunicación viaja en forma de ondas que se propagan por el aire, de manera que un atacante puede interceptar dichas ondas.

En el caso de los vehículos, la gran mayoría ofrecen la posibilidad de abrir las puertas mediante un mando a distancia. Para ello, el mando emite una señal y el vehículo la recibe. Si esta señal es válida, entonces el vehículo se abre.

En automóviles más modernos, existen otro tipo de llaves que funcionan por proximidad (tienen un menor rango de alcance). El protocolo de comunicación con este tipo de llaves (conocidas como *keyless*) es más complejo en el sentido de que puede ser bidireccional y puede depender de otros canales de comunicación.

2.1. Ataques al mando a distancia

Antiguamente, el mando a distancia enviaba una señal con un código. El vehículo recibía ese código y, si era correcto, entonces se abría. Este código siempre era el mismo, lo cual es un problema.

Existe un ataque conocido como ataque de *replay*, que consiste en capturar la señal del mando y replicarla. En un caso como este, con que un atacante capture una

vez la señal del mando ya le basta para poder abrir el vehículo todas las veces que quiera. Esto es un problema grave ya que permite a un atacante entrar al vehículo sin forzar una cerradura o romper una ventana, lo cual puede desembocar en robo de objetos, robo de información o incluso robo de vehículo si lo consigue arrancar, entre otras consecuencias.

En este caso habría que analizar si el código es totalmente fijo o tiene algún tipo de caducidad con el paso del tiempo.

Para corregir este problema del código único, se implementaron los códigos evolutivos (*rolling codes* en inglés). En este caso, el mando y el vehículo contienen un algoritmo generador de números pseudo-aleatorios (PRNG) inicializado con la misma semilla. De esta manera, el vehículo genera una lista de códigos válidos (por ejemplo, 100 números) y el mando va enviando uno a uno el siguiente número generado por el algoritmo. Así, si el vehículo recibe un código de la lista de códigos válidos, se abrirá. Además, en principio, los códigos anteriores al código recibido deberían descartarse y se deberían generar más códigos para tener una lista con la misma cantidad de códigos [1].

En primer lugar, existe una vulnerabilidad intrínseca de denegación de servicio. Si el mando se pulsa un número elevado de veces hasta salirse de la ventana de códigos válidos del vehículo (más de 100 veces, por ejemplo), entonces el mando deja de estar sincronizado con el vehículo y será necesario abrirlo de forma manual o volver a sincronizar el mando.

El ataque de *replay* ya no funciona de manera tan sencilla. Sin embargo, se puede realizar de una manera algo más sofisticada. El código capturado, en este caso, será de un solo uso, pero suficiente para poder abrir el vehículo.

A la hora de capturar un código válido del mando, es necesario que el vehículo no reciba la señal. De lo contrario, el vehículo se abrirá normalmente y el código será descartado. Entonces, es necesario introducir interferencias (mediante una antena que haga de *jammer*) en la banda de trabajo del mando para que el vehículo no reciba el código correctamente, como se muestra en la Fig. 2.1.

Paralelamente, con otra antena se deben capturar al menos dos códigos del mando (suponiendo que la víctima pulsa repetidas veces el mando para abrir su vehículo). Y la parte importante es que el vehículo debe abrirse replicando el primer código capturado y no con un código del mando de la víctima (de lo contrario, todos los códigos capturados quedarían invalidados). Así, se obtendría al menos un código

válido para poder abrir el vehículo una vez que la víctima ha terminado de utilizarlo. La otra opción es esperar a que la víctima abra y cierre el coche de forma manual [2].

Otra alternativa es capturar una serie de señales, analizarlas para extraer los códigos e intentar hacer ingeniería inversa para obtener el algoritmo de PRNG y la semilla utilizada [1].

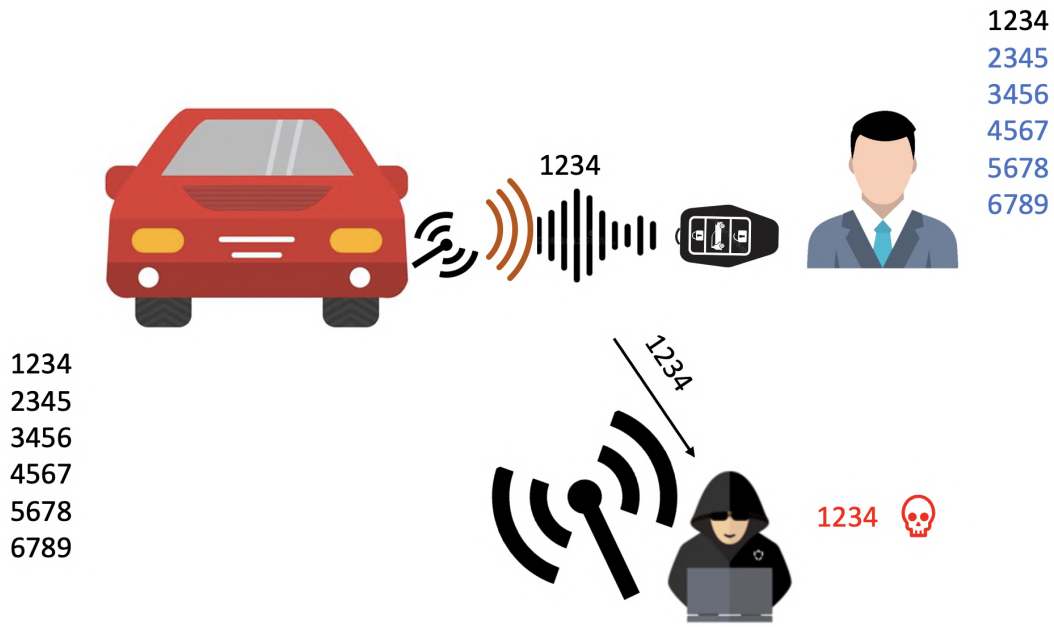

Fig. 2.1. Escenario de ataque en mandos con código evolutivo

2.2. Ataques a llaves sin contacto

Primeramente, hay que tener en cuenta que el protocolo de llaves *keyless* es de tipo pregunta-respuesta. Cuando se inicia el protocolo, el vehículo envía una pregunta, la llave responde y entonces el coche se abre. Como funcionalidad adicional, las llaves *keyless* permiten arrancar el motor sin introducir la llave en el contacto (normalmente se pulsa un botón dentro del vehículo).

Los métodos empleados para iniciar el protocolo de llaves *keyless* varían mucho

en función del fabricante. En algunos casos, es necesario poner la mano en la maneta del coche, mientras que en otros basta con acercarse al automóvil. Otros fabricantes, optan por usar canales de baja frecuencia (LF) o de alta frecuencia, como Bluetooth Low Energy (BLE) [3].

En este caso, para comprometer el vehículo se precisa de un ataque de *relay* (o *proxy*). El ataque consiste en capturar la señal que emite la llave y transferirla al vehículo mediante un repetidor, para lo cual se necesita un grupo de al menos dos atacantes [4], como se puede observar en la Fig. 2.2.

El escenario de ataque se da cuando la víctima está alejada de su vehículo y uno de los atacantes consigue aproximarse a su llave para capturar la señal de respuesta. Al momento de capturar la señal, esta se envía al segundo atacante, que está próximo al vehículo de la víctima y consigue abrir el vehículo. En caso de que el vehículo se pueda arrancar con la llave *keyless*, entonces por el mismo procedimiento el segundo atacante consigue encender el motor.

En estos casos no siempre funciona el ataque de *replay* clásico, ya que la señal de la llave puede tener un tiempo de expiración, por eso es necesario transmitirla justo al recibirla. En caso de que funcione, deberá realizarse habiendo instalado previamente un *jammer* en el vehículo de la víctima para obligarle a que tenga que abrir y cerrar el coche de forma manual, a la par que se capturan los códigos emitidos por la llave [2].

Para prevenir este tipo de ataques, algunas llaves *keyless* incorporan una protección que consiste en que se desactivan mientras no detecten movimiento. De esta manera, ya no están respondiendo a señales continuamente, sino solo durante un tiempo determinado. Aun así, mediante técnicas de ingeniería social, se podría conseguir activar la llave *keyless* y realizar el ataque de *relay*. Otra forma de prevención consiste en guardar la llave en una funda con protección contra ataques de radiofrecuencia (una jaula de Faraday) [5].

De nuevo, otra alternativa es analizar la señal que emite la llave *keyless* y realizar ingeniería inversa para poder obtener un algoritmo que genere señales válidas y así poder abrir (y arrancar) el vehículo de forma correcta. Este proceso puede ser más complicado que en el caso de los mandos a distancia porque existen llaves *keyless* que incorporan algoritmos criptográficos, tanto estándares como propietarios [3].

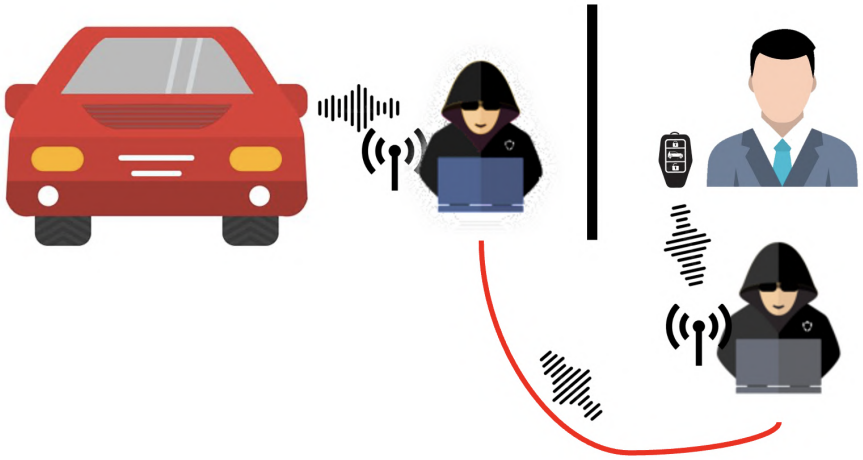

Fig. 2.2. Escenario de ataque a llaves *keyless*

Capítulo 3

Definición del trabajo

En este capítulo se explica el porqué de la realización de este proyecto. Se describirán la justificación, los objetivos a conseguir y la metodología utilizada, así como la planificación y estimación económica del proyecto.

3.1. Justificación

El propósito de este proyecto es analizar el estado de ciberseguridad de los vehículos en circulación en cuanto a comunicaciones por radiofrecuencia.

Este Trabajo Fin de Máster puede ser de gran utilidad para fabricantes de vehículos y compañías aseguradoras, ya que se presenta de manera detallada cómo se producen las comunicaciones entre los mandos de coche de los usuarios y sus respectivos vehículos, analizando posibles vulnerabilidades y ataques.

Por otro lado, el Reglamento nº 155 de las Naciones Unidas hará que los fabricantes mantengan un sistema de gestión de ciberseguridad (*CyberSecurity Management System*, CSMS) [6]; y el Reglamento nº 156 de las Naciones Unidas hará que estos tengan un sistema certificado de gestión de actualizaciones de *software* (*Software Update Management System*, SUMS) [7]. A partir de 2022, el CSMS deberá estar certificado para nuevas homologaciones [8]; y a partir de 2024, para nuevos registros.

3.2. Objetivos

Como se ha mencionado en el Capítulo 1, este Trabajo Fin de Máster forma parte del proyecto TIBET, realizado en colaboración con CESVIMAP. El objetivo de este proyecto es analizar las distintas circunstancias en las que se puede producir un ciberataque en el marco de los vehículos actuales para poder considerar las vulnerabilidades existentes en los seguros de vehículos que ofrece MAPFRE.

En el ámbito de la ciberseguridad, el proyecto TIBET pretende definir una metodología que ofrezca la posibilidad de evaluar y clasificar los vehículos dependiendo de su nivel de exposición y protección frente a ataques que permitan:

1. Robar el vehículo (ya sea mediante ataques de radiofrecuencia o mediante ataques a aplicaciones que controlen el acceso al vehículo).

2. Robar datos contenidos en el vehículo (siendo especialmente relevantes los de carácter personal).

3. Secuestrar el vehículo.

4. Controlar el vehículo de manera remota.

Para conseguir este objetivo, en primer lugar, se llevará a cabo un análisis en profundidad del estado del arte centrado en definir un modelo de riesgos, amenazas y vulnerabilidades comunes a los vehículos actuales, así como en identificar los recursos y herramientas disponibles actualmente para llevar a cabo ataques y para protegerse frente a ellos.

Este Trabajo Fin de Máster se centra exclusivamente en ataques de radiofrecuencia, cubriendo los dos primeros puntos de la lista anterior (robo de vehículo y robo de información contenida dentro del vehículo). Por este motivo, se han definido unos objetivos más concretos para completar en este TFM:

- Analizar el estado del arte: Se investigarán distintos tipos de ataque y metodologías de análisis de señales para vulnerar tanto vehículos como dispositivos que utilicen canales de RF.

- Definir una metodología de análisis y de ataque: Se entregará a CESVIMAP una metodología detallada que sirva para evaluar la seguridad de un deter-

minado vehículo en cuanto a comunicaciones en RF y posibles maneras de vulnerar el sistema.

- Desarrollar una herramienta de ataque semi-automática: Se trabajará en una herramienta *software* para capturar señales y realizar ataques de RF en base a unos parámetros dados.

3.3. Metodología de trabajo

Conforme se vayan realizando avances en la investigación de vulnerabilidades y ataques a vehículos por canales de radiofrecuencia, se irán haciendo pruebas con los recursos *hardware* y *software* disponibles para ponerlos en práctica.

Por otro lado, se mantendrán reuniones semanales con los directores del proyecto y con los encargados en CESVIMAP para llevar un seguimiento cercano y poder definir próximos pasos y solventar posibles bloqueos.

3.4. Planificación y estimación económica

Respecto a la planificación del proyecto, esta se muestra de forma visual mediante un diagrama de Gantt en la siguiente Fig. 3.1.

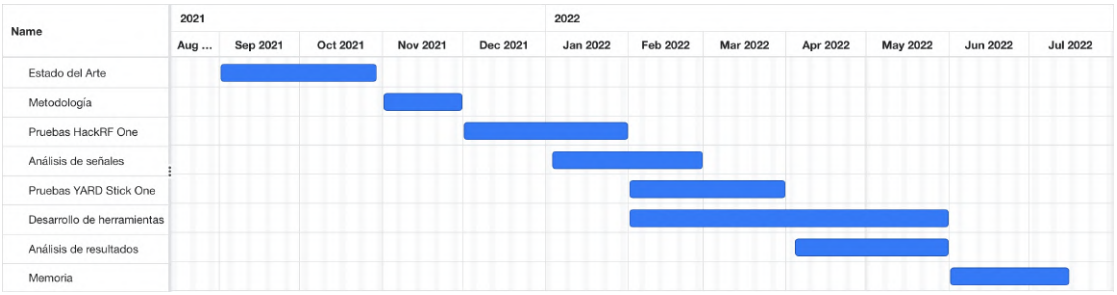

Fig. 3.1. Planificación del proyecto

Cabe mencionar que los tiempos establecidos para cada tarea son flexibles, en el sentido de que hay tareas que se iniciaron antes de lo estipulado y tareas que continuaron más allá de la fecha de finalización indicada. No obstante, este diagrama de Gantt sí describe los pasos seguidos para la realización del proyecto y la importancia de cada tarea, de manera orientativa.

Respecto a la estimación económica, se han tenido en cuenta los costes del *hardware* utilizado y de los trabajadores. Los costes de *software* son nulos, ya que se han usado tecnologías *open-source*. La Tabla 3.1 muestra los costes considerados.

Tabla 3.1. Estimación de los costes del proyecto

Concepto	Justificación	Coste (€)
Ordenador	MacBook Pro (13-inch, 2021), macOS Catalina, procesador Apple M1, memoria RAM de 16 GB, disco duro de 256 GB	1 849.00
HackRF One	Tarjeta SDR para transmitir y recibir señales analógicas (2 unidades)	2 * 295.99
YARD Stick One	Dispositivo SDR para transmitir y recibir señales digitales	105.22
Investigadores	Un investigador para llevar a cabo el proyecto, con un coste de 50 € por hora	25 000.00
Software	Programas y tecnologías utilizados	0.00
Total		27 546.20

Capítulo 4

Recursos y herramientas

En este capítulo se presentan las herramientas *hardware* y *software* utilizadas durante el proyecto para realizar ataques de RF en vehículos y analizar señales de distintos mandos.

4.1. *Hardware*

Como dispositivos *hardware*, se emplearán HackRF One y YARD Stick One, ambos de la empresa Great Scott Gadgets.

HackRF One servirá para realizar un reconocimiento básico de radiofrecuencia. Este dispositivo es un transceptor *half-duplex* que permite operar en frecuencias desde 1 MHz hasta 6 GHz y que puede llegar a una razón de 20 millones de muestras por segundo. Además, se trata de un dispositivo *open-source* [9] (ver Fig. 4.1).

Con HackRF One y el *software* adecuado, se puede visualizar el espectro de frecuencia en tiempo real, se pueden capturar señales y también replicar dichas capturas.

YARD Stick One (Fig. 4.2), por su parte, también es un transceptor *half-duplex* que opera en frecuencias 300-348 MHz, 391-464 MHz y 782-928 MHz. Este dispositivo acepta distintos tipos de modulaciones digitales (ASK/OOK, GFSK, 2FSK, 4FSK, MSK) y razones binarias de hasta 500 kbps [10].

Con YARD Stick One y `rfcat` se pueden capturar señales sabiendo sus parámetros de radiofrecuencia y también generar señales sintéticas a partir de los bits que se quieren transmitir.

Fig. 4.1. Dispositivo HackRF One

Fig. 4.2. Dispositivo YARD Stick One

4.2. *Software*

Como *software*, se utilizará GNU Radio Companion, GQRX, `inspectrum` y `rfcat`. Estos programas se pueden instalar fácilmente en un sistema operativo Linux (como Ubuntu o Kali Linux).

GNU Radio Companion es un proyecto *open-source* que proporciona un entorno de programación gráfico basado en bloques de procesamiento de señales para interactuar con dispositivos de *Software-Defined Radio* (SDR) [11]. Se usará junto con HackRF One para capturar señales y reproducir señales capturadas.

GQRX es un programa basado en GNU Radio Companion que muestra el espectro en frecuencia en formato *waterfall* y es capaz de procesar la señal de radio recibida [12]. El uso que se dará a este programa será para identificar la frecuencia de trabajo con HackRF One.

`inspectrum` es un programa que muestra la potencia de una señal en tiempo y frecuencia. Se utiliza para analizar una captura de señal y poder extraer sus características e incluso los símbolos codificados como bits [13].

`rfcat` es una librería de Python que se utiliza exclusivamente para interactuar con YARD Stick One mediante una consola de comandos interactiva o bien mediante un *script* [14].

Estos dispositivos y programas son utilizados en investigaciones de Raúl Siles como [15], en la que el resultado final es un programa en Python que enciende y apaga bombillas LED que funcionan por radiofrecuencia a voluntad mediante YARD Stick One y `rfcat`, sin necesidad de usar el mando a distancia del fabricante; o en proyectos como ToorChat, un programa de mensajería instantánea usando dispositivos YARD Stick One [16]. Estos ejemplos sirven como prueba de concepto del potencial que tienen los recursos y herramientas descritos anteriormente.

Capítulo 5

Sistema desarrollado

En este capítulo se explica en detalle la metodología propuesta y el código desarrollado en el presente proyecto.

En primer lugar, se explicará la metodología de análisis y ataque requerida por CESVIMAP (MAPFRE) para poder evaluar la ciberseguridad de los vehículos e incluir dicha valoración en el cálculo de las pólizas de seguros.

Y en segundo lugar, se mostrará el programa desarrollado para implementar la metodología de análisis anterior, de manera que pueda ser de utilidad para el personal que evalúe la seguridad de los protocolos de radiofrecuencia de cada modelo de vehículo.

5.1. Metodología propuesta

En primera instancia, hay que identificar la banda de trabajo del vehículo y el mando o la llave *keyless* (normalmente es la banda de 433 MHz o 868 MHz, bandas no licenciadas). Para el caso de llaves *keyless*, es posible que su banda de trabajo sea en baja frecuencia (canales de 22 kHz, 125 kHz y 134.2 kHz) o en alta frecuencia (BLE, en 2.4 GHz) [3].

Una vez identificada la frecuencia, se procede a realizar una captura de la señal con HackRF One y GNU Radio Companion. Esta captura puede ser reproducida para

efectuar un ataque de *replay*. No obstante, si el ataque se queda en este punto, sería un estilo *script-kiddie*. Además, la captura no es perfecta, contiene ruido, y es probable que el ataque no funcione como debería.

Como se ha comentado anteriormente, esta metodología se amplía realizando un análisis de la señal capturada digitalmente. Para entender la señal, será necesario tomar varias muestras y representarlas en `inspectrum`, para poder extraer los símbolos y los bits codificados. Esta parte del análisis es manual, ya que es necesario insertar cursores y extraer los símbolos de la señal. En algunos casos (por ejemplo, en señales con modulación ASK/OOK), la extracción de símbolos se puede realizar de forma automática con `inspectrum`.

Una vez que se conoce el formato binario de la señal capturada, se puede usar YARD Stick One para sintetizar una señal que porte la información extraída. Para que esta señal sintética funcione correctamente, será necesario realizar las siguientes tareas de manera iterativa:

- Configurar los parámetros de YARD Stick One.
- Transmitir la señal sintética con YARD Stick One.
- Capturar la señal sintética con HackRF One.
- Analizar la captura de la señal sintética con `inspectrum` y comparar con la señal que se quiere suplantar.

En el momento en que la señal original sea equivalente a la señal sintética, el dispositivo YARD Stick One estará correctamente configurado tanto para transmitir como para recibir. Además, YARD Stick One será capaz de demodular una señal y extraer los bits codificados de forma automática.

Con esto, se consigue que el análisis de la información enviada por un cierto dispositivo (una llave de vehículo) sea mucho más eficiente. Simplemente, se requiere una determinada configuración de `rfcat` en función de los parámetros de RF utilizados por un cierto modelo de vehículo.

Esta metodología se puede expresar gráficamente con un diagrama de actividad como el que aparece en la Fig. 5.1.

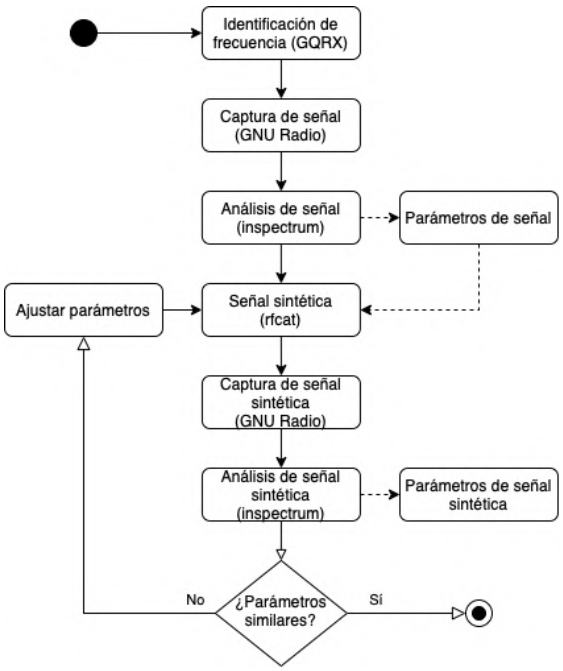

Fig. 5.1. Diagrama de actividad para la metodología desarrollada

5.2. Programa de recepción y transmisión desarrollado

El programa desarrollado se llama `rf_attack.py` y está escrito en Python (versión 3.x). Como dependencias, utiliza la librería `rflib` (API que viene incluida con `rfcat`), además de módulos de la propia API de Python.

5.2.1. Diseño del código

El programa `rf_attack.py` está pensado para usarse en una consola, es decir, se trata de una herramienta con una interfaz de línea de comandos (*Command Line Interface*, CLI). Para usar la herramienta, es necesario usar una serie de parámetros según la función que se quiera emplear.

Hay un panel de ayuda para recordar el uso de cada parámetro (ver Fig. 5.2).

Fig. 5.2. Panel de ayuda de la herramienta `rf_attack.py`

El parámetro principal es `-f/--file`, en el que se indica la ruta a un archivo de configuración en JSON. En este fichero se indican los parámetros de radiofrecuencia del protocolo con el que se está trabajando. Un ejemplo de este archivo es el Código 5.1, usado para los modelos de vehículos de Marca A.

Se ha decidido usar este método de configuración mediante un archivo externo para no incluirlo en el propio código ni tener que configurar la herramienta mediante parámetros individuales o mediante la entrada estándar (`stdin`). De esta manera, se pueden almacenar diversos archivos con configuraciones de distintos modelos de vehículos y tener un mayor control sobre dichas configuraciones y versiones.

A continuación del parámetro del archivo de configuración, se deberá usar o bien `-r/--receive` o bien `-t/--transmit` para recibir o transmitir, respectivamente.

Se ha implementado una gestión de errores en caso de que exista algún parámetro faltante, algún parámetro incorrecto o algún error de formato.

```
{
  "MdmSyncSequence": "fffffffffffe",
  "MdmSyncMode": 1,
  "Freq": 433900000,
  "PktPQT": 0,
  "MdmDRate": 1997,
  "PktFLEN": 21,
  "MaxPower": true,
  "MdmModulation": "MOD_2FSK",
  "EnableMdmManchester": true,
  "Fields": {
    "Sync": [0, 7],
    "Command": [7, 8],
    "Variable": [8, 9],
    "Key ID": [9, 13],
    "Rolling code": [13, 21]
  }
}
```

Código 5.1. Configuración con parámetros de RF para vehículos de Marca A

En la Sección B.6 se explica el uso de cada campo del archivo de configuración.

5.2.2. Función de recepción

En esta función, se configura YARD Stick One para recibir señales con los parámetros de RF indicados en el archivo de configuración.

El dispositivo estará en escucha indefinidamente hasta que se fuerce la salida del programa. En el momento en el que se recibe una señal que empiece por los primeros 16 bits de la secuencia indicada en el campo `"MdmSyncSequence"` del archivo de configuración, dicha trama binaria pasará a ser analizada.

En el caso de que se detecte la secuencia de sincronismo completa en el contenido de la señal, indicada en la clave `"MdmSyncSequence"`, la trama podrá ser descompuesta en distintos campos, según la estructura definida en `"Fields"`. Si se añade el parámetro `-d/--debug` al comando de `rf_attack.py`, se podrán visualizar todas las señales recibidas, además de las que se analicen con mayor detalle.

Adicionalmente, se puede emplear el parámetro `-o/--output` para especificar un archivo en el que almacenar el contenido de las señales que se han conseguido analizar en detalle (en formato binario).

5.2.3. Función de transmisión

En esta función, se configura YARD Stick One para transmitir una señal a partir de su contenido en binario (codificado en hexadecimal) con los parámetros de RF indicados en el archivo de configuración.

Esta señal se transmite de forma indefinida hasta que se termine la ejecución del programa de manera forzada. El uso de un bucle infinito se debe a que es necesario enviar la señal múltiples veces para asegurarse de que el sistema receptor recibe la señal correctamente.

5.2.4. Pruebas de funcionamiento

A continuación, se muestran unos ejemplos de funcionamiento mediante capturas de la consola. En la Fig. 5.3 se puede observar el modo de recepción de señales con modo de depuración activado para visualizar todas las tramas recibidas. En este caso existen dos señales que se consiguen descomponer en los campos que aparecen en la configuración del Código 5.1. El comando `7b` significa abrir el vehículo, mientras que `3b` se usa para cerrarlo.

Cabe resaltar que las señales que no se consiguen descomponer, son señales válidas del vehículo, pero no comienzan exactamente por la secuencia de sincronismo indicada. Una de las señales tiene una secuencia de sincronismo más larga (un byte), por lo que no se puede obtener su código evolutivo correctamente, ya que no estaría completo.

En la otra señal recibida en modo depuración, se ve que la secuencia de sincronismo termina en `fc` y no en `fe` como aparece en la configuración. Una posible explicación es que la primera secuencia de sincronismo se encuentra desplazada un bit a la izquierda, ya que `0xfc` es `"1111 1100"` en binario, y `0xfe` es `"1111 1110"`. Se podría tratar de corregir esta señal para analizarla en profundidad, pero es probable que contenga más errores de recepción.

Por otro lado, en la Fig. 5.4 se ve cómo se envía una señal con contenido en hexadecimal `ffffffffffffe3b3lee66491d367f0c378461cb18` de forma indefinida hasta que se fuerza la salida del programa.

Fig. 5.3. Recepción de señales con `rf_attack.py`

Cabe mencionar que las pruebas realizadas con `rf_attack.py` se han llevado a cabo en dos modelos distintos de vehículos de Marca A, en distintos mandos con códigos evolutivos. En ambos casos, la estructura de las tramas era la misma, por lo que se empleó un solo archivo de configuración para analizar ambas llaves.

No se ha podido usar la herramienta desarrollada en llaves con protocolo *keyless* debido a que no se ha conseguido analizar las señales de este tipo de llaves en profundidad.

Fig. 5.4. Transmisión de señales con `rf_attack.py`

Capítulo 6

Análisis de resultados

Para los resultados obtenidos hasta la fecha, no se indicarán los fabricantes de vehículos analizados por motivos de privacidad.

6.1. Análisis de señales de mandos tradicionales

La Tabla 6.1 compara las señales usadas por modelos de vehículos de distintos fabricantes.

Tabla 6.1. Características de señales capturadas

Fabricante	Modulación	Longitud
Marca A	2FSK	Una trama de 168 bits
Marca B	2FSK	3 tramas de 120 bits
Marca C	ASK/OOK	Una trama de 312 bits
Marca D	ASK/OOK	6 tramas de 312 bits y 27 tramas de 32 bits

Como se observa en la Tabla 6.1, los fabricantes analizados utilizan para transmitir la información modulaciones digitales sencillas como 2FSK (Fig. 6.1) o ASK/OOK (Fig. 6.2). Además, en todos los casos analizados se utiliza codificación Manchester, en la que la información se codifica por flanco (por ejemplo, flanco de subida: 0

lógico y flanco de bajada: 1 lógico, o viceversa según el convenio), proporcionando así sincronismo a nivel de símbolo.

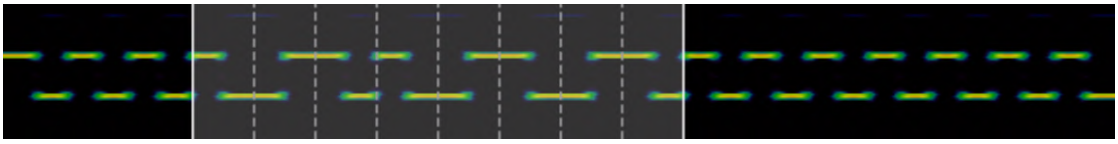

Fig. 6.1. Muestra de señal capturada para un vehículo de la Marca A

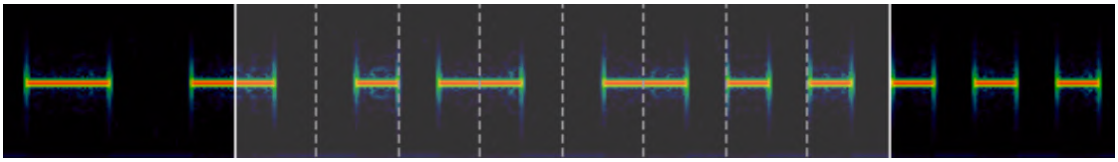

Fig. 6.2. Muestra de señal capturada para un vehículo de la Marca D

Todas las señales cuentan con una secuencia de sincronismo al comienzo de la trama (Fig. 6.3) que consiste en una sucesión de 1 lógicos hasta llegar a un 0 lógico que indica el comienzo de los datos transmitidos. El uso de la codificación Manchester permite precisamente que esta secuencia de sincronismo tenga efecto en el receptor y pueda establecer el periodo de símbolo para demodular la señal.

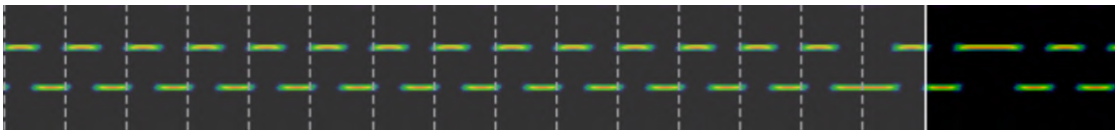

Fig. 6.3. Secuencia de sincronismo de señal capturada para un vehículo de la Marca A

6.2. Resultados individuales para el modelo de la Marca A analizado

Tras realizar el procedimiento descrito en la Sección 5.1 con la señal capturada del modelo de Marca A, se consiguió configurar YARD Stick One para poder transmitir y recibir usando `rfcat`.

Estos son algunos ejemplos de la información transmitida por el mando del vehículo cuando se pulsa el botón de apertura (en hexadecimal):

- `fffffffffffffe7b5fee66491d0592c722769755eb`.
- `fffffffffffffe7b9fee66491db9af74d04ba1ae4e`.
- `fffffffffffffe7b1fee66491d34a91610d4848854`.
- `fffffffffffffe7befee66491deb94957806d52b0c`.

Como se puede observar, hay varios dígitos que coinciden en las cuatro muestras, lo cual indica que la información está dividida en diferentes secciones binarias. Tomando la primera muestra, se pueden identificar las siguientes secciones:

- Secuencia de sincronismo: `fffffffffffffe`.
- Acción: `7b`.
- Valor cambiante: `5f`.
- Valor fijo: `ee66491d`.
- Código evolutivo: `0592c722769755eb`.

Hasta el momento, se sabe que la acción será `7b` para abrir el vehículo, `3b` para cerrarlo y `5d` para abrir el maletero. Por otro lado, por el momento se desconoce la función del valor cambiante, aunque se cree que puede tratarse de algún método de detección de errores. El valor fijo se corresponde con un identificador de la llave. De esta manera, solamente las llaves configuradas para un vehículo funcionarán (típicamente, dos llaves: la de uso habitual y la de repuesto). Por último, el código evolutivo es un número pseudo-aleatorio de 64 bits, lo cual es suficientemente robusto ante un ataque de fuerza bruta.

Finalmente, cabe destacar que se logró transmitir una de estas secuencias de bits extraídas mediante YARD Stick One y abrir el vehículo, disponiéndose de un vídeo que lo demuestra que podrá mostrarse en la defensa del TFM.

6.3. Análisis de llaves *keyless*

En esta sección se muestran algunas pruebas realizadas en automóviles con sistema de apertura y arranque *keyless*. Pese a no tener suficientes muestras de los vehículos analizados, se han podido extraer algunas conclusiones generales sobre la seguridad que presentan los protocolos utilizados.

6.3.1. Captura y análisis de señales *keyless*

Se empezó por analizar una furgoneta de Marca E. Este vehículo utiliza un protocolo *keyless* tanto para la apertura de puertas como para el arranque del vehículo.

En primer lugar, se vio que, al acercarse el usuario al coche en posesión de la llave, el vehículo detecta la presencia de la llave y se abre. En este punto, se analizó el espectro de frecuencia en la banda de 433 MHz y se distinguieron dos señales que se transmitían de manera periódica (ver Fig. 6.4 y Fig. 6.5). La primera señal corresponde a una pregunta que realiza el vehículo y la segunda señal es la respuesta por parte de la llave.

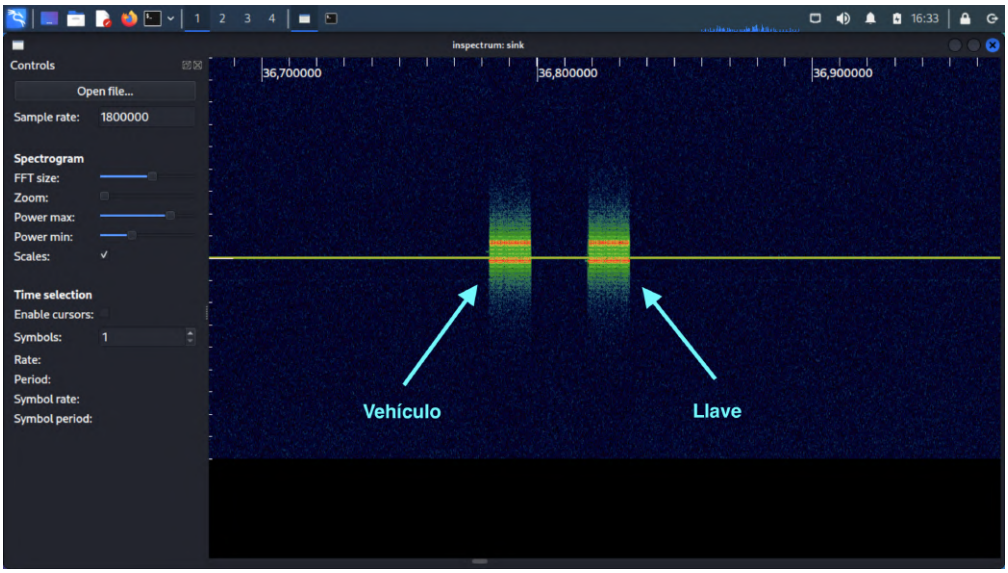

Fig. 6.4. Modelo de pregunta-respuesta en coche de Marca E (*keyless*)

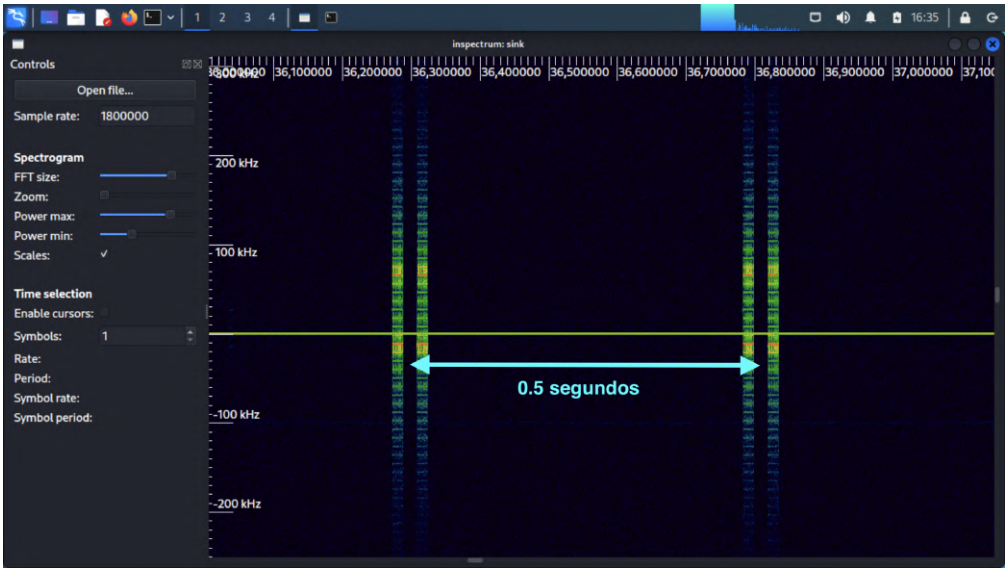

Fig. 6.5. Periodo entre interacciones en coche de Marce E (*keyless*)

Este par de señales se repite cada medio segundo. Se deduce, por tanto, que es el mecanismo utilizado por el vehículo para confirmar que la llave está y se mantiene cerca del vehículo.

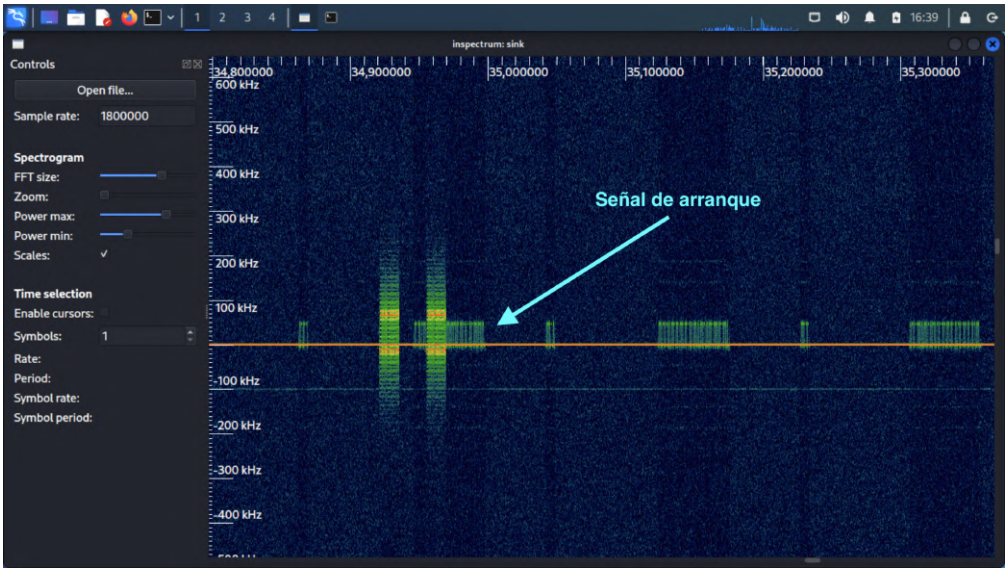

Fig. 6.6. Señal de arranque de motor en coche de Marce E (*keyless*)

A la hora de arrancar el motor, es necesario que la llave esté dentro del vehículo y que el usuario pulse un botón de encendido. En este momento, aparece una señal nueva en el espectro de frecuencia, tal y como se observa en la Fig. 6.6.

Cabe mencionar que, para que el arranque del motor se produzca, debe existir una comunicación bidireccional entre el coche y la llave (cada medio segundo).

Por otro lado, se capturaron otro tipo de señales de las que no se tiene explicación, como las mostradas en la Fig. 6.7.

Además, con las capturas obtenidas, no es posible analizar el contenido de las mismas y extraer el mensaje codificado en binario, como se hizo con los mandos a distancia tradicionales. Quizás se deba a que la configuración de la tarjeta HackRF One en GNU Radio Companion no era correcta para este escenario.

Por último, se puede decir que se desconoce cómo se inicia el protocolo de pregunta-respuesta explicado anteriormente. En el caso del coche de Marca E, bastaba con acercarse al vehículo con la llave para que se iniciara el protocolo, no siendo necesario tocar la maneta de la puerta. Es posible que exista otro canal de comunicación a baja frecuencia (posiblemente en 125 kHz) o bien a alta frecuencia (en 2.4 GHz, posiblemente Bluetooth). Desafortunadamente, con los dispositivos SDR que se tienen, no es posible analizar estas frecuencias.

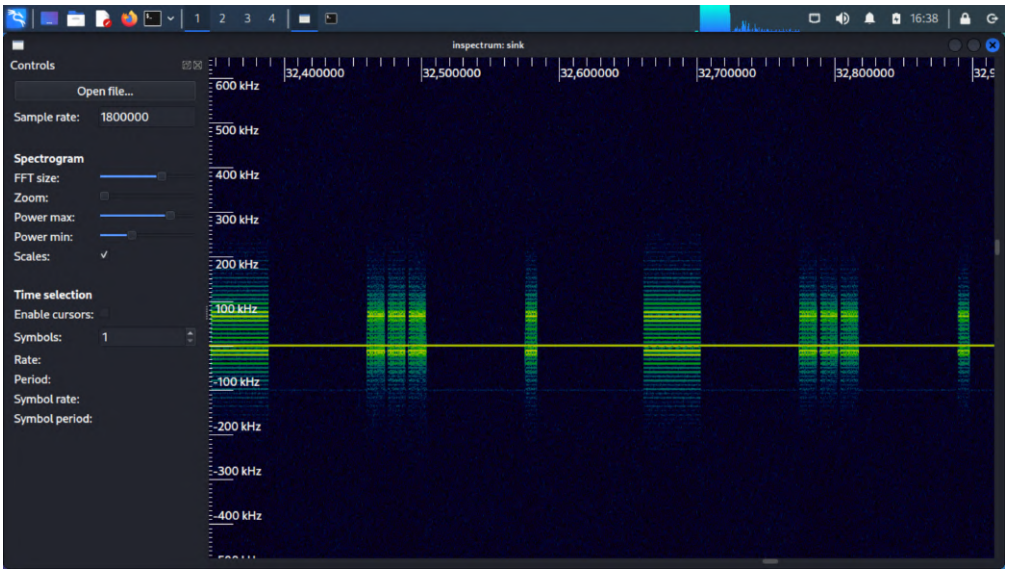

Fig. 6.7. Otras señales capturadas para el vehículo de Marce E (*keyless*)

6.3.2. Análisis de *chipset*

En este apartado se analiza una placa de circuito impreso (*Printed Circuit Board*, PCB) de una llave *keyless* de un vehículo de Marca F. Con este análisis se pretende identificar distintos *chips* presentes en la PCB para poder deducir el protocolo de comunicación utilizado entre la llave y el vehículo.

La Fig. 6.8 muestra una cara de la PCB, donde se puede distinguir un *chip* con identificador F7953AC1500. Se trata de un receptor que funciona en baja frecuencia (LF, 125 kHz) con un microcontrolador integrado.

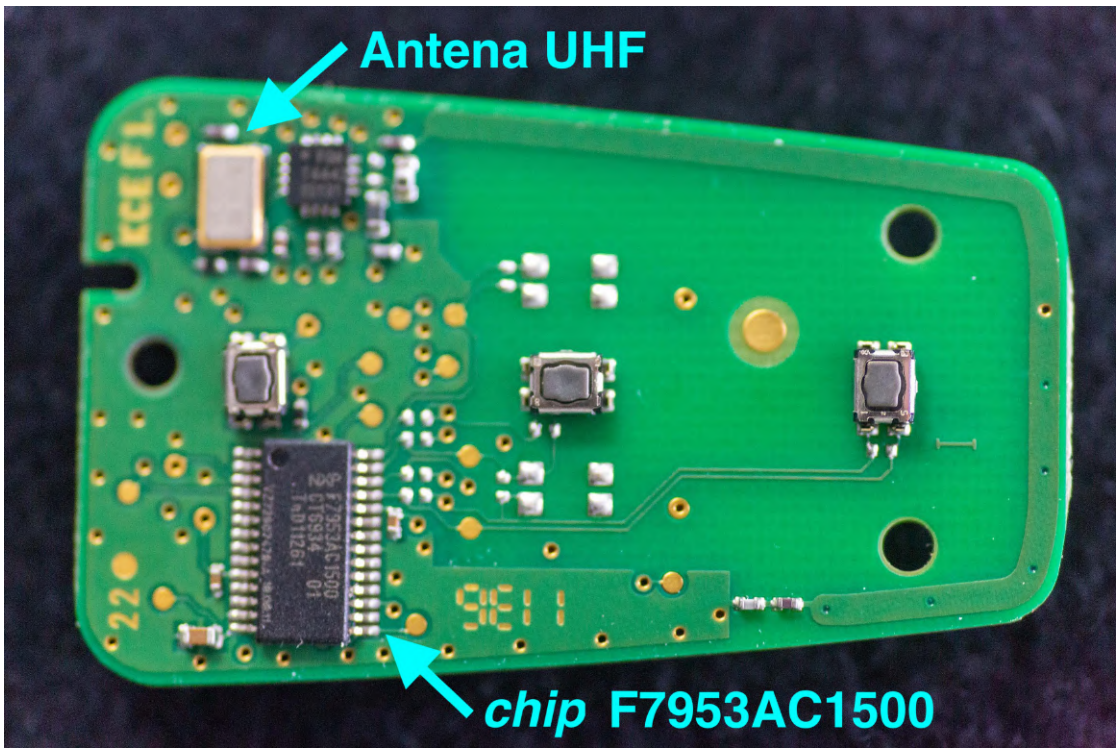

Fig. 6.8. Placa de circuito impreso de una llave *keyless* (1)

Analizando la *datasheet* del componente, disponible en [17], se ve que está diseñado para funcionar con un microcontrolador que active un transmisor UHF (banda de frecuencia entre 300 MHz y 3 GHz).

El funcionamiento del protocolo comienza con que el usuario se acerca al vehículo y pone su mano en la maneta de la puerta, haciendo que el vehículo emita una señal

en LF para activar la llave *keyless*. La llave responde si la información contenida en la señal coincide con la que está grabada en la propia llave. Si es así, la llave envía su señal de respuesta encriptada en UHF. El vehículo compara la respuesta de la llave con la información que tiene guardada y determina si la autenticación es correcta o no. Si es correcta, las puertas del coche se abren.

A la hora de arrancar el vehículo, se reinicia el protocolo para verificar que la llave está dentro del coche.

Este funcionamiento encaja con el *chipset* presente en la PCB, ya que aparece la antena UHF en la parte superior izquierda de la Fig. 6.8 y la antena LF en la parte posterior de la PCB (ver Fig. 6.9). Ambas antenas son inductivas.

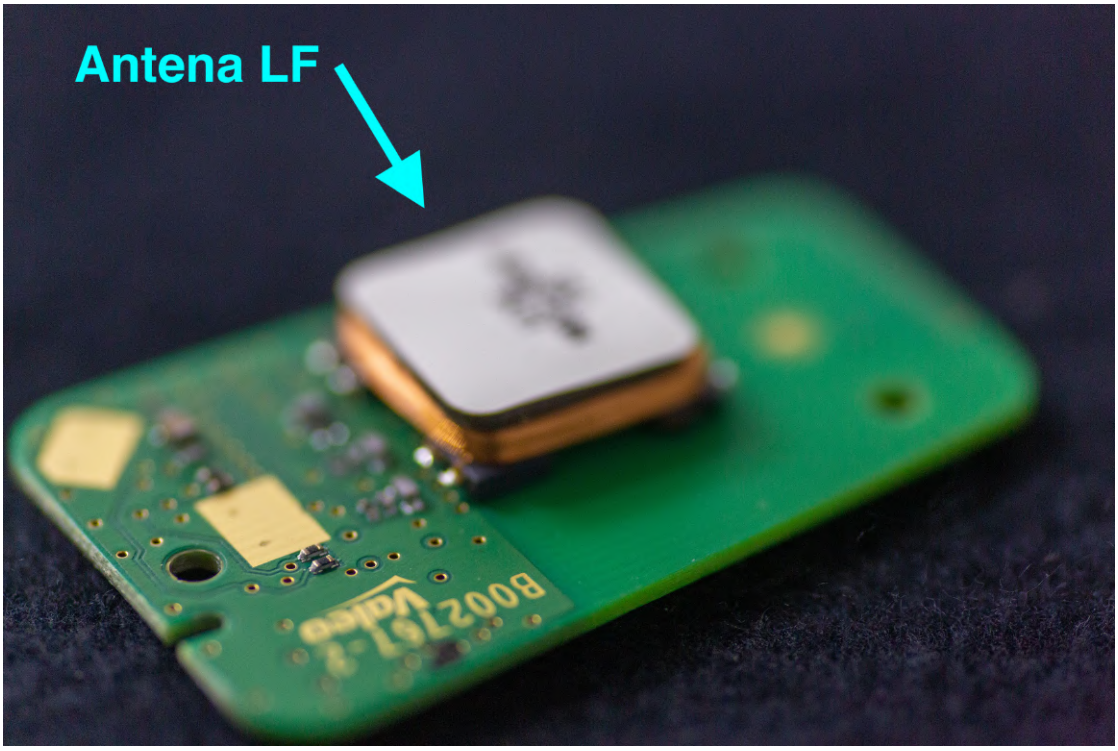

Fig. 6.9. Placa de circuito impreso de una llave *keyless* (2)

La siguiente Fig. 6.10 muestra un esquema de componentes necesarios para que el protocolo funcione, según el fabricante del *chip* con identificador F7953AC1500:

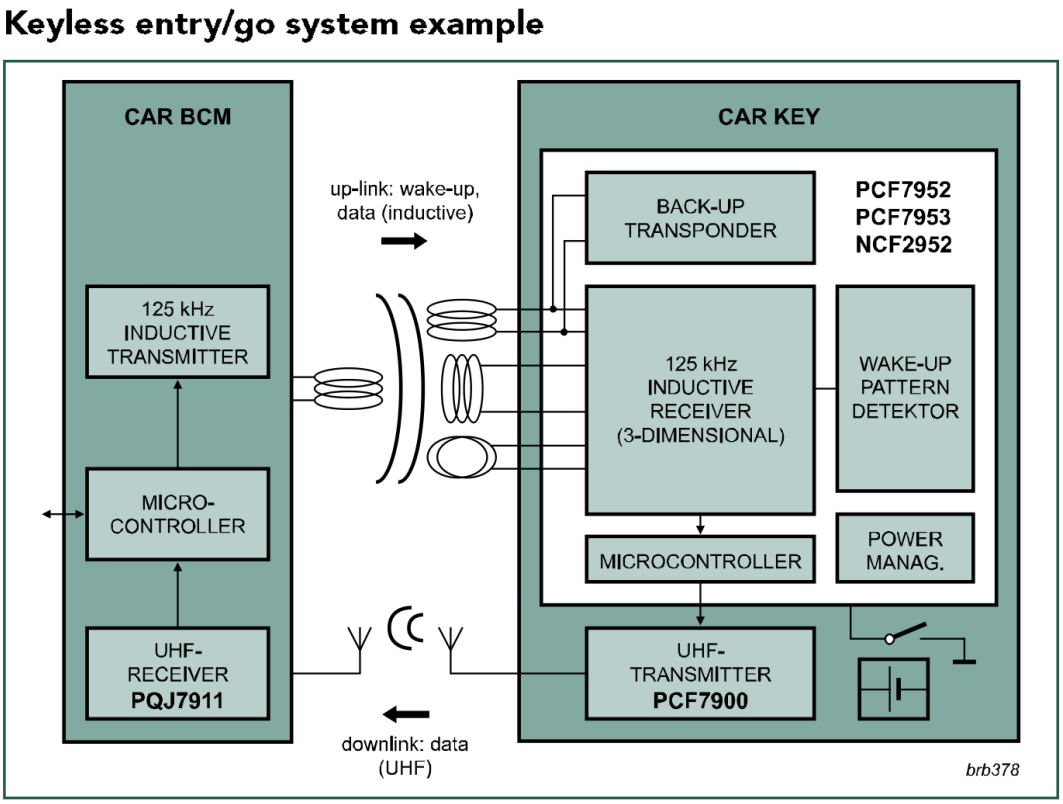

Fig. 6.10. Esquema de funcionamiento de sistema *keyless* [17]

Capítulo 7

Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones más importantes de la realización de este Trabajo Fin de Máster, analizando los objetivos propuestos y los resultados obtenidos. Además, se indicarán una serie de trabajos futuros en relación con el proyecto desarrollado.

7.1. Conclusiones y resultados principales

En primer lugar, cabe mencionar que se han cumplido los tres objetivos expuestos en la Sección 3.2:

- Se ha realizado un análisis del estado del arte y se han identificado los tipos de ataques de radiofrecuencia documentados en la literatura.
- Se ha definido una metodología de análisis de señales de RF en digital para poder evaluar la seguridad de los protocolos de comunicación implementados por los fabricantes de vehículos.
- Se ha desarrollado una herramienta llamada `rf_attack.py` que permite capturar señales y realizar la demodulación y análisis del contenido digital de la señal de forma automática, dados unos parámetros de RF. Además, también es posible transmitir señales a partir de una trama en binaria con los mismos parámetros de radiofrecuencia.

Por otro lado, se ha conseguido realizar un ataque de *replay* exitoso generando la señal de apertura desde su representación en digital, en un vehículo con sistema de códigos evolutivos. El hecho de poder analizar el contenido binario de las señales proporciona mucha información sobre el protocolo y permite comprender cómo funciona, de manera que los ataques dejan de ser de tipo *script-kiddie*.

Adicionalmente, parte del TFM se publicó en un *paper* para las VII Jornadas Nacionales de Investigación en Ciberseguridad (JNIC), accesible en [18]. La presentación se realizó en Bilbao (28 de julio de 2022), ante personalidades del INCIBE, de TECNALIA, de RENIC y del CSIC, entre otras, además de multitud de doctores y alumnos de doctorado de reconocido prestigio.

Este acontecimiento es un buen comienzo en el ámbito de la investigación en ciberseguridad y fue una experiencia muy enriquecedora, al poder presentar personalmente mi investigación, como parte de mi TFM, y al tener contacto con otras personas que se dedican a tareas similares y realizan investigaciones sobre tecnologías y conceptos muy variados, compartiendo sus conocimientos y resultados con la comunidad.

7.2. Trabajos futuros

Una vez terminado el presente proyecto, surgen nuevas ideas para continuar con la investigación y otras líneas de investigación de interés para empresas como CESVIMAP (MAPFRE). Se pueden considerar las siguientes tareas adicionales:

- Continuar con el análisis de protocolos *keyless*: Estos protocolos son más complejos y variados al necesitar una comunicación bidireccional entre el vehículo y la llave, que puede darse en bandas de frecuencia distintas. Además, en este tipo de vehículos, el arranque de motor también funciona mediante protocolo *keyless*, con lo que un atacante podría robar el vehículo más fácilmente.

- Comprobar la gestión de códigos evolutivos: Realizar capturas de muestras consecutivas de las señales de un mando con códigos evolutivos, transmitir la última muestra capturada (debería abrir el vehículo) y después el resto para comprobar si funcionan. En caso afirmativo, se podrá concluir que el vehículo no invalida los códigos anteriores a un cierto código evolutivo si no han sido usados, lo cual sería un problema de ciberseguridad porque un

atacante podría capturar códigos válidos mediante el ataque documentado en la Sección 2.1 y que sean válidos de forma indefinida, ya que no se descartan al usar los siguientes códigos generados por el algoritmo de PRNG.

- Analizar el algoritmo de generación de códigos evolutivos: El PRNG utilizado para generar los distintos códigos evolutivos tiene un mecanismo de resincronización para que se pueda volver a sincronizar un mando en caso de que no funcione (por ejemplo, si se pulsa el botón repetidas veces y el siguiente paso del PRNG no aparece en la lista de códigos válidos que admite el vehículo). Esta propiedad puede ser útil para aplicar ingeniería inversa y obtener el algoritmo de PRNG, la semilla o bien una manera de obtener el siguiente código en función de los anteriores.

- Analizar flotas de vehículos de *car-sharing*: Las compañías aseguradoras como MAPFRE tienen especial interés en este tipo de vehículos, los cuales son de uso extendido en la actualidad. En este caso, la apertura del vehículo se realiza mediante una aplicación móvil y protocolos como Bluetooth, RFID o incluso comunicaciones móviles. Por este motivo, existe una superficie de exposición mayor a un coche particular, ya que entran en juego aplicaciones móviles, autenticación de usuarios y otros protocolos de comunicación.

- Investigar el uso de Bluetooth: Existen dispositivos comerciales OBD (*On Board Diagnostics*) que se conectan al bus CAN (*Controller Area Network*) principalmente para tareas de diagnóstico. Entre la amplia gama de dispositivos OBD, algunos que establecen conexión por Bluetooth, lo cual permite controlar dichos dispositivos desde aplicaciones móviles. Este tipo de conectores OBD con Bluetooth con cada vez más populares porque resultan más económicos y fáciles de manejar que los conectores por cable; y podrían quedar conectados en el vehículo y pasar desapercibidos, siendo posible para un atacante recabar información del vehículo y actuar sobre el bus CAN, lo cual podría ocasionar accidentes catastróficos y pérdidas económicas [19].

Estas nuevas pruebas y líneas de investigación podrían formar parte de nuevos *papers* y publicarse en revistas científicas con temática de ciberseguridad, como se realizó en las VII JNIC (2022).

Las líneas de trabajo relativas a gestión de códigos evolutivos y análisis de números pseudo-aleatorios son posibles gracias a poder trabajar con las señales en digital. Este es un gran paso, pues permite comprender el funcionamiento del protocolo empleado.

Además, debido a la entrada en vigor de normativa que obliga a los fabricantes de automóviles a tener homologaciones de seguridad (ver Sección 3.1), se espera una ola de investigación e innovación en esta área. En este sentido, se descubrirán nuevas metodologías, vulnerabilidades y ataques que puedan comprometer vehículos de distintas formas.

Bibliografía

[1] A. Mohawk, *Bypassing Rolling Code Systems*, 5 de feb. de 2016. [En línea]. Disponible en: https://www.andrewmohawk.com/2016/02/05/bypassing-rolling-code-systems/ (Accedido: 05-04-2022) (citado en pp. 12, 13).

[2] O. Ibrahim, A. Hussain, G. Oligeri y R. Pietro, «Key is in the Air: Hacking Remote Keyless Entry Systems,» en. 10 de ene. de 2019, págs. 125-132, ISBN: 978-3-030-16874-2. DOI: 10.1007/978-3-030-16874-2_9 (citado en pp. 13, 14).

[3] L. Wouters, B. Gierlichs y B. Preneel, «My other car is your car: compromising the Tesla Model X keyless entry system,» *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, n.º 4, págs. 149-172, ago. de 2021. DOI: 10.46586/tches.v2021.i4.149-172. [En línea]. Disponible en: https://tches.iacr.org/index.php/TCHES/article/view/9063 (Accedido: 28-05-2022) (citado en pp. 14, 25).

[4] J. Edkins, *Keyless car theft: what is it and how to prevent it*, 11 de feb. de 2022. [En línea]. Disponible en: https://www.carwow.co.uk/blog/keyless-car-theft-prevention (Accedido: 05-04-2022) (citado en p. 14).

[5] *Keyless car theft: What is a relay attack, how can you prevent it, and will your car insurance cover it?* Leasing.com. [En línea]. Disponible en: https://leasing.com/guides/relay-car-theft-what-is-it-and-how-can-you-avoid-it/ (Accedido: 05-04-2022) (citado en p. 14).

[6] «Reglamento nº 155 de la Comisión Económica para Europa (CEPE) de las Naciones Unidas - Disposiciones uniformes relativas a la homologación de los vehículos de motor en lo que respecta a la ciberseguridad y al sistema de gestión de esta [2021/387],» 9 de mar. de 2021. [En línea]. Disponible en: https://www.boe.es/doue/2021/082/L00030-00059.pdf (Accedido: 16-06-2022) (citado en p. 17).

[7] «Reglamento nº 156 de las Naciones Unidas - Disposiciones uniformes relativas a la homologación de vehículos en lo que respecta a las actualizaciones de software y al sistema de gestión de actualizaciones de software [2021/388],» 9 de mar. de 2021. [En línea]. Disponible en: https://www.boe.es/doue/2021/082/L00060-00074.pdf (Accedido: 16-06-2022) (citado en p. 17).

[8] U. Mezcua, «Los coches deberán tener certificado de ciberseguridad para venderse a partir de 2022,» 12 de ago. de 2020. [En línea]. Disponible en: https://www.abc.es/motor/reportajes/abci-coches-deberan-tener-certificado-ciberseguridad-para-venderse-partir-2022-202008120103_noticia.html (Accedido: 31-05-2022) (citado en pp. 17, 74).

[9] *HackRF One*, Great Scott Gadgets. [En línea]. Disponible en: https://greatscottgadgets.com/hackrf/one/ (Accedido: 02-07-2022) (citado en p. 21).

[10] *YARD Stick One*, Great Scott Gadgets. [En línea]. Disponible en: https://greatscottgadgets.com/yardstickone/ (Accedido: 02-07-2022) (citado en p. 21).

[11] *About GNU Radio*, GNU Radio. [En línea]. Disponible en: https://www.gnuradio.org/about/ (Accedido: 02-07-2022) (citado en p. 22).

[12] A. C. OZ9AEC, *Gqrx SDR - Open source software defined radio by Alexandru Csete OZ9AEC*, Gqrx SDR. [En línea]. Disponible en: https://gqrx.dk/ (Accedido: 02-07-2022) (citado en p. 23).

[13] *Inspectrum: A New Tool for Analysing Captured Signals*, RTL-SDR, 23 de mar. de 2016. [En línea]. Disponible en: https://www.rtl-sdr.com/inspectrum-a-new-tool-for-analyzing-captured-signals/ (Accedido: 02-07-2022) (citado en p. 23).

[14] *rfcat*. [En línea]. Disponible en: https://github.com/atlas0fd00m/rfcat (Accedido: 02-07-2022) (citado en pp. 23, 53).

[15] R. Siles, *La cena de los idIoTas*, 2016. [En línea]. Disponible en: https://www.dinosec.com/docs/La_cena_de_los_idIoTas-RaulSiles-DinoSec-RootedCON2016_v1.0.pdf (Accedido: 01-07-2022) (citado en p. 23).

[16] *ToorChat*, Hak5. [En línea]. Disponible en: https://github.com/hak5/ToorChat (Accedido: 01-07-2022) (citado en p. 23).

[17] *NXP keyless entry/go solutions*, NXP, abr. de 2012. [En línea]. Disponible en: https://tvsat.com.pl/PDF/P/PCF7952-3_NXP.pdf (Accedido: 10-06-2022) (citado en pp. 39, 41).

[18] R. G. Miñarro *et al.*, «Metodología y herramientas para análisis y evaluación de seguridad frente a ataques de radiofrecuencia en vehículos,» *Investigación en Ciberseguridad. Actas de las VII Jornadas Nacionales (JNIC 2022)*, págs. 167-171, jun. de 2022. [En línea]. Disponible en: https://2022.jnic.es/Actas_JNIC_2022_v11.pdf (Accedido: 30-06-2022) (citado en p. 44).

[19] M. Bozdal, M. Samie, S. Aslam e I. Jennions, «Evaluation of CAN Bus Security Challenges,» *Sensors (Basel)*, 21 de abr. de 2020. DOI: 10.3390/s20082364. [En línea]. Disponible en: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7219335/ (Accedido: 01-07-2022) (citado en p. 45).

[20] gamb1t, *Updating Kali*, Kali, 13 de dic. de 2021. [En línea]. Disponible en: https://www.kali.org/docs/general-use/updating-kali/ (Accedido: 08-06-2022) (citado en p. 51).

[21] *Objetivos de Desarrollo Sostenible*, ONU. [En línea]. Disponible en: https://www.un.org/sustainabledevelopment/es/ (Accedido: 01-07-2022) (citado en pp. 71, 73).

[22] *Desarrollo Sostenible. ¿Qué es y cómo alcanzarlo?* Acciona. [En línea]. Disponible en: https://www.acciona.com/es/desarrollo-sostenible/ (Accedido: 10-06-2020) (citado en p. 71).

[23] A. Delgado, «La década prodigiosa de la seguridad vial,» 30 de ene. de 2020. [En línea]. Disponible en: https://revista.dgt.es/es/noticias/nacional/2020/01ENERO/0130-ODS-Seguridad-Vial.shtml (Accedido: 31-05-2022) (citado en p. 74).

Anexo A. Guía de instalación

En este anexo se detalla el procedimiento para instalar y preparar las herramientas utilizadas durante el proyecto para el análisis de señales de radiofrecuencia y para llevar a cabo los distintos ataques mencionados en la memoria.

Las herramientas presentadas funcionan para la fecha en la que se realizó el proyecto. Es posible que con el paso del tiempo aparezcan nuevas versiones y las que se muestran a continuación queden obsoletas. Aún así, el procedimiento de instalación será similar, además de que funciona para arquitecturas amd64 y arm64 y también en entornos virtuales.

9.3. Preparación del sistema operativo

En primer lugar, hay que arrancar un sistema operativo Linux (preferiblemente Kali Linux o Ubuntu, puede ser en una máquina virtual) y abrir una terminal. Lo primero que se debe hacer es actualizar repositorios y paquetes de `apt`. En Kali Linux, esto se hace de la siguiente manera (tal y como aparece en [20]):

```
sudo apt update

sudo apt full-upgrade -y
```

En Ubuntu, se pueden usar los siguientes comandos:

```
sudo apt update

sudo apt upgrade -y
```

9.4. Instalación de herramientas de SDR

Una vez preparado el sistema operativo, se procede a instalar las librerías y aplicaciones para trabajar con programas de *Software-Defined Radio* (SDR). Son las siguientes: librerías de HackRF y paquetes para `gqrx`, `gnuradio`, e `inspectrum`.

```
sudo apt install hackrf gqrx-sdr gnuradio inspectrum
```

A continuación, hay que conectar la tarjeta HackRF One al ordenador mediante el cable USB (si se utiliza una máquina virtual con VMware o Virtualbox, hay que asegurarse de que la tarjeta se conecta al sistema operativo *guest* y no al *host*).

Para comprobar que el sistema operativo detecta la tarjeta HackRF One, se puede utilizar el comando `hackrf_info`. Si sale información similar a la mostrada en la Fig. A.1, la tarjeta se reconoce correctamente.

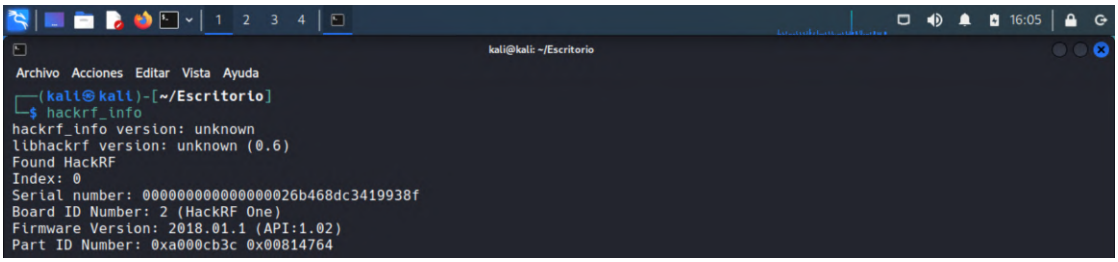

Fig. A.1. Instalación correcta de las librerías de HackRF

9.4.1. Instalación de `rfcat`

La instalación de `rfcat` puede ser algo más laboriosa y es posible que existan fallos por el cambio de versión de Python (2.7 a 3.x) u otras dependencias. Todo el código en Python desarrollado en este proyecto está en Python 3.x, ya que la versión 2.7 se encuentra en desuso y sin soporte.

Hoy en día, la mayoría de las distribuciones de Linux llevan Python instalado. Aun así, es conveniente instalar `pip`, el gestor de paquetes para Python y agregar a la variable de entorno `PATH` la ruta donde `pip` guardará binarios ejecutables:

```
sudo apt install python3-pip

echo 'export PATH=/home/$USER/.local/bin:$PATH' >> ~/.zshrc

source ~/.zshrc

pip install -U pip
```

Para Ubuntu, basta cambiar `.zshrc` por `.bashrc` (según el tipo de *shell*). Con `pip` se puede instalar la librería `rfcat`, pero es conveniente instalarlo desde el repositorio de GitHub (disponible en [14]). En primer lugar, hay que instalar dependencias:

```
sudo apt install python3-usb libusb-dev make
```

A continuación, es necesario instalar SDCC versión 3.5.0 (importante que sea esta versión). Este paquete no se puede instalar con un gestor como `apt`, porque ya no se encuentra dicha versión disponible. Hay que descargarlo del repositorio de Debian (https://packages.debian.org/stretch/sdcc y https://packages.debian.org/stretch/sdcc-libraries) y elegir la versión indicada para la arquitectura correspondiente (amd64 o arm64).

Una vez descargados ambos paquetes, basta con instalarlos con los siguientes comandos desde el directorio en el que se encuentran los archivos:

```
sudo dpkg -i sdcc-libraries_3.5.0+dfsg-2_all.deb

sudo dpkg -i sdcc_3.5.0+dfsg-2+b1_arm64.deb
```

En este punto ya se puede instalar `rfcat`. Para ello, se han de ejecutar los siguientes comandos:

```
git clone https://github.com/atlas0fd00m/rfcat.git

cd rfcat

sudo python3 setup.py install
```

Ahora solo falta agregar unas configuraciones de permisos para permitir que `rfcat` se pueda ejecutar sin permisos de super-usuario:

```
sudo cp etc/udev/rules.d/20-rfcat.rules /etc/udev/rules.d

sudo udevadm control --reload-rules
```

Y ya estaría todo listo. Con `rfcat -r` se puede entrar a una consola interactiva para controlar el dispositivo YARD Stick One:

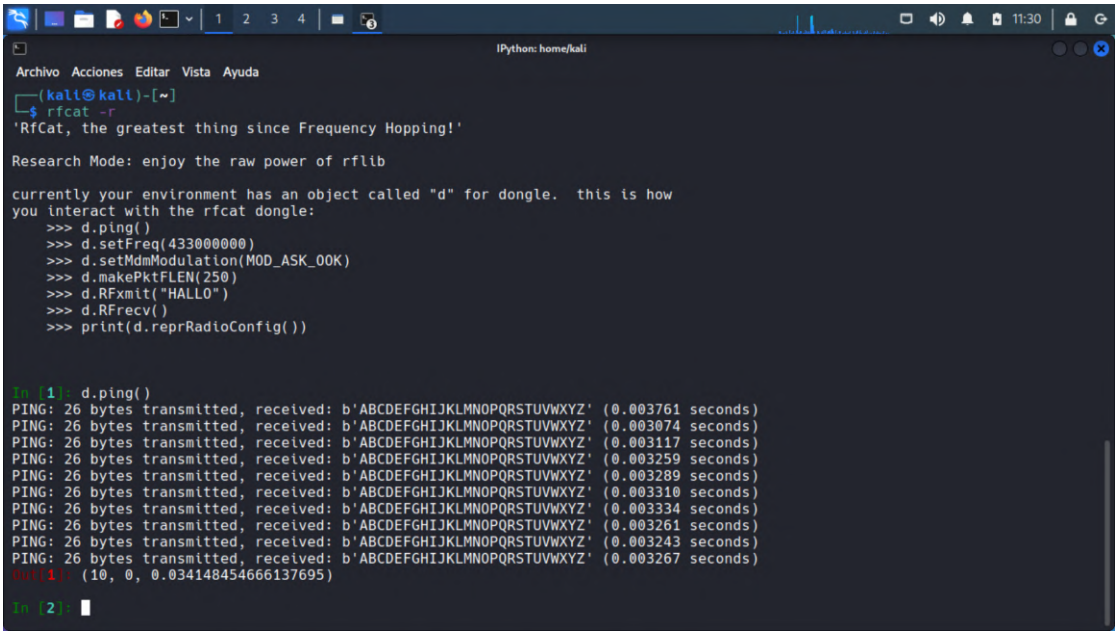

Fig. A.2. Instalación correcta de `rfcat`

Anexo B. Manual de usuario

B.1. Identificación de señales de radiofrecuencia con GQRX

Antes de realizar un ataque a un sistema de comunicaciones por radiofrecuencia, es necesario identificar la banda de trabajo del propio sistema. Para ello, se va a utilizar el programa GQRX.

Al introducir el comando `gqrx` en una terminal, aparecerá una interfaz gráfica. Es posible que aparezca un error de que no se reconoce la antena. En tal caso, seleccionar el dispositivo que tiene "HackRF" repetido en el nombre, como se muestra en la Fig. B.1.

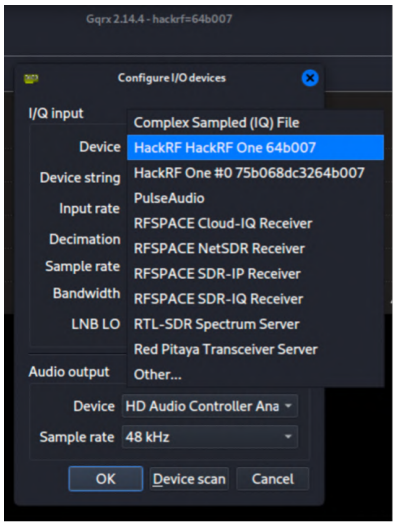

Fig. B.1. Elección de tarjeta HackRF One en GQRX

A continuación, se puede seleccionar la frecuencia de trabajo (en hercios) y darle al botón de "*Play*". En este punto, la antena está escuchando y se muestra el espectrograma de frecuencia detectado en modelo *waterfall* (ver Fig. B.2).

Si se acciona una llave de coche, se verá la señal en el espectrograma (siempre que se haya elegido bien la frecuencia):

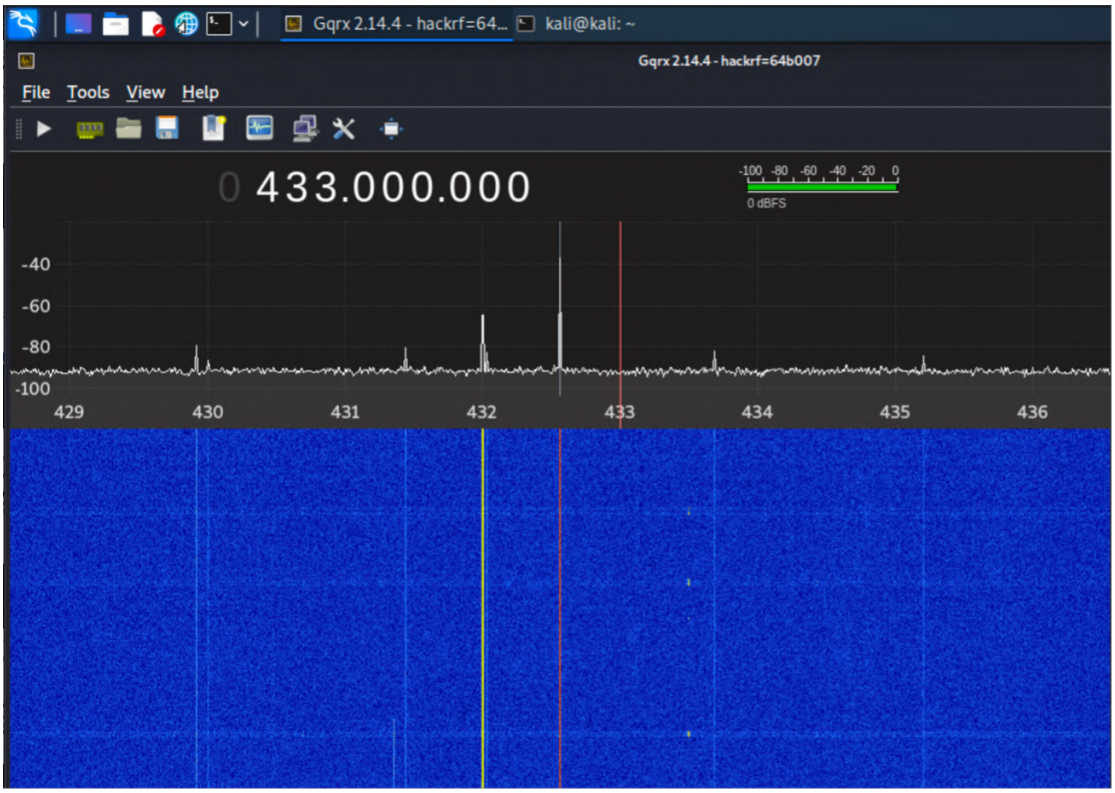

Fig. B.2. Espectrograma de GQRX en modelo *waterfall* (1)

Si no se ve ninguna señal, probar a cambiar la frecuencia de trabajo y volver a mirar. Se puede aumentar el nivel de detalle con la rueda del ratón sobre el eje de frecuencias, para ver la señal de manera más clara, como en la Fig. B.3.

Con esto, ya tendríamos identificada la frecuencia de trabajo. A continuación, hay que cerrar la ventana de GQRX para continuar con el siguiente apartado (si no se cierra, habrá problemas con el uso de la tarjeta HackRF One).

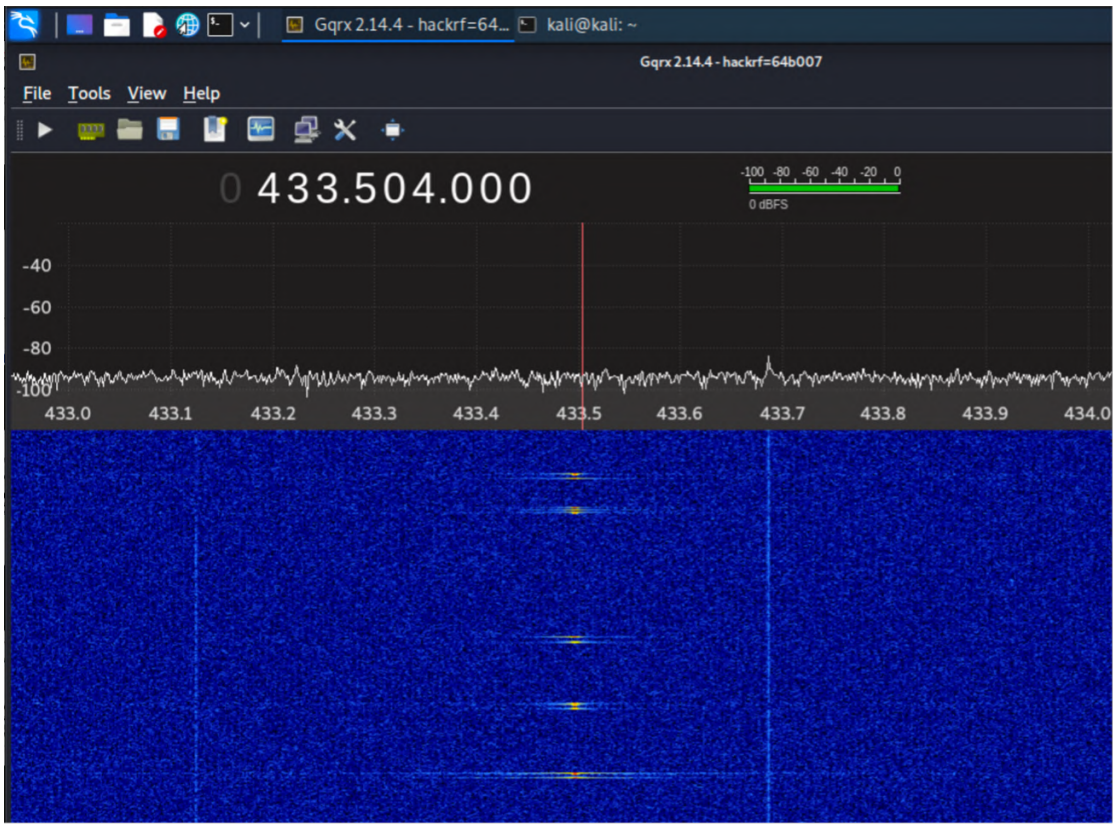

Fig. B.3. Espectrograma de GQRX en modelo *waterfall* (2)

B.2. Captura de señales con GNU Radio Companion

Se puede abrir GNU Radio Companion introduciendo el comando `gnuradio-companion` en una terminal.

Este programa es un entorno de programación gráfica que permite crear distintos objetos, configurarlos y conectarlos entre sí. Por detrás, el programa genera un código en Python que contiene la programación gráfica realizada.

Para capturar una señal, será necesario poner la tarjeta HackRF One en modo receptor y almacenar la captura en un archivo. Adicionalmente, se puede añadir una pantalla para visualizar el espectro en tiempo real.

Se necesitan añadir, por tanto, los siguientes objetos (utilizar el buscador):

- `osmocom Source`: Ajustar aquí la frecuencia de trabajo (propiedad llamada `Ch0: Frequency (Hz)`).
- `File Sink`: Indicar la ruta del archivo donde se va a guardar la captura.
- `QT GUI Frequency Sink`: En la pestaña `Config`, seleccionar `Yes` en la propiedad `Control Panel`.

Adicionalmente, poner un valor de `1.8e6` en la variable `samp_rate`. Los objetos se conectan como aparecen en la Fig. B.4.

A continuación, hay que guardar el proyecto y darle al botón de "*Play*". Aparecerá una gráfica para ver la potencia de las señales en la frecuencia indicada, como se ve en la Fig. B.5.

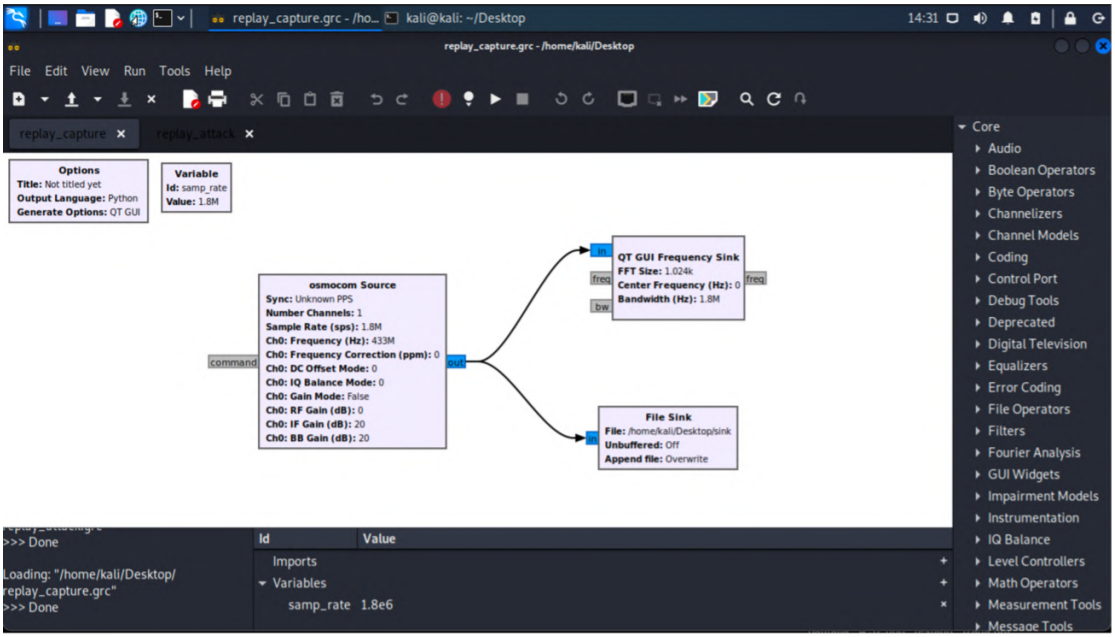

Fig. B.4. Proyecto para capturar señales en GNU Radio Companion

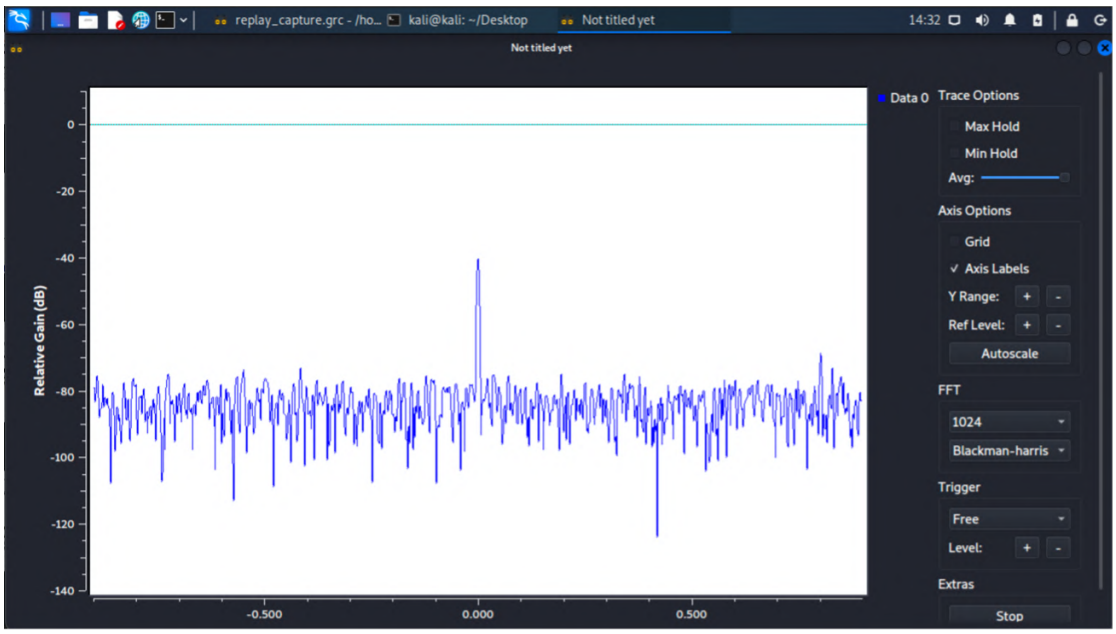

Fig. B.5. Vista del espectro en tiempo real

Es conveniente indicar en el panel de la derecha "*Max Hold*" para que se vea en color verde la ganancia de señal máxima detectada.

En este punto, el programa está grabando todo lo que ocurre en esa banda de frecuencia. Si se envía una señal, esta será capturada y se verá en el gráfico en color verde (ver Fig. B.6). Una vez grabada la señal, ya se puede cerrar la gráfica de frecuencia.

B.3. Repetición de señales (ataque de *replay*)

Ahora, hay que abrir un nuevo proyecto de GNU Radio Companion. En este caso, para reproducir una señal previamente grabada se necesita poner la antena en modo transmisor e indicar el archivo donde está la señal a transmitir. Adicionalmente, se puede añadir una pantalla para visualizar el espectro en tiempo real.

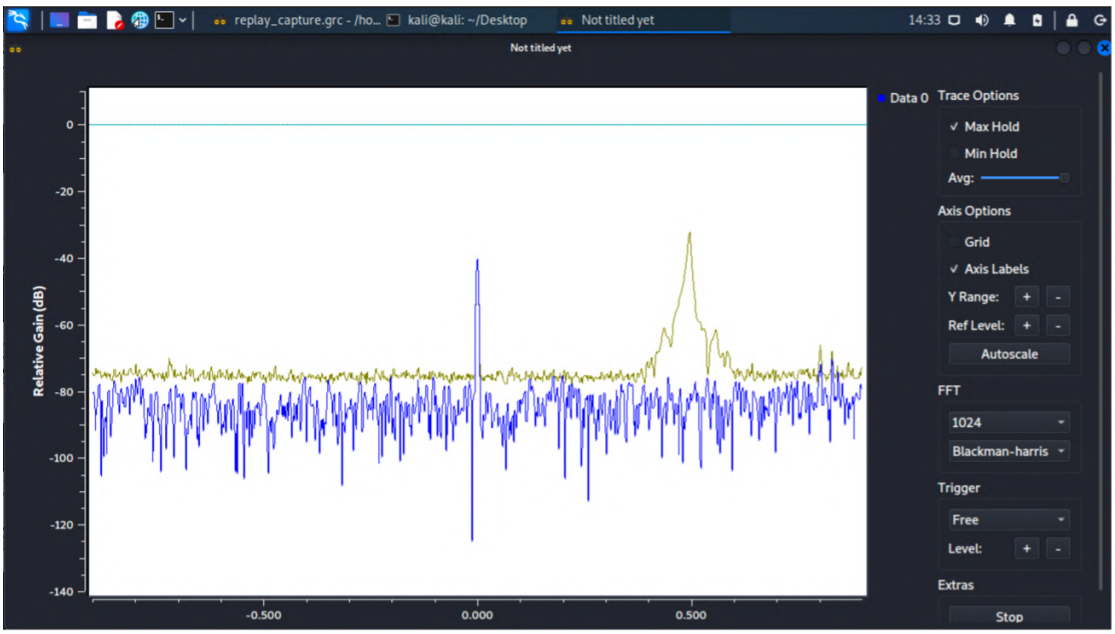

Fig. B.6. Vista del espectro en tiempo real con "*Max Hold*"

Se necesitan añadir, por tanto, los siguientes objetos (utilizar el buscador):

- `File Source`: Indicar la ruta del archivo en el que está la señal a enviar (se puede indicar `Yes` en la propiedad `Repeat` si una vez que termine la grabación, se repita indefinidamente).
- `osmocom Sink`: Ajustar aquí la frecuencia de trabajo (propiedad llamada `Ch0: Frequency (Hz)`).
- `QT GUI Frequency Sink`: En la pestaña `Config`, seleccionar `Yes` en la propiedad `Control Panel`.
- `Throttle`.

Los objetos se conectan como aparecen en la Fig. B.7. Al guardar y darle al botón de "*Play*", aparecerá un gráfico de frecuencia como el de antes. En este caso, se está reproduciendo la grabación guardada anteriormente. Si se activa "*Max Hold*" en el panel de la derecha, se verá claramente cómo se envía la señal capturada.

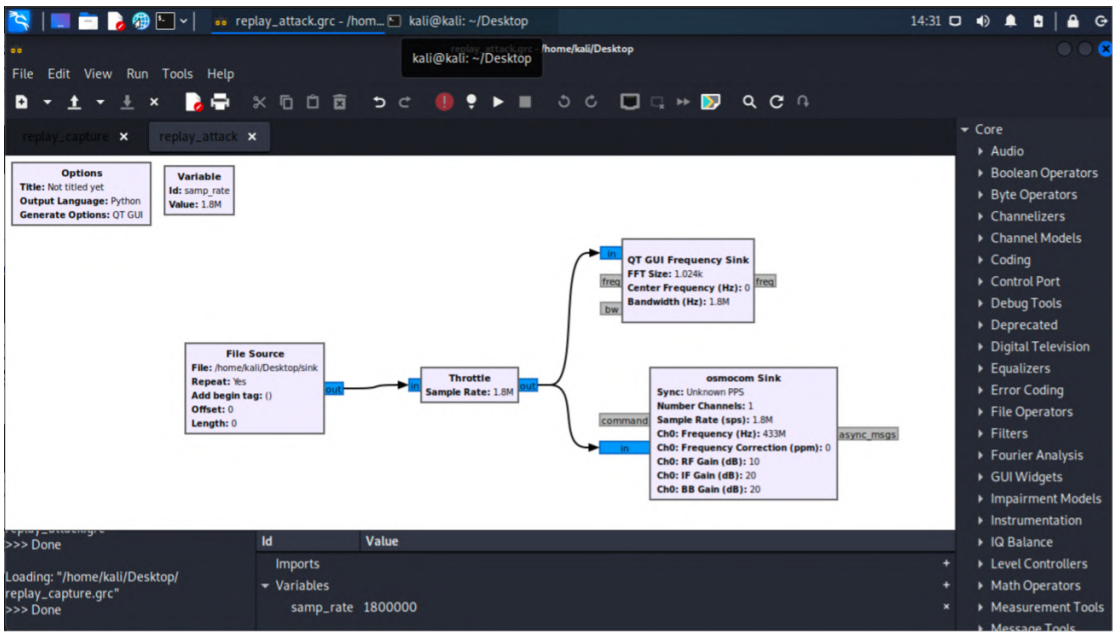

Fig. B.7. Proyecto para transmitir capturas de señales en GNU Radio Companion

B.4. Análisis de señales con `inspectrum`

Para iniciar `inspectrum`, es necesario usar el siguiente comando en una terminal:

```
inspectrum <filename>
```

Adicionalmente, se puede añadir un "`&`" al final para dejar el proceso en segundo plano y poder seguir usando la terminal. Tras unos segundos de procesamiento del archivo de captura, aparecerá la ventana de `inspectrum` con la vista de la señal en tiempo y frecuencia (ver Fig. B.8). Hay poner la misma tasa de muestreo (*sample rate*) usada en GNU Radio Companion a la hora de capturar la señal.

En este punto, es necesario usar la barra de desplazamiento horizontal para encontrar la señal en la captura (ver Fig. B.9).

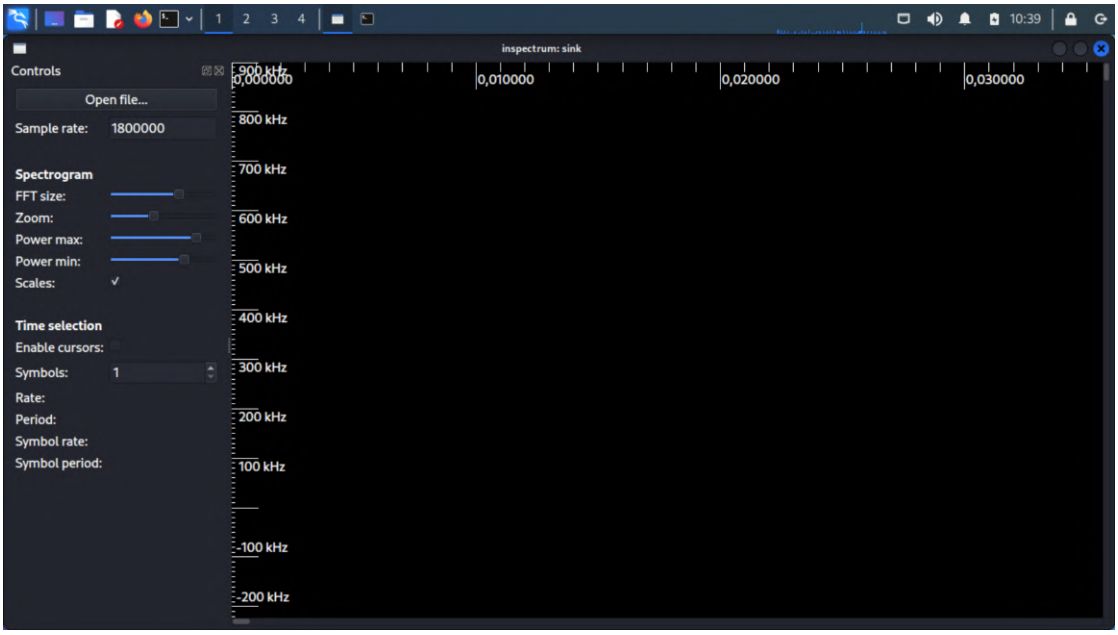

Fig. B.8. Inicialización de `inspectrum`

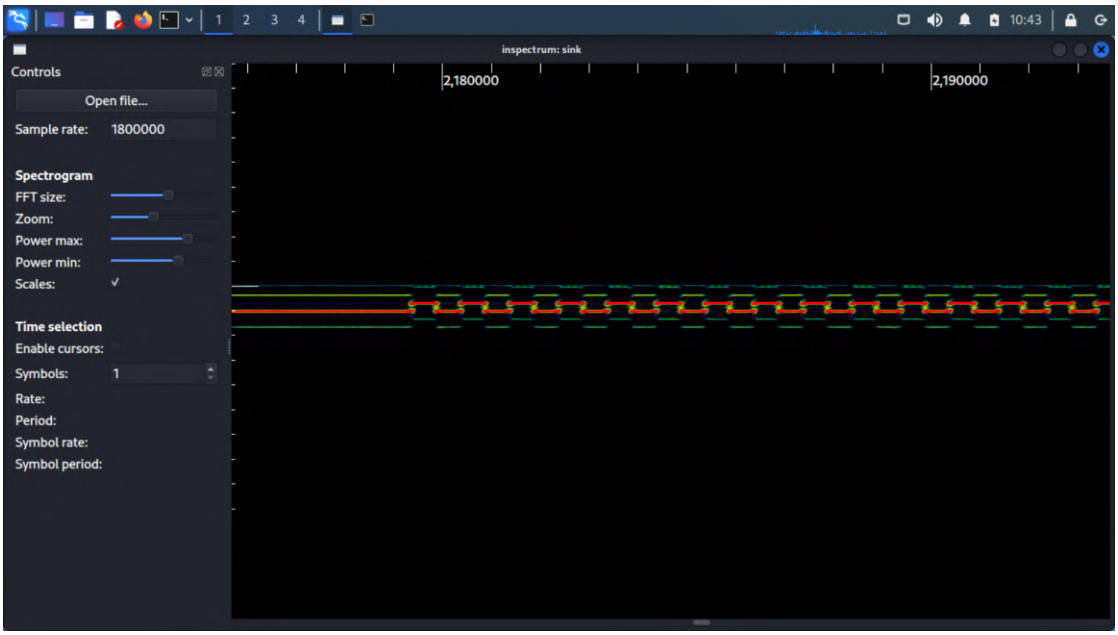

Fig. B.9. Búsqueda de la señal capturada en `inspectrum`

Para realizar el análisis de la señal, es necesario utilizar cursores. Para ello, hay que activar un *checkbox* titulado "*Enable cursors*" y desplazar los cursores hasta que coincidan aproximadamente con un símbolo de la señal:

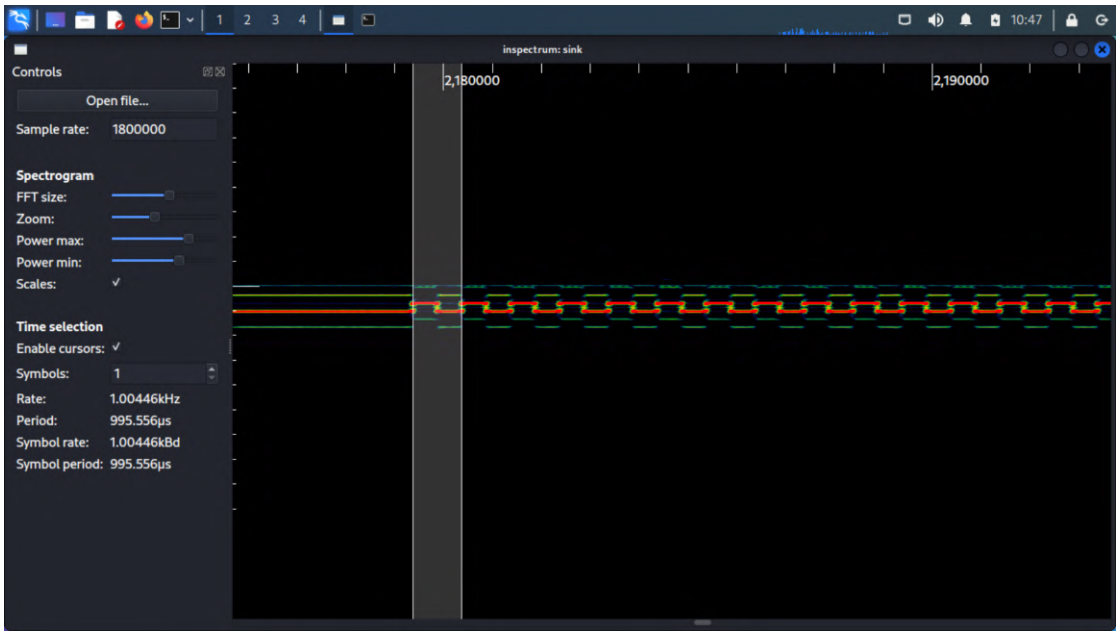

Fig. B.10. Uso de cursores en `inspectrum` (1)

En este caso, la señal utiliza modulación 2FSK con codificación Manchester. Ahora hay que ir aumentando el número de símbolos donde pone "*Symbols*". Al mismo tiempo que se van aumentando los símbolos, hay que ajustar los cursores para que los símbolos queden bien cuadrados, como se ve en la Fig. B.11.

Una vez puestos todos los cursores necesarios, ya se pueden extraer los bits de forma manual o automática. Cabe mencionar que `inspectrum` no siempre consigue extraer bits de las señales con modulación 2FSK.

Si la señal está modulada en ASK/OOK, los cursores se deben poner de manera que cojan los pulsos de señal, independientemente de si utilizan codificación Manchester u otro tipo de codificación (ver Fig. B.12).

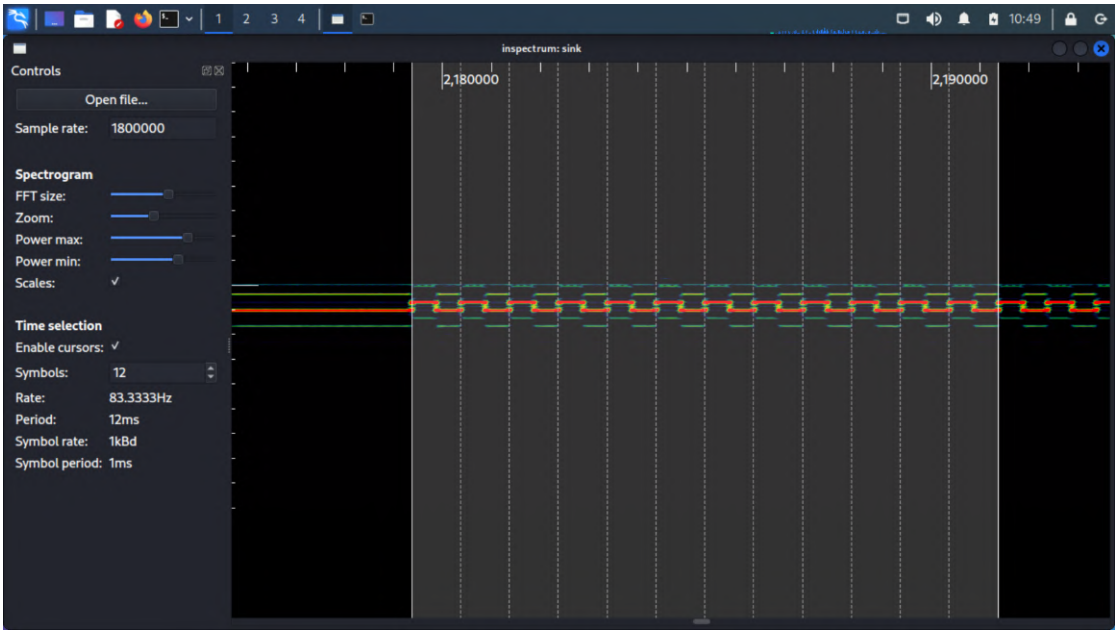

Fig. B.11. Uso de cursores en `inspectrum` (2)

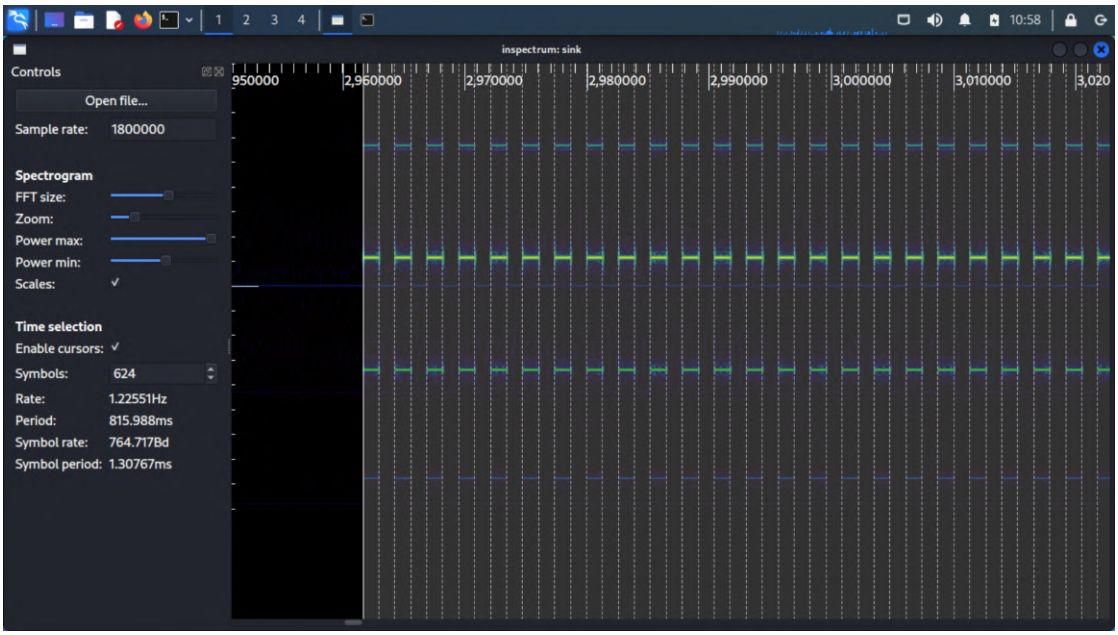

Fig. B.12. Uso de cursores en `inspectrum` (3)

En este punto, se puede usar clic derecho y seleccionar "*Add derived plot > Add amplitude plot*". Aparecerá un cursor horizontal que habrá que poner encima de la frecuencia fundamental de la señal, como se muestra en la Fig. B.13.

En la gráfica inferior aparecerá la representación de amplitud de la señal. De nuevo, habrá que añadir otro gráfico con clic derecho y seleccionar "*Add derived plot > Add threshold plot*" (ver Fig. B.14).

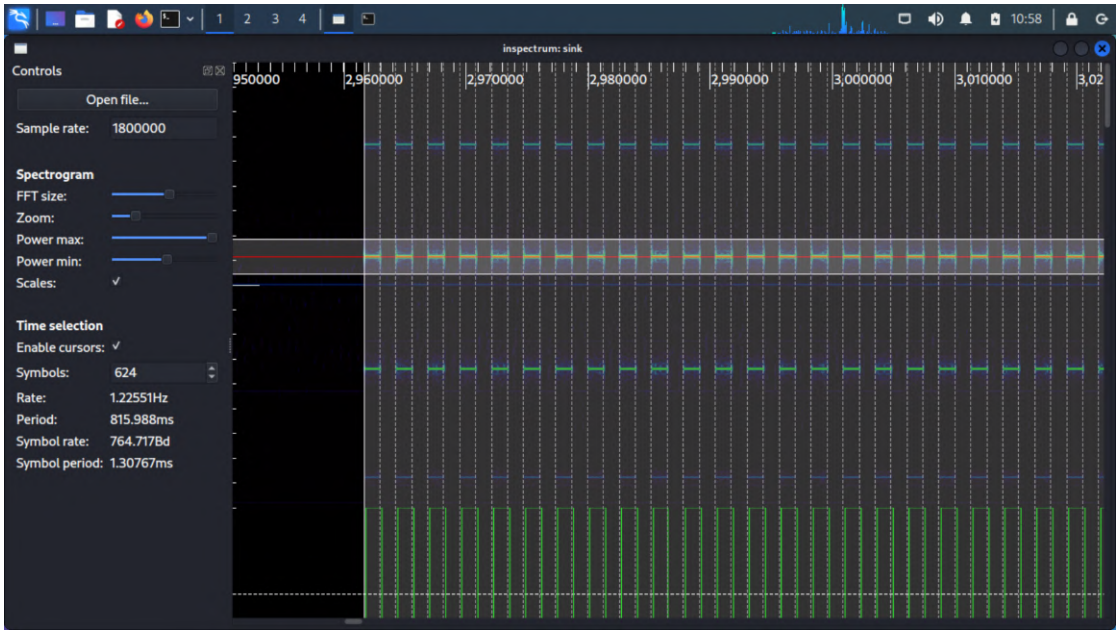

Fig. B.13. Extracción de símbolos de una señal con `inspectrum` (1)

Aparecerá un tercer gráfico como el de la Fig. B.14 en el que se podrá hacer clic derecho y seleccionar "*Extract symbols*". La opción de "*To stdout*" mostrará los bits en la consola donde se inició `inspectrum`, como se ve en la Fig. B.15; y la opción "*Copy to clipboard*" copiará los bits en el portapapeles.

Ahora hay que decodificar estos bits, porque la codificación usada en la señal es Manchester. Normalmente, el convenio es: `01 -> 0` y `10 -> 1`. Esto se puede hacer fácilmente con Python con un *script* sencillo o directamente en la consola con el siguiente "*one-liner*" (ver Fig. B.16):

```
python3 -c "print(hex(int(''.join('<bits>'.split(', ')[::2]), 2)))"
```

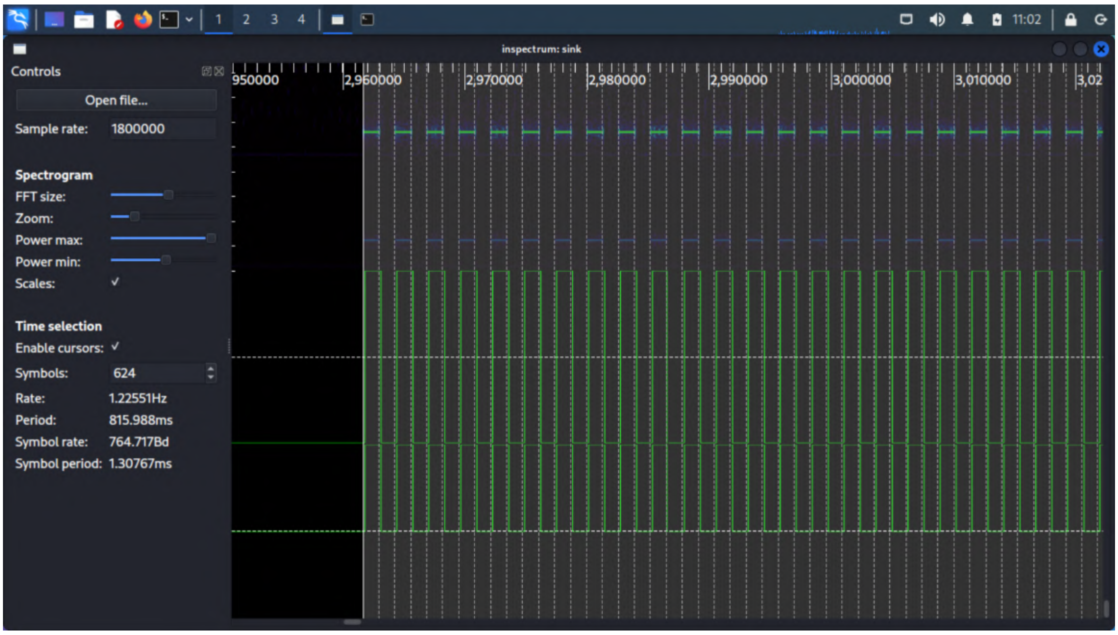

Fig. B.14. Extracción de símbolos de una señal con `inspectrum` (2)

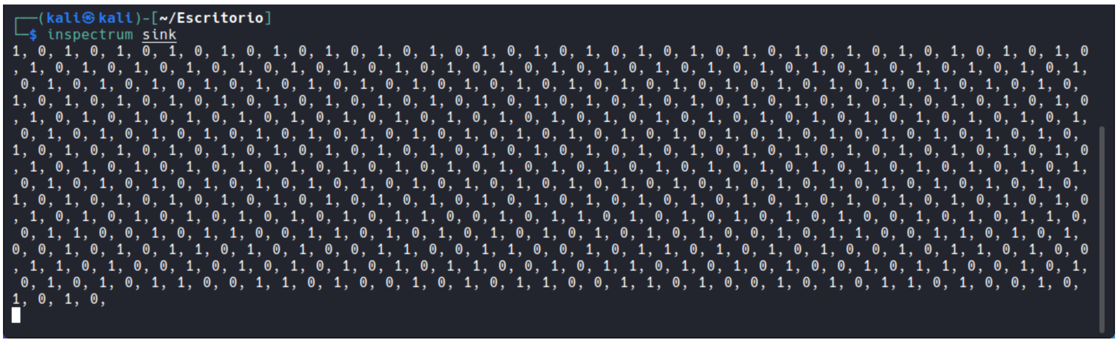

Fig. B.15. Extracción de símbolos de una señal con `inspectrum` (3)

Fig. B.16. Tratamiento de una secuencia de bits con codificación Manchester

B.5. Transmisión y recepción de señales con `rfcat`

Para usar `rfcat`, conviene empezar por su modo interactivo (`rfcat -r`), que está montado sobre `ipython3`. Esto permite tener autocompletado y búsqueda de funciones para hacer más ágil y sencillo el uso de `rfcat`.

El objeto fundamental de `rfcat` es la variable `d` (de *dongle*). Algunas configuraciones importantes son:

- `d.setFreq`: Sirve para ajustar la frecuencia de trabajo.
- `d.setMdmDRate`: Se utiliza para indicar la tasa de datos (*data rate*).
- `d.setMdmModulation`: Sirve para se indicar el tipo de modulación usada (típicamente `MOD_ASK_OOK` o `MOD_2FSK`).
- `d.setEnableMdmManchester`: Indica a `rfcat` si decodificar los símbolos con codificación Manchester o no.
- `d.makePktFLEN`: Configura el tamaño de la trama de símbolos en recepción.
- `d.setMdmSyncWord`: Utiliza 2 bytes para que `rfcat` sepa cuándo comienza una trama de símbolos.

Todos estos comandos se pueden utilizar directamente en la consola interactiva de `rfcat`. Para ver los parámetros necesarios de cada función, se puede usar `help` poniendo como argumento la función en cuestión (`d.setFreq`, por ejemplo).

Para usar `rfcat` en un *script* de Python, es necesario importar la librería `rflib` y crear el objeto `d` para reutilizar los comandos usados en la consola interactiva. A continuación ya se pueden poner las configuraciones necesarias para recibir o transmitir usando las funciones `d.RFrecv` y `d.RFxmit`, respectivamente.

```
#!/usr/bin/env python3

from rflib import RfCat

d = RfCat()
```

Código B.1. Código inicial para usar `rfcat` mediante un *script* de Python

B.6. Configuración de `rf_attack.py`

La herramienta desarrollada `rf_attack.py`, descrita en la Sección 5.2, sirve para recibir y transmitir señales en digital mediante YARD Stick One y `rfcat`.

Como se indica en la Sección 5.2.1, los parámetros de radiofrecuencia de cada tipo de señal se especifican mediante un archivo de configuración en formato JSON con las siguientes claves (un ejemplo de este se puede ver en el Código 5.1):

- `"MdmSyncSequence"`: En este campo hay que indicar la secuencia de sincronización hasta el bit que indica el comienzo de trama. En el caso de los vehículos de Marca A analizados, la secuencia de sincronización es `fffffffffffe`, es decir, una secuencia de 55 bits a `1` y el último bit a `0`.

- `"MdmSyncMode"`: Este valor indica si se tiene en cuenta la secuencia de sincronización o no. Lo más habitual es ponerlo a `1`.

- `"Freq"`: Aquí se especifica la frecuencia de trabajo, en hercios.

- `"PktPQT"`: Este campo sirve para añadir un umbral de calidad en la recepción del preámbulo (PQT, *Preamble Quality Threshold*). Un valor de `0` lo deshabilita, que es lo que hace `rfcat` por defecto.

- `"MdmDRate"`: Este campo contiene la tasa de datos en baudios. Cabe mencionar que, si la señal tiene codificación Manchester, es necesario duplicar esta tasa de datos (en el caso del coche de Marca A, se utilizan 1997 baudios, el doble de 998.9, ver Fig. B.17).

- `"PktFLEN"`: Este campo indica la longitud de la trama binaria en bytes. En el caso de las señales de los vehículos de Marca A, cada señal contiene 168 bits, es decir, 21 bytes.

- `"MaxPower"`: Se recomienda poner a `true` para transmitir.

- `"MdmModulation"`: En esta clave, se indica el tipo de modulación. Normalmente, será `"MOD_2FSK"` o `"MOD_ASK_OOK"`, aunque `rfcat` soporta otros tipos de modulaciones digitales.

- `"EnableMdmManchester"`: Hay que poner este campo a `true` si la señal está codificada en Manchester.

- `"Fields"`: En este atributo se puede indicar la estructura de la trama para que, cuando se reciba una señal, esta pueda ser analizada en mayor medida y se pueda descomponer en distintos campos (por ejemplo: secuencia de sincronización, comando, identificador de llave o código evolutivo). La manera de indicar esta estructura es mediante otro objeto JSON en el que se añade el nombre asignado al campo y su posición en la trama. Por ejemplo, la secuencia que aparece en `"MdmSyncSequence"` consta de 7 bytes, por lo que su posición en la trama se indicará como `[0, 7]`.

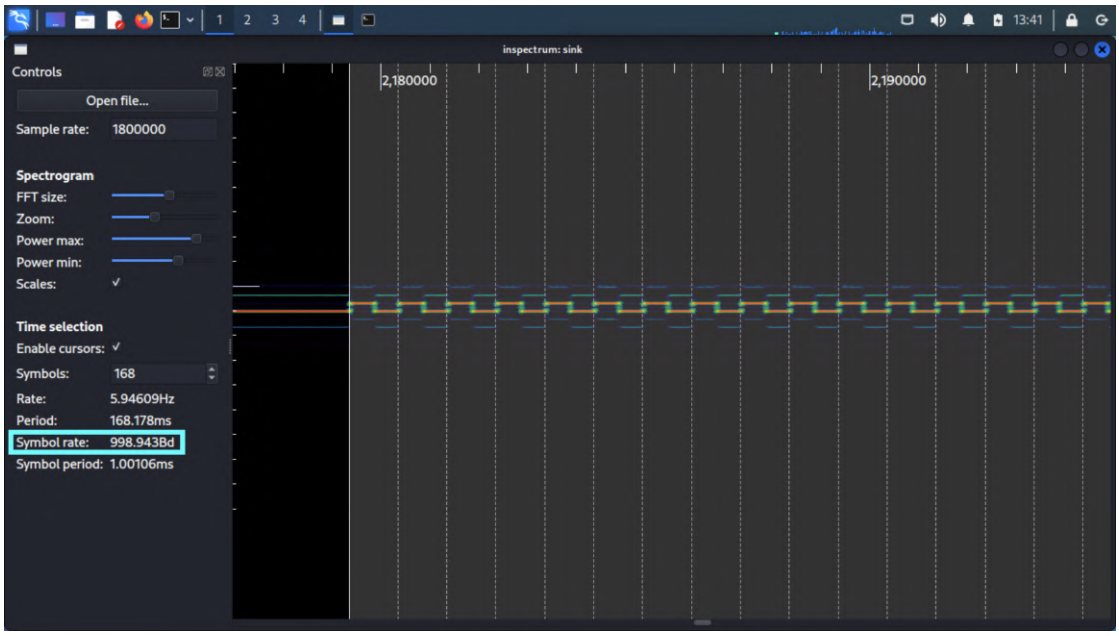

Fig. B.17. Tasa de datos para señales codificadas en Manchester

Si se utiliza el parámetro `-o/--output` para guardar el contenido de las señales capturadas, luego se pueden recuperar con el comando `xxd`, indicando la longitud de la trama como en la Fig. B.18, con señales de Marca A, de 21 bytes (168 bits).

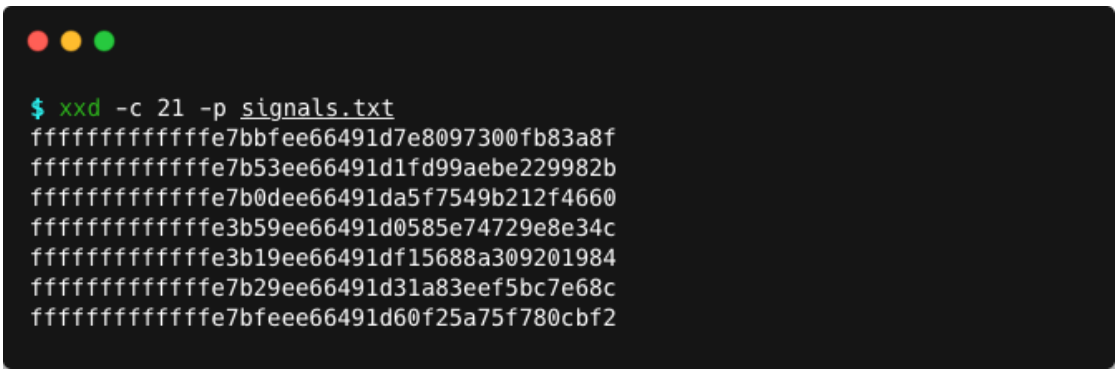

Fig. B.18. Lectura de señales almacenadas en archivo de salida

Anexo C. Objetivos de Desarrollo Sostenible

En este capítulo se muestra la relación que tiene el presente Trabajo Fin de Máster con los Objetivos de Desarrollo Sostenible (ODS) aprobados por la ONU en 2015, en la denominada Agenda 2030 sobre el Desarrollo Sostenible [21].

Cada uno de estos 17 objetivos (ver Fig. C.1) consiste en un conjunto de metas específicas que deben alcanzarse en los próximos 8 años (hasta el año 2030). La adopción de estos objetivos por parte de los líderes mundiales tiene el fin de erradicar la pobreza, proteger el planeta y asegurar la prosperidad.

Tal y como se dice en [22], "la sostenibilidad es el desarrollo que satisface las necesidades del presente sin comprometer la capacidad de las futuras generaciones, garantizando el equilibrio entre el crecimiento económico, el cuidado del medio ambiente y el bienestar social".

Los 17 Objetivos de Desarrollo Sostenible persiguen este concepto de sostenibilidad. Estos ODS se pueden clasificar según la dimensión con la que se relacionan. Así, existe una dimensión económica, una dimensión social y una dimensión medioambiental (referida a veces con el término biosfera). En la Fig. C.2 se muestran estas tres dimensiones y los ODS que intervienen en cada una.

Puede parecer que este concepto de sostenibilidad no tenga relación con el desarrollo de *software*, la ciberseguridad, los vehículos y las nuevas tecnologías. Sin embargo, sí que existe una pequeña relación. Por este motivo, el proyecto realizado en este Trabajo Fin de Máster persigue algunos de los 17 Objetivos de Desarrollo Sostenible. En la Tabla C.1 se muestra el ODS de cada dimensión que más relación tiene con el proyecto desarrollado, su nivel de importancia, y las metas específicas que más se ajustan.

Fig. C.1. Objetivos de Desarrollo Sostenible

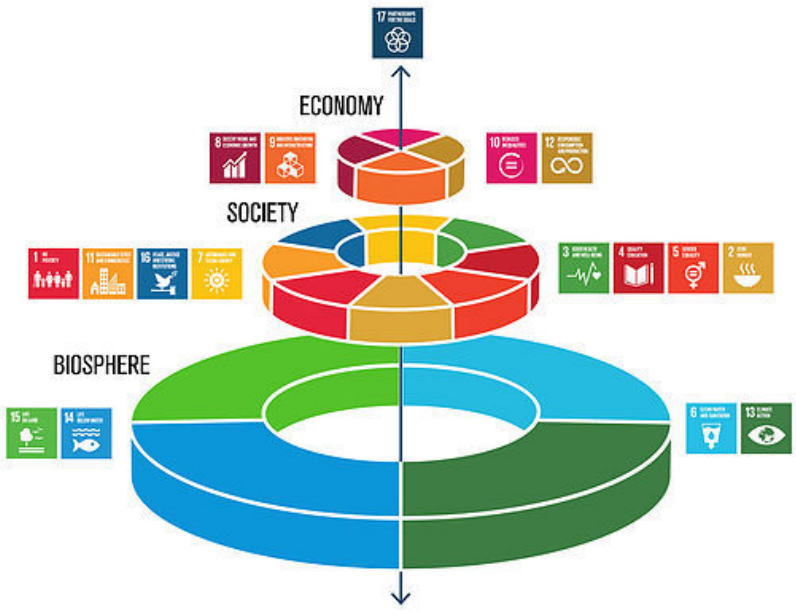

Fig. C.2. Dimensiones de los ODS

Tabla C.1. Objetivos de Desarrollo Sostenible relacionados con el proyecto [21]

Dimensión	ODS	Rol	Meta
Biosfera[1]	-	-	-
Sociedad	**ODS11.** Ciudades y comunidades sostenibles	Secundario	**11.2.** De aquí a 2030, proporcionar acceso a sistemas de transporte seguros, asequibles, accesibles y sostenibles para todos y mejorar la seguridad vial, en particular mediante la ampliación del transporte público, prestando especial atención a las necesidades de las personas en situación de vulnerabilidad, las mujeres, los niños, las personas con discapacidad y las personas de edad
Economía	**ODS9.** Industria, innovación e infraestructura	Primario	**9.1.** Desarrollar infraestructuras fiables, sostenibles, resilientes y de calidad, incluidas infraestructuras regionales y transfronterizas, para apoyar el desarrollo económico y el bienestar humano, haciendo especial hincapié en el acceso asequible y equitativo para todos

[1]No ha sido posible encontrar un Objetivo de Desarrollo Sostenible relacionado con el medio ambiente que se ajuste al presente proyecto.

ANEXO C. OBJETIVOS DE DESARROLLO SOSTENIBLE

Se ha considerado que el ODS9 es el que más relación tiene con el proyecto desarrollado. Esto se debe a que se trata de un proyecto de innovación que busca analizar el comportamiento de las tecnologías existentes en los vehículos para intentar mejorarlas y hacerlas más seguras en el futuro. Con la salida de la nueva normativa de ciberseguridad aplicada a vehículos [8], los fabricantes y las empresas de homologación tendrán que verificar que los nuevos vehículos que vayan a salir al mercado sean ciertamente seguros empleando técnicas como las presentadas en este Trabajo Fin de Máster.

Por otro lado, el ODS11 se podría relacionar con el presente proyecto. Debido a la investigación realizada sobre los mecanismos utilizados en vehículos actuales, será posible mejorar los de vehículos futuros para que sean más seguros. Con ello, se reducirá el número de robos de vehículos de particulares e información sensible, lo cual llevará a una comunidad más sostenible.

Por último, es interesante mencionar algunos aspectos relacionados con el futuro de los automóviles y la seguridad en vehículos. Los objetivos son alcanzar 0 muertos en accidentes de tráfico en el año 2050 y reducir los heridos graves en 2030 [23].

En definitiva, aunque *a priori* los proyectos de ciberseguridad y tecnología no parezcan tener relación con los Objetivos de Desarrollo Sostenible, al analizar en detalle los beneficios que producen, se ve más claro que sí están relacionados. Con esto, se consigue aportar valor a la sostenibilidad perseguida por la ONU.