ctOS-TPMS

Por
Jorge María Martín, Alberto Rodríguez Fuentes y Martín García Fuentes

UNIVERSIDAD COMPLUTENSE
MADRID

Grado en Ingeniería Informática
Facultad de Informática

Dirigido por
José Luis Vázquez Poletti y Juan Carlos Fabero Jiménez

ctOS-TPMS

Madrid, 2020-2021

ctOS-TPMS

Sistema de detención remota de vehículos a través del protocolo de monitorización de presión de los neumáticos

Memoria que se presenta para el Trabajo de Fin de Grado

Jorge María Martín, Alberto Rodríguez Fuentes y Martín García Fuentes

Dirigido por

José Luis Vázquez Poletti y Juan Carlos Fabero Jiménez

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

Madrid, 2020-2021

Agradecimientos

En primer lugar, los miembros de este Trabajo de Fin de Grado queremos agradecer a nuestros directores José Luis Vázquez Poletti y Juan Carlos Fabero Jiménez por su implicación en el desarrollo de este trabajo, brindándonos la oportunidad de participar en este proyecto, además de proporcionarnos toda la ayuda que hemos podido necesitar así como orientarnos y guiarnos en el descubrimiento de tecnologías que desconocíamos.

También queremos agradecer a todos nuestros familiares y allegados, quienes nos apoyaron desde el inicio del proyecto, apoyo sin el cual no hubiera sido posible la realización de este trabajo tan importante para nuestro desarrollo profesional como personal.

Finalmente queremos agradecer a dos compañeros de la universidad. Por un lado a David Pacios Izquierdo, por ofrecernos su ayuda cuando lo necesitáramos a la hora de realizar esta memoria en LaTeX, ya que nunca antes habíamos utilizado este lenguaje. Por otro lado, también agradecer a Alberto Caballero Gámez, por facilitarnos su trabajo realizado el año anterior que nos sirvió de base desde donde partir.

Resumen

Hoy en día los elementos de seguridad de los vehículos se han convertido en una pieza clave a la hora de evitar accidentes y de reducir las posibles lesiones que estos puedan producir. De entre todos los sistemas de seguridad existentes, este proyecto se enfoca en el TPMS, el sistema de monitorización de presión de los neumáticos obligatorio desde 2014 en todos los vehículos nuevos de la categoría M1, que proporciona información sobre el estado de los neumáticos y se comunica con el sistema central del vehículo mediante señales de radio, emitiendo este una serie de alertas en función de los datos de temperatura y presión que reciba. Ahora bien, estas señales se emiten sin cifrar, por lo que con las herramientas apropiadas es posible captar dicha señal y replicarla con datos falsos con el fin de confundir al vehículo. Para generar esta señal falsa hay que seguir varios pasos, empezando por la construcción de la trama de datos que se quiere enviar, continuando por la codificación de dicha trama (en este caso Manchester o Manchester diferencial) con el fin de provocar transiciones que permitan la sincronía de reloj o sincronía de bit, además de minimizar los posibles errores de transmisión, y por último la generación de la señal mediante la modulación de los datos digitales, lo que permite el envío de información a través de un medio analógico. Sin embargo, antes de generar la señal hay que demodular la señal recibida para obtener algunos datos importantes. Teniendo esto en cuenta, a lo largo del proyecto se explica el proceso para conseguir replicar dicha señal con éxito.

Palabras clave

- TPMS (Tire Pressure Monitoring System)
- RTL 433
- Codificación Manchester o Manchester diferencial
- Modulación FSK (Frequency Shift Keying) o ASK (Amplitude Shift Keying)
- Radiofrecuencia
- Telecomunicación
- Demodulación y modulación de señales

Abstract

Nowadays the safety elements of vehicles have become a key part of the time to reduce the number of accidents and to reduce the possible injuries that they could produce. Among all the existing security systems, this project is focused on the TPMS, the tire pressure monitoring system which is required in all new vehicles since 2014 whose category is M1. This system provides information about the state of the tires and connects with the main system of the vehicle by radio signals, which send different types of alarms based on the data of the temperature and the pressure that have been received. Having said that, these signals are sending without any encoding, so with the appropriate tools, it is possible to receive this signal and replicate it with some false data to confuse the vehicle. The process to generate the false signal is based on a series of steps. First of all, it is necessary to build the data frame that we want to send, the process continues with the coding of this frame (in this case Manchester or Differential Manchester) to reproduce the transitions that allow the clock synchrony or the bit synchrony and to reduce the possible transmission errors. The last step is the generation of the signal by the way of the modulation of the digital data, this allows us to send the information in an analog way. Nevertheless, before the generation of the signal, it is necessary to demodulate the received signal to obtain some important data. Knowing all of this, this project explains the process to successfully replicate the signal.

Key word

- TPMS (Tire Pressure Monitoring System)
- RTL 433
- Manchester or differential Manchester encoding
- FSK (Frequency Shift Keying) or ASK (Amplitude Shift Keying) Modulation
- Radiofrequency
- Telecommunication
- Signal demodulation/modulation

Índice general

Índice de figuras

Índice de cuadros

Siglas

ABS Antiblockiersystem. 6, 7

AEC Automotive Electronics Council. 8

ASIL Automotive Safety Integrity Level. 9

ASK Amplitude Shift Keying. 12

CLI Command Line Interface. 28

CRC Cyclic Redundancy Check. 52, 61

CSV Comma-Separated Values. 20

DMR Radio Móvil Digital. 14

ECU Engine Control Unit. 8, 15, 35, 39, 43, 49, 50, 58, 59

ESP Elektronisches Stabilitätsprogramm. 6, 7

FSK Frequecy Shift Keying. 12, 31

GPIO General Purpose Input/Output. 25–28

GTK Gnome Graphical Toolkit. 30, 32

JSON JavaScript Object Notation. 20

NHTSA National Highway Traffic Safety Administration. 7, 48, 57

PC Personal Computer. 18

PCB Printed Circuit Board. 19, 26

RACE Real Automóvil Club de España. 7

RF Radiofrecuencia. 9, 10, 19, 21, 39

SDR Software Defined Radio. 18

SPI Serial Peripheral Interface. 26, 28, 31, 38, 42, 46

TETRA Terrestrial Trunked Radio. 14

TFG Trabajo de Fin de Grado. 16, 20, 31, 32

TPMS Tire Pressure Monitoring System. 7–9, 15, 16, 18, 20, 21, 25, 47, 48, 50–53, 56–61, 72

URH Universal Radio Hacker. 39, 43, 46

USB Universal Serial Bus. 18, 26, 27

Capítulo 1

Introducción

Este Trabajo de Fin de Grado fue planteado como la continuación del trabajo realizado por un compañero de la facultad, Alberto Caballero Gámez. En su trabajo, Alberto ofrece un estudio previo de la tecnología utilizada para la comunicación de los sistemas TPMS, así como un estudio de la propia comunicación, realizando un exhaustivo análisis del formato de los mensajes que transmiten distintos protocolos.

Esta continuación se centrará en la realización de una demostración práctica, verificando así que el marco teórico desarrollado por Alberto es cierto. Se realizará una implementación de toda la comunicación, tanto de la parte de recepción y decodificación de los mensajes enviados por los sistemas TPMS, como de la parte de la codificación de los mensajes y el envío de los mismos.

Para empezar, el proyecto se centrará en desarrollar dos protocolos que Alberto había documentado y estudiado en su Trabajo de Fin de Grado. Posteriormente, y en caso de que el tiempo lo permita, se continuará ampliando el número de protocolos desarrollados con el objetivo de mostrar que esto se puede aplicar a cualquier protocolo y que supone un riesgo real.

1.1. Problema

La idea principal de este TFG es demostrar que existe un riesgo real ante la posibilidad de explotar la vulnerabilidad existente en los sensores TPMS instalados en los vehículos, los cuales transmiten estos mensajes sin ningún tipo de cifrado y pueden ser suplantados sin una dificultad significativa.

Para ello se ha realizado un análisis exhaustivo de las tramas de cada protocolo para poder replicarlas modificando aquellos datos que nos sean de interés, siempre manteniendo las adecuadas comprobaciones de detección de errores para garantizar así que la trama modificada es correctamente decodificada por el vehículo en cuestión.

1.2. Objetivos

El objetivo de este TFG se puede desglosar en dos partes bien diferenciadas. Una primera parte que englobaría la recepción y decodificación de la señal emitida por los

sensores TPMS de los vehículos, y una segunda parte que consistiría en la codificación y transmisión de la señal simulada, además de conseguir que sea decodificada por el vehículo.

En la primera parte se hace uso de un dispositivo especializado en la recepción de señales de radio (RTL-SDR) y del software RTL-433 para recibir y decodificar las señales encontradas.

En la segunda parte se desarrolla un programa mediante el cual una vez se recibe y decodifica una señal captada, es posible replicar esta señal adulterando los valores que se deseen, consiguiendo así que el vehículo sea capaz de decodificarla. De esta forma se puede demostrar el riesgo que existe asociado a esta vulnerabilidad.

1.3. Estructura de la memoria

En esta sección se puede encontrar una breve explicación sobre cada capítulo que aparece en esta memoria:

- **Introducción:** En este primer capítulo se describe una introducción del proyecto desarrollado y se expone el principal problema que se va a tratar en la memoria.
- **Introduction:** Incluye el mismo contenido que la introducción traducido al inglés.
- **Estado del arte:** Se detalla la situación con la que nos encontramos, previo al desarrollo del proyecto. Se explican soluciones parecidas a las nuestras y las limitaciones que estas poseen, además de explicar en detalle la base de la que se parte para la realización de este proyecto.
- **Arquitectura del sistema:** Este es el capítulo central de la memoria. Contiene la explicación de todo el desarrollo del proyecto. También se explica detalladamente cada uno de los módulos que componen el sistema, así como las tecnologías utilizadas en cada uno de ellos.
- **Validación del sistema:** En este punto se expone toda la fase de pruebas que se realiza para comprobar el correcto funcionamiento del proyecto y así demostrar los objetivos planteados. Además se expone un caso de uso por cada protocolo implementado, en el cual se explica en detalle el paso a paso de la ejecución, demostrando así que se obtienen los resultados esperados.
- **Discusión, conclusiones y trabajo futuro:** En esta parte se expone una reflexión final del desarrollo del proyecto, así como las conclusiones sacadas durante el mismo. Además se valora la consecución de los objetivos planteados y se detallan posibles formas de extender el trabajo realizado.
- **Discussion, conclusions and future work:** Este capítulo es una traducción al inglés del capítulo anterior.
- **Distribución del trabajo:** En este apartado se indica de manera breve como se ha distribuido el trabajo entre los distintos miembros del equipo, indicando en que partes han tenido más incidencia cada uno de ellos. Además, se da una explicación de la metodología de trabajo empleada para desarrollar este TFG.

- **Bibliografía y enlaces de referencia:** En este capítulo se incluyen todas las referencias bibliográficas usadas durante el desarrollo de la memoria, así como ciertos enlaces utilizados para obtener información o enlaces que pueden contener información relevante que complemente lo desarrollado en esta memoria.

Capítulo 2

Introduction

This End-of-Degree Project was proposed as the continuation of the project that was realized by our mate, Alberto Caballero Gámez. In his work, Alberto offers previous research about the technology which is used for the communication between TMPS systems. Besides his work contains research about the communication in which analyzed the format of the messages that transmit the different protocols.

This continuation will focus on the development of a practical demonstration, verifying by this way that the theoretical studio developed by Alberto is true. The implementation of the communication will be realized including the reception and the decoding of the sent TPMS signals, the part of the messages codifications and the signal sending out.

In the beginning, this project will focus on developing the protocols that Alberto has researched in his End-of-Degree Project. Afterwards, the project will be increasing the number of developed protocols to demonstrate that it is possible to develop whatever protocol and demonstrate that mean a real risk.

2.1. Problem

The main idea of this TFG is to demonstrate that exists a real risk in the possibility of taking advantage of a vulnerability in the TPMS sensors which are installed in the vehicles. These sensors transmit the messages without any type of encoding so they can be forged without a great difficulty.

Therefore, a thorough analysis has been carried out about the frame of any protocol to be able to replicate the frame modifying those data in which are interested. When we modify any data is necessary to maintain the appropriate error detection verifications to guarantee that the modified frame is correctly decoded for the vehicle.

2.2. Goals

The purpose of this TFG can be broken down into two different sections. The first part comprehends the reception and the decoding of the sent out signals by the vehicles TPMS sensors. In the other part, the signal will be encoded and transmitted, furthermore, the vehicle must detect this simulated signal.

To complete the first part will use a specialized device to receive Radio signals (RTL-SDR) and the RTL-433 software to decode the received signals.

The second part will develop software that receives and decode a captured signal, it is possible to replicate that signal and modify the desired values, attaining that the vehicle can decode the generated signal. In this way, it is possible to demonstrate that exists a tangible danger associate with this vulnerability.

2.3. Document structure

In this section you can find a brief explanation of each chapter that appears in this document:

- **Introducción:** This first chapter describes the introduction of the developed project and explains the main problem to be dealt with in the document.
- **Introduction:** Includes the same content that "Introducción" translated into English.
- **Estado del arte:** Details the situation in which we are, before project development. Solutions similar to ours and their limitations are explained, in addition to explaining in detail the basis from which it is started for the development of this project.
- **Arquitectura del sistema:** This is the main chapter of the document and contains an explanation of the whole development of the project. Each of the modules that make up the system is also explained in detail just us the technologies used in each of them.
- **Validación del sistema:** In this chapter, the entire testing phase is exposed to verify the correct operation and in this way demonstrate the established objectives. Also, a use case is exposed for each implemented protocol, in which the step by step of execution is explained in detail, demonstrating in this way the expected results are obtained.
- **Discusión, conclusiones y trabajo futuro:** This part exposes a final reflection of the project's development and its final conclusions. Besides, the achievement of the objectives set is valued and some ways of expanding the work carried out are detailed.
- **Discussion, conclusions and future work:** This chapter is an English translation of the previous chapter.
- **Distribución del trabajo:** In this section it is briefly explained the way that how was distributed the work between all the team, pointing out in what parts work with more emphasise. In the final, there is an explanation about the work's methodology used to develop this TFG.
- **Bibliografía y enlaces de referencia:** This chapter includes all the bibliographic references used during the process of developing this document. Besides includes some links that were used to obtain information or links that could contain important information that complements the information of this document.

Capítulo 3

Estado del arte

3.1. Sistemas de seguridad

A día de hoy los elementos de seguridad de los vehículos se han convertido en una pieza clave a la hora de evitar accidentes y a la hora de reducir las posibles lesiones que estos puedan producir.

El primer elemento de seguridad reseñable fue la luna delantera laminada, con Henry Ford como pionero en su uso entorno al 1926. Sustituyó las tradicionales lunas de la época, las cuales en caso de colisión se astillaban en trozos altamente cortantes que en muchas ocasiones causaban lesiones más severas que las provocadas por la propia colisión del automóvil.

Con respecto a los elementos de seguridad propios de la carrocería, es importante destacar que Mercedes, entorno al año 1952, comenzó a producir coches equipados con zonas de deformación programada. Estas zonas absorben la mayor cantidad posible de energía producida en la colisión, evitando así que esta energía recaiga en los ocupantes del vehículo pudiendo ocasionarles lesiones mayores.

Fue ya en 1959, cuando Nils Bohlin inventó el cinturón de seguridad de tres puntos de anclaje, hecho que supuso un punto de inflexión en la historia de la seguridad en los automóviles ya que desde entonces ha permitido salvar la vida de más de un millón de personas cada década.

Seguidamente, y también en la década de los 50, fue el ingeniero americano John Hetrick quien inventó el primer airbag. Este primer diseño no tenía el resultado esperado por lo que fue necesario que Ford, en 1968 con la inestimable colaboración de la universidad de Yale, desarrollase el Auto-Ceptor. Sin embargo no fue hasta 1971 cuando Mercedes-Benz consiguió patentar el sistema que se utiliza hoy en día, el cual ha ido evolucionando a lo largo de los años.

A medida que avanzaron los años se siguieron introduciendo nuevos sistemas de seguridad, como por ejemplo los frenos Antiblockiersystem (ABS), desarrollados por Bosch en 1978. Pocos años después, y también de la mano de Bosch en 1986, nacieron los primeros sistemas de control de tracción. Sería casi una década después, cuando en 1995 Bosch junto con Mercedes desarrollase el control de estabilidad o Elektronisches Stabilitätsprogramm (ESP), hecho que supuso una auténtica revolución, y que ha pasado a

ser obligatorio en todos los turismos nuevos que se comercialicen desde de noviembre de 2014.

Sin embargo, también existen algunas regulaciones contractuales como puede ser el caso de Latinoamérica. Siguiendo esta norma, los sensores Tire Pressure Monitoring System (TPMS) son considerados como un accesorio por lo que no son reconocidos como obligatorios. Este hecho puede afectar a vehículos de alta gama importados de Estados Unidos y Europa que no cuenten con este tipo de sistemas en caso de que hayan sido fabricados siguiendo este tipo de normativas.

Todo este proceso de innovación y mejoras en los sistemas de seguridad de los automóviles nos lleva al punto actual en el que nos encontramos. Con la llegada del siglo XXI y los innumerables avances de la tecnología comienzan a incorporarse a los automóviles los conocidos como sistemas de seguridad "proactiva". Estos sistemas tienen como objetivo prever y evitar tanto accidentes como posibles situaciones de riesgo que puedan derivar en un futuro accidente.

Con respecto a este tipo de sistemas podríamos nombrar las innumerables creaciones que existen, así como todos aquellos que están en proceso de desarrollarse. En nuestro caso nos centraremos en el más relevante para este proyecto.

El TPMS, también conocido como sistema de monitorización de la presión de los neumáticos, tiene como finalidad advertir al conductor de la existencia de una presión deficiente en alguno de los neumáticos del vehículo. Este sistema proporciona una alerta, generalmente mediante el encendido de un testigo en el cuadro de instrumentos cuando existe una variación en la presión de alguno de los neumáticos con una diferencia del 25 % o mayor.

Según un estudio realizado por la National Highway Traffic Safety Administration (NHTSA) [1] se confirma que en el año 2017, 738 accidentes mortales fueron debidos a problemas en los neumáticos. Otro estudio, en este caso desarrollado por Real Automóvil Club de España (RACE) [2], indica que el 5 % de los vehículos del parque automovilístico nacional, podrían poseer defectos en los neumáticos. Por todo esto, y el riesgo que esto supone, a partir de noviembre de 2014, la Comisión Europea hizo obligatoria la existencia de un control automático de presión en todos los vehículos nuevos de la categoría M1 de acuerdo con la directiva EC661-2009. Sin embargo, en otros lugares como Estados Unidos lleva siendo obligatorio desde 2008 con la aplicación de la normativa EFMVSS138 desarrollada por la NHTSA. En otros países como pueden ser Korea, Japón, Rusia, Kazajistán, Indonesia, Israel, Malasia, Filipinas, Turquía o Bielorusia no comenzó en entrar en vigor una normativa sobre su uso hasta comienzos de 2015 o finales de 2014.

3.1.1. Sistemas TPMS

Una vez aquí, y comprendida la gran importancia que los sistemas de seguridad tienen en el diseño de los automóviles, vamos a profundizar más en el último que hemos mencionado, el TPMS.

En cuanto a los sistemas de monitorización de presión podemos diferenciar dos tipos, indirectos y directos.

El TPMS indirecto consiste en la utilización de sensores existentes en el ABS o en el ESP encargados de determinar la presión de los neumáticos a través de sencillos algoritmos

que tienen en cuenta la velocidad de giro de estos y el diámetro de los mismos y, una vez calculada, enviar estos datos a la Engine Control Unit (ECU), "Unidad de Control Central", para poder alertar de presiones anómalas en caso de que se diese.

Una limitación a destacar de este tipo de sistemas, la encontramos cuando los usamos con neumáticos de perfil bajo. Este tipo de neumáticos cuentan con unas paredes laterales rígidas y cortas, en los cuales una pérdida de presión hace que la disminución del diámetro sea muy poco significativa, lo que dificulta en gran medida la detección de estas pérdidas de presión. Por contraposición, los neumáticos con unas relaciones de aspecto más altas y paredes laterales más grandes son mucho más compatibles con este tipo de sistemas. Un último detalle a tener en cuenta en cuanto a este tipo de sensores, sería que los sensores TPMS indirectos no son capaces de detectar una pérdida de presión simultánea en todas la ruedas del vehículo.

Por su parte, el TPMS directo consiste en un sensor incluido dentro del neumático, que se encarga de medir tanto la presión como la temperatura y enviar estos datos mediante una señal de radio al receptor central, que será el encargado de avisar al conductor de la existencia de alguna medida de presión deficiente. Este tipo de sensores, por norma general, suelen alertar cuando se produce una pérdida de presión superior a los 10 kpa (aproximadamente 0.1 bar).

En cuanto a este tipo de sistemas, una limitación importante que queremos reseñar es la vida de la batería. La vida media de la batería de estos sensores ronda los 7 años, sin embargo, a pesar de ser baterías de litio de alta duración, esta varía entre los 3 y los 10 años dependiendo de muchos factores, como pueden ser la frecuencia de muestreo del sensor, el uso del vehículo e incluso la temperatura ambiente. Es importante tener en cuenta esta duración de la vida de la batería, pues en la mayor parte de los sensores TPMS directos la batería está sellada, por lo que no se puede reemplazar únicamente la batería, sino que habría que reemplazar el sensor completo. Además es importante destacar que la mayoría de los expertos recomiendan no reemplazar únicamente el sensor con la batería gastada, sino que se recomienda reemplazar todos los sensores del vehículo ya que al haber sido instalados todos los sensores en la misma fecha y haber estado recibiendo el mismo uso todos ellos, se estima que cuando se produce la necesidad de remplazar uno de los sensores será necesario remplazar los restantes en un breve periodo de tiempo ya que todos habrían sufrido un desgaste similar y por tanto las baterías deberían tener una duración similar.

A pesar de las limitaciones comentadas anteriormente, comparando ambos sistemas podemos observar ciertas ventajas por parte de los sistemas TPMS directos. La principal ventaja a destacar es su mayor precisión en la detección de los datos, así como su mayor volumen de datos detectados. Mientras que el TPMS indirecto solo detecta la falta de presión, el TPMS directo puede detectar la pérdida de presión exacta, indicar en qué neumático se produce esta pérdida de presión, así como también la temperatura a la que se encuentra el neumático.

Es importante remarcar que todos los sistemas TPMS son valorados y certificados de acuerdo con el análisis AEC-Q100, "Failure Mechanism Based Test Quialification For Integrated Circuits" [1], desarrollado por el Automotive Electronics Council (AEC) o bien

[1] http://www.aecouncil.com/Documents/AEC_Q100_Rev_H_Base_Document.pdf

siguiendo el análisis ASIL-QM target B [2], "does not therefore require safety measures in accondance with ISO 26262", desarrollado por el Automotive Safety Integrity Level (ASIL).

3.2. Radiofrecuencia

Una vez aquí, ya hemos visto una explicación de qué es, qué tipos existen y qué ventajas tienen los distintos sistemas TPMS. Ahora bien, como sabemos estos sistemas utilizan señales de radio para transmitir sus mensajes, comprendidas en el rango que va desde los 315MHz hasta los 915MHz, dependiendo de la región en la que estemos transmitiendo. En nuestro caso, al tratarse de Europa, se utiliza la frecuencia de 433MHz.

Si entramos en el campo de las telecomunicaciones, se estudia la transmisión y recepción de señales de cualquier naturaleza, típicamente electromagnéticas. El conjunto de señales electromagnéticas se denomina espectro electromagnético, que se extiende desde la radiación de menor longitud de onda, como son los rayos gamma, hasta las ondas de mayor longitud de onda, como son las ondas de radio. Dentro de este espectro se encuentran las microondas, generalmente comprendidas entre frecuencias de 300MHz hasta los 300GHz.

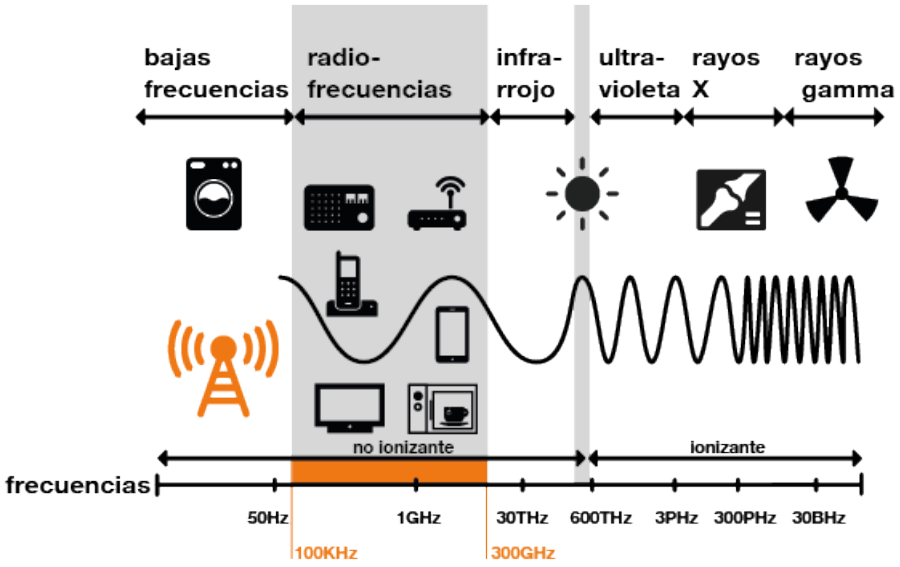

Figura 3.1: Espectro electromagnético

Radiofrecuencia (RF), también denominado espectro de radiofrecuencia, se sitúa entre los 3Hz y los 300GHz, abarcando así ondas de radio, radar y microondas.

3.3. Telecomunicación

La necesidad de comunicarse a distancia ha estado presente desde siempre, y gracias a la comunicación entre máquinas es posible satisfacer esta necesidad, además de facilitar

[2]https://training.ti.com/sites/default/files/pmic_safety_detroit_techday.pdf

otras que se presentan a diario. Tecnologías como la televisión, radio, teléfono y telefonía móvil, y comunicaciones de redes informáticas, Internet... nacieron para satisfacer necesidades científicas o militares pero que con el tiempo se fue ampliando su uso hasta ámbitos más comunes de la vida cotidiana. La principal característica de estos sistemas es que son capaces de transmitir y recibir señales electromagnéticas mediante unos dispositivos. Los módulos RF son un ejemplo de estos dispositivos.

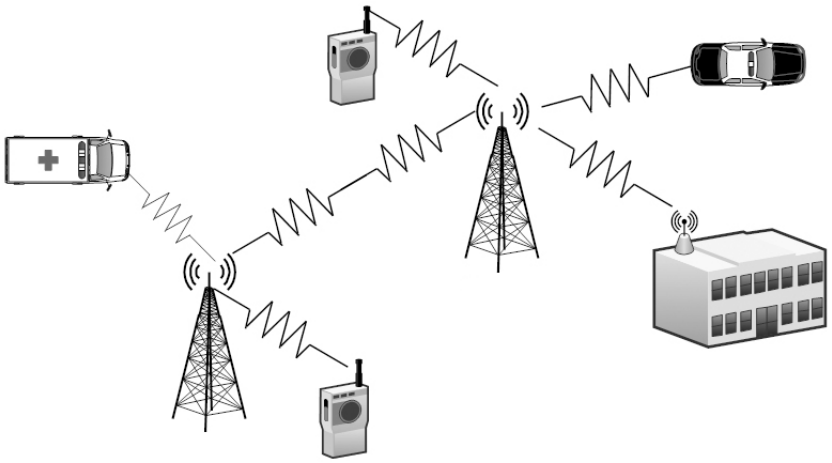

Figura 3.2: Ejemplo de comunicación por radio

Estos módulos RF se utilizan en multitud de aplicaciones de control remoto, como ventanas, puertas de garaje, control remoto de robots... pero también se utilizan en estaciones meteorológicas domésticas, que informan de las condiciones ambientales, y, lo que más nos interesa, en sistemas TPMS para informar del estado de los neumáticos del vehículo.

A la hora de hablar de la radiocomunicación hay que destacar que se pasa por diversas etapas, empezando por la construcción de la trama de datos que se desea transmitir(emisor) hasta culminar con la recepción de dicha trama por la otra parte(receptor). En cuanto a qué podemos transmitir, existe multitud de información que se puede transmitir, como imágenes, sonidos, datos... Sin embargo, algo imprescindible a la hora de realizar la comunicación entre emisor y receptor es que deben tratar a la señal de la misma manera.

3.3.1. Tipos de señales

Dicho esto, existen dos tipos de señales ampliamente diferenciados, por un lado las señales analógicas, utilizadas normalmente para las transmisiones por radio, y por otro lado las señales digitales se utilizan para transmitir por cable.

Con respecto a las señales analógicas, destacar que se trata de señales continuas, es decir, señales en las que varía su amplitud y periodo en función del tiempo. Sin embargo, las señales digitales se tratan de señales discretas que poseen un número de valores especificado, los cuales son los únicos que puede tomar la señal. Por ejemplo, en un sistema binario solo existen dos valores, alto y bajo (1 y 0).

En el contexto de este proyecto, nos vamos a centrar en las señales de tipo digital por lo que a continuación procederemos a explicar los distintos tipos de codificación que existen en estos tipos de señales, ya que es un proceso clave para la posible transmisión de datos digitales.

3.3.2. Tipos de codificación

Las dos codificaciones más importantes de cara a nuestro proyecto son la codificación Manchester y la codificación Manchester diferencial. La codificación Manchester es una codificación bifase autosincronizada en la que en cada tiempo de bit hay una transición entre dos niveles que representa uno de los dos valores posibles. Una transición de polaridad positiva a negativa representaría un 0 y una transición de polaridad negativa a positiva representaría un 1, esto según el estándar IEEE. Otros autores como G.E. Thomas establecen el valor al contrario y es el valor que se utiliza en el caso de Citroën, un 1 para representar una transición de polaridad positiva a negativa y un 0 para una transición de polaridad negativa a positiva.

Por su parte, la codificación Manchester diferencial se basa en la presencia o ausencia de transiciones para indicar el valor de la señal, es decir, al tratarse de una codificación autosincronizada se utiliza una transición en mitad del intervalo del bit para proporcionar esta sincronización. Además de esto, existe la posibilidad de transición al principio del intervalo del bit, donde la existencia de esa transición al inicio del intervalo representaría un 0 y la ausencia de esa transición representaría un 1. En la codificación Manchester diferencial, si el 1 es representado por una transición, entonces el 0 es representado por dos transiciones y viceversa.

A continuación se puede observar una imagen con un resumen con las principales codificaciones digitales existentes además de las explicadas previamente.

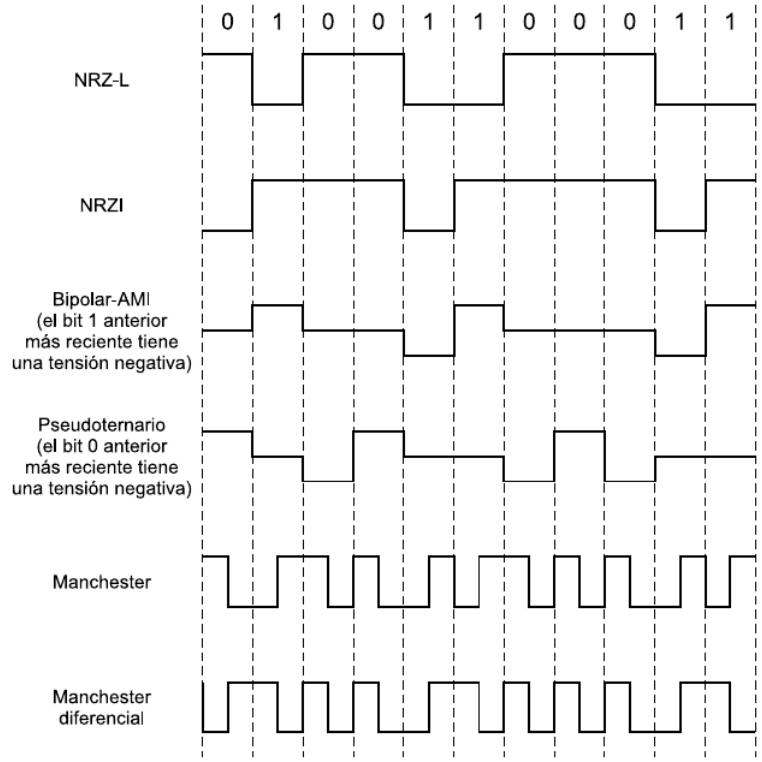

Figura 3.3: Codificaciones digitales
Fuente: Jamj2000 - Trabajo propio, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=36452370

3.3.3. Modulación

Ahora bien, una vez explicadas las posibles codificaciones de la señal, vamos a explicar los distintos tipos de modulaciones, es decir, las distintas técnicas disponibles para transportar información sobre una onda portadora. En el contexto de este proyecto nos centraremos en dos tipos concretos, la modulación por desplazamiento de frecuencia y la modulación por desplazamiento de amplitud.

La modulación por desplazamiento de frecuencia, también conocida como Frequecy Shift Keying (FSK), se trata de una técnica que utiliza dos frecuencias base, con valores de +- la frecuencia portadora para representar los valores de 1 y 0, pero sin variación de amplitud ni de fase. Sin embargo, la modulación por desplazamiento de Amplitud, también conocida como Amplitude Shift Keying (ASK), se trata de una técnica que utiliza una onda modulada plana para representar el valor de 0 y una onda de la frecuencia de la portadora con una amplitud determinada que representa el valor de 1.

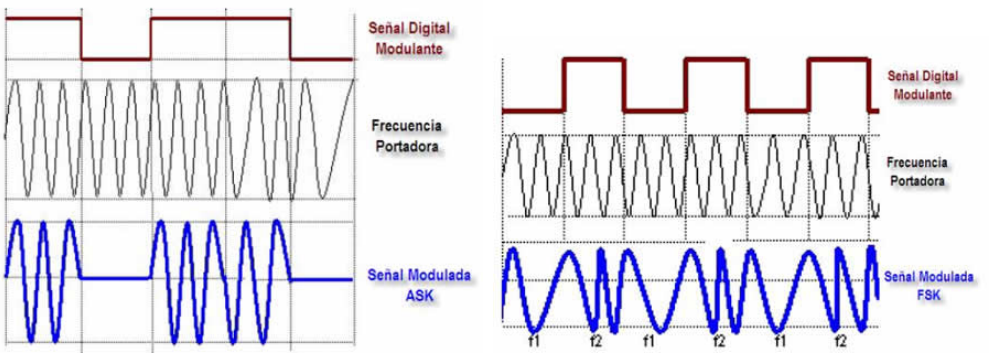

Figura 3.4: Modulación ASK y FSK
Fuente: https://ikastaroak.ulhi.net/edu/es/IEA/ICTV/ICTV02/es_IEA_ICTV02_Contenidos/website_551_modulacin_digital_ask_y_fsk.html

3.4. Seguridad en la transmisión de señales

Una vez explicados tanto el espectro electromagnético como el espectro de radiofrecuencia, vamos a centrarnos en los posibles aspectos de seguridad a tener en cuenta.

En primer lugar, se encuentran los protocolos de seguridad empleados en las señales de radio analógicas, las cuales emplean sistemas de criptofonía o secrafonía por inversión simple, que permiten distorsionar el mensaje para hacer la conversación ininteligible, pudiendo ser clarificadas con gran facilidad con el software apropiado. Este tipo de cifrado tiene diferentes métodos, entre los que destacan cuatro.

El primer método a abordar, y el que se considera el mejor de todos, es la partición del tiempo. Consiste en que la señal es almacenada y dividida en fracciones de tiempo, dividiendo así el tiempo total de la señal, en periodos de tiempo menores, los cuales son reordenados siguiendo una secuencia clave, conocida por el receptor, y enviados. El principal problema que se encuentra en este método es la sincronización, ya que no es posible una sincronización en tiempo real, lo que produce un retraso en todas las comunicaciones.

El segundo método es la partición de frecuencias, que consiste en dividir las frecuencias en grupos que engloban un rango menor dentro de la frecuencia total, que luego son barajados, convirtiendo las frecuencias reales en grupos de frecuencias menores, los cuales serán mezclados y ordenados creando una especie de puzle que nos permite proteger las comunidades. Otro método muy similar es la partición de frecuencia rodante, la cual está basada en el mismo concepto que la anterior pero los cambios en las frecuencias van cambiando varias veces por minuto, es decir, no se utiliza una única división de frecuencia como en el caso anterior, sino que esta varía creando así grupos de distinto tamaño que engloban rangos de frecuencia que abarcan un mayor o menor subconjunto dentro del espectro total.

El último método a destacar es la inversión del espectro. Se trata del método más sencillo y comúnmente utilizado que consiste en convertir las frecuencias graves en agudas y viceversa. En las comunicaciones de VHF y UHF, que son en las que se centra este proyecto, se invierte sobre los 4KHz, quedando las frecuencias de 1KHz convertidas en frecuencias de 3KHz y viceversa, mientras que las frecuencias de 2KHz se quedan

igual.

En el siguiente grupo de tecnologías se encuentran aquellas redes de radio que no pueden ser escuchadas de forma directa debido a que están codificadas digitalmente. Sin embargo, estas comunicaciones no poseen ningún tipo de cifrado por lo que pueden ser decodificadas en tiempo real usando el software adecuado. En este grupo encontramos las comunicaciones digitales como Radio Móvil Digital (DMR) y Terrestrial Trunked Radio (TETRA) no protegidas, sin embargo, ambos sistemas pueden ser protegidos utilizando cifrados, consiguiendo así que la comunicación sea segura.

Para el caso de las comunicaciones haciendo uso de DMR se observa que cuentan con diferentes mecanismos de seguridad, que van desde los más básicos, conocidos como "Basic Privacy", los cuales son sensibles a ataques de fuerza bruta, llegando hasta un cifrado de las comunicaciones más robusto utilizando implementación de estándares AES, además de contar con controles de acceso a la red como pueden ser los RAS (Restricted Access to System).

Con respecto a la redes TETRA, utilizan un cifrado extremo a extremo, lo que las brinda de un alto nivel de seguridad siempre con el objetivo de garantizar los cinco aspectos claves de la seguridad de la información, que son la autenticación, la confidencialidad, la integridad, la disponibilidad y el no repudio. A pesar de todo esto, estas redes son susceptibles de recibir ataques, como pueden ser "Jamming" también conocidos como interferencia intencionada, ataques a la infraestructura IP, denegación de servicios (DoS) o monitorización del tráfico y obtención de inteligencia a partir de las escuchas. Para abordar estas amenazas de seguridad, las redes TETRA disponen de mecanismos de seguridad para garantizar estos cinco aspectos claves de seguridad.

El primer mecanismo de seguridad a destacar es el proceso de autenticación de los terminales de radio, así como la posibilidad de autenticar en las consolas de dispatcher a los usuarios que van a hacer uso de los aplicativos, permitiendo de esta manera tener siempre controlado quien hace uso de las consolas para poder rastrearlo en caso de que se produjese un uso inadecuado de las mismas. Además, una vez que se ha producido la autenticación de los usuarios, se les concede un nivel de autorización, lo cual permite limitar los permisos de los usuarios, brindándole acceso solo a aquellas funcionalidades que debe utilizar, y limitando por esta vía el acceso a diferentes recursos a los que no deba poder acceder.

Para garantizar la confidencialidad de los mensajes, las redes TETRA cuentan con un sistema de cifrado del interface aire-aire, es decir, entre el terminal y la estación base, a lo que habría que sumarle la codificación extremo a extremo, también conocido como cifrado "end-to-end". Posteriormente, en la parte de backend de las redes TETRA, se cuenta con mecanismos de recuperación de la información almacenada en las bases de datos, además de mecanismos para evitar el uso no autorizado de software de dispatcher, los cuales han sido explicados anteriormente,así como controles con el objetivo de monitorizar el estado de los terminales y medidas de monitorización para llevar un control de los dispositivos registrados en la red.

Con el objetivo de garantizar la disponibilidad de las redes TETRA, estas cuentan con un plan de recuperación (DRP), procedimientos de reemplazo de las unidades que puedan fallar y procesos de monitorización tanto de los enlaces, permitiendo así modificar los caminos empleados y re-enrutar la información, como de la carga de los enlaces

para evitar posibles colapsos o saturaciones de la red. También dispone de un despliegue de procedimientos de "fall-back" y capacidad de operar en modo DMO (Direct Mode Operation).

3.5. Vulnerabilidades y ejemplos

Ahora que ya se han explicado los diferentes métodos que utilizan otros protocolos para proteger sus comunicaciones y garantizar la seguridad en las mismas, explicaremos las posibles vulnerabilidades relacionadas con la forma de trasmitir mensajes en "claro" de los sistemas TPMS.

El primer artículo que se desea destacar es el titulado "Predictive safety control for road vehicles after a tire blowout"[3], desarrollado por Wang et al. En este estudio se explica como en un futuro será posible, aprovechando estos sensores de presión, no solo alertar al conductor de posibles deficiencias de presión, sino que también, teniendo en cuenta una serie de parámetros como son la velocidad, el momento o la inercia del vehículo, los controladores de seguridad predictivos[3] con los que estarán dotados los vehículos serán capaces de corregir la trayectoria del vehículo, evitando así posibles accidentes.

Otro artículo muy relevante para el desarrollo de este proyecto sería el titulado "Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study"[4], desarrollado por Ishtiaq Rouf et al. En este artículo se puede encontrar un estudio realizado con dos sistemas TPMS diferentes, mediante el cual es posible concluir con relativa certeza aspectos como pueden ser el rango de detección de las señales haciendo uso de dispositivos de bajo coste, o incluso podemos observar ciertos requisitos necesarios a la hora de transmitir nuestras señales para lograr engañar a la ECU. El artículo concluye con una serie de recomendaciones con el objetivo de evitar que este tipo de ataques se produzcan.

Otro gran hecho sobre el que se debería reflexionar y el cual hizo reflexionar especialmente al cofundador y director general de Testa, Elon Musk, cuando en 2019 en la Pwn2Own, concurso anual de piratería informática celebrado en la conferencia de seguridad CanSecWest, decidieron exponer su último modelo hasta la fecha, el Tesla Model 3, para que los aspirantes tratasen de hackear el vehículo y pudieran comprobar si contaba con alguna brecha de seguridad. Fue entonces cuando el equipo conocido como "Fluoroacetate" y compuesto por los investigadores Amat Cama y Richard Zhu fue capaz de hackear el vehículo cuando lograron infiltrar el software del coche a través de su navegador de internet integrado tras detectar un error JIT (Just in Time), consiguiendo por esta vía alterar el código del firmware del vehículo y tomar el control del mismo.

Relativo a todas las vulnerabilidades comentadas anteriormente, es importante destacar que desde septiembre de 2016, existe un grupo de personas, el cual se conoce como ISO/TC 22/SC 32, que está tratando de desarrollar un estándar de ciberseguridad, que será conocido como ISO 21434, y cuyo objetivo es aplicar dicho estándar al mundo del automóvil y reducir el potencial riesgo de producirse un ataque.

Antes de empezar a desarrollar el proyecto, el interés por saber si existían proyectos como el nuestro para conocer más posibilidades que ofrece todo esto, nos llevó a investigar, encontrando así algunos proyectos interesantes.

[3]Predictive safety controller

En primer lugar, un proyecto desarrollado por Steve Mould consistía en captar la señal que manda una llave de un coche con el fin de replicarla para poder desbloquear el coche. A lo largo del estudio, y tras investigar, se da cuenta que tanto el coche como la llave tiene un conjunto de códigos estático que simplemente va rotando el código que se envía, por lo tanto puede captar dicho código que envía la llave cuando esté fuera del alcance del coche y así poder posteriormente reenviar dicha señal. A pesar de que es posible desbloquear el coche, hay que decir que hay muchos factores por los que sería difícil llevar acabo dicho ataque contra otra persona. Todo esto está explicado en un vídeo de YouTube[4].

Otro ejemplo que vimos y que nos resultó curioso fue el proyecto desarrollado por un usuario de GitHub, que consiste en la recepción y transmisión de señales con el fin de tener un dispositivo que funcione como un mando de garaje. Enlace al repositorio[5].

Todo esto nos llevó a querer demostrar las posibles vulnerabilidades que poseen los sistemas TPMS con el objetivo de que puedan subsanarse con posteriores mejoras de estos sistemas, evitando así posibles ataques que puedan llegar a suponer una causa más que incremente el número de accidentes de tráfico.

3.6. Trabajo previo

Para demostrar esto partimos de la base teórica demostrada por nuestro compañero Alberto Caballero Gámez, en su Trabajo de Fin de Grado (TFG) titulado "Estudio de la seguridad del protocolo TPMS"[5]. En este TFG Alberto desarrolla el marco teórico que posibilitaría un ataque aprovechando una vulnerabilidad existente en los TPMS directos que emiten sus señales de radio sin ningún tipo de cifrado.

Un último trabajo que deseamos remarcar sería la conferencia ofrecida por Pedro Cancel, también conocido como "s4ur0n", en la "Hackplayers c0nference MMXX" titulada "Spoofing Tire Pressure Monitor System (TPMS)[6], en la cual Pedro realiza un análisis de todo lo que engloban los sistemas TPMS, partiendo desde los detalles técnicos de dichos sistemas, hasta explicar un posible proceso de análisis y transmisión de la señal, así como detallando las distintas herramientas que él utiliza durante la realización de su proyecto, las cuales pueden sernos de gran ayuda en el desarrollo del nuestro.

Partiendo de esta base, y contando también con algún que otro estudio adicional como los que hemos mencionado anteriormente, comienza este proyecto en el que tratamos de demostrar que este marco teórico que explicaba Alberto es cierto y que podemos simular un ataque aprovechando esta vulnerabilidad existente en los sistemas TPMS directos.

[4]https://www.youtube.com/watch?v=5CsD8I396wo&ab_channel=SteveMould
[5]https://github.com/LSatan/Simu_Remote_CC1101

Capítulo 4

Arquitectura del sistema

Este es el capítulo principal de la memoria cuya finalidad es explicar y dar a conocer la estructura del proyecto y como ha sido el desarrollo del mismo, diferenciando las partes principales y especificando en ellas las tecnologías que se han empleado.

Los primeros cuatro puntos en los que se divide esta sección son Recepción de la señal, Generación de la señal, Transmisión de la señal y Desarrollo de la interfaz. Estas son las pequeñas metas que se establecieron durante el desarrollo del proyecto con el fin de estructurar el trabajo del equipo de una manera ordenada y viable. Estos puntos siguen la misma disposición, empezando por una descripción y explicando cual es la implicación que tienen para el proyecto. Seguidamente se explica como se realiza o como se lleva acabo este en el proyecto. Al final de cada punto se listan algunas herramientas Software o Hardware importantes para la realización de dicho punto, explicándolas con un detalle un poco más técnico.

En el quinto punto se hace un repaso de las tecnologías que se han utilizado y algunas de las que se han descartado, explicando brevemente los motivos.

4.1. Recepción de la señal

En esta sección se explica una de las partes más importantes para el desarrollo de este TFG. El objetivo principal de este punto es poder recibir las señales TPMS con una antena RTL-SDR, como se explica a lo largo de esta sección. Se busca conseguir poder consumir la información captada por el PC para las siguientes partes del proyecto.

La relevancia de este punto viene dada por la necesidad de poder recibir una señal para poder estudiar y analizar lo recibido, ya que en un principio no se tenía casi ningún conocimiento sobre el mundo de la Radio Frecuencia y Radio Hacking.

El principio de esta parte fue útil para poder comprender las señales TPMS y los diferentes campos que abarca esta tecnología, ya que sirvió para investigar y documentarse sobre estos temas. Se forjó la base y el conocimiento para desarrollar los diferentes puntos de este proyecto.

¿Cómo se realiza esta recepción? Para realizar este paso es necesario un dispositivo RTL-SDR capaz de escuchar señales en la frecuencia 433,92 Mhz, ya que, como se ha comentado en el Estado del Arte, las señales TPMS obligatoriamente tienen que ocupar

esta frecuencia según el Cuadro Nacional de Atribución de Frecuencias (CNAF)[7]. Este tipo de dispositivos captan señales de radio y mediante un demodulador convierte la señal analógica a digital haciéndola comprensible para un computador. La conexión de este hardware al Personal Computer (PC) se realiza mediante una conexión a un puerto Universal Serial Bus (USB).

Una vez procesada la información proveniente de la antena receptora se hace uso de diferentes softwares que interpretan de manera diferente la información. Existen programas que recaban datos para mostrar el espectro de Radio como GQRX[1], en los cuales se puede ver como llega la información por las diferentes frecuencias junto con el ruido ambiente de una manera instantánea. Otros tipos de programas muestran la información recolectada en forma de onda ya decodificada y los respectivos bits que forma en datagrama de la señal. Para este proyecto se han utilizado varias aplicaciones para recepción de las señales. Por ejemplo GQRX principalmente sirvió para ver en qué frecuencia del espectro de radio se recibían la señales TPMS y en un punto más avanzado para ver las señales que se han logrado transmitir. Otras dos aplicaciones que se han utilizado son RTL_433 y Universal Radio Hacker (URH) que se explicarán en las siguientes subsecciones con un nivel más técnico y detallado.

En cuanto al dispositivo RTL-SDR, tras barajar varios tipos de receptores que cumplían las especificaciones necesarias, se decidió utilizar el modelo de Nooelec, "NESDR SMArTee v2 Bundle", ya que a priori sus especificaciones parecían de las mejores opciones de cara al proyecto.

4.1.1. Módulo receptor Nooelec NESDR SMArTee v2 Bundle

Este módulo es una Software Defined Radio (SDR) fundamentada en RTL2832U y R820T2. Según sus especificaciones es capaz de recibir cualquier señal RF aproximadamente 25MHz-1700MHz.

El paquete utilizado en este proyecto incluye el módulo principal SDR que se conecta mediante un puerto USB al equipo. También viene con un cable para conectar la antena con el módulo USB de una longitud de 2 metros. Por último contiene tres tipos de antena con diferentes características de fácil instalación que permite recibir diferentes rangos.

Para poder utilizar este dispositivo con nuestro equipo Linux lo primero que hay que hacer es instalar los drivers, siguiendo los pasos indicados por el fabricante en las instrucciones.

Especificaciones generales SDR indicadas por el fabricante

- Demodulador RTL2832U e interfaz USB IC
- IC, sintonizador R820T2
- 4.5V 250mA de polarización siempre activa.
- Ruido de fase ultrabajo TCXO a 0.5PPM

[1]Pagina ofical de GQRX: https://gqrx.dk/

- Regulador de voltaje apto para RF
- Inductor principal blindado
- Disipador de calor integrado
- Entrada hembra de antena SMA
- Carcasa de aluminio cepillado negro de alta calidad
- Muestreo directo en las almohadillas de la Printed Circuit Board (PCB)

Figura 4.1: Pack modulo RTL-SDR Nooelec

4.1.2. RTL 433

Es un software de código abierto que permite recibir señales para las bandas ISM 433.92 MHz, 868 MHz (SRD), 315 MHz, 345 MHz, y 915 MHz. Este software fue alojado en un repositorio[8] de github por el usuario Benjamin Larsson, bajo el nombre de usuario de merbanan.

Una vez instalado el dispositivo RTL-SDR lo utilizaremos para escuchar y poder recibir las señales mediante este programa.

Este software, tal y como indican en el repositorio, es válido para sistemas operativos Linux, Windows y MacOs. Para instalar este software en la distribución Ubuntu, o cualquiera basado en la distribución Debian(19.10+), simplemente utilizar la gestión de paquetes para descargar e instalar este software, mediante el paquete rtl-433. Para más detalle consultar el github de este proyecto.

En cambio para la Raspberry OS y algún otro Sistema Operativo hay que instalarlo compilando el código e instalar las librerías necesarias para el funcionamiento de esta, a consultar en el repositorio de github.

Esta aplicación tiene un interfaz de líneas de comandos. Permite la entrada de parámetros de los cuales depende para su funcionamiento y su posterior salida. Además, se puede configurar el tipo de señales que se quieren recibir. Una de las partes fundamentales de este TFG es recibir señales TPMS y gracias a la configuración que ofrece esta aplicación, podremos filtrar las señales, incluso hasta por la marca de los vehículos.

En nuestro proyecto se hace uso de este software para la parte de la recepción de la señal de la aplicación. Se escogió esta opción debido a su facilidad a la hora de filtrar los vehículos y su salida de datos. Esta salida es configurable de varias maneras, como puede ser Comma-Separated Values (CSV), JavaScript Object Notation (JSON)... La opción de JSON es la que se utiliza ya que a la hora de utilizar la información en nuestra aplicación es más sencillo gestionar todo en este formato.

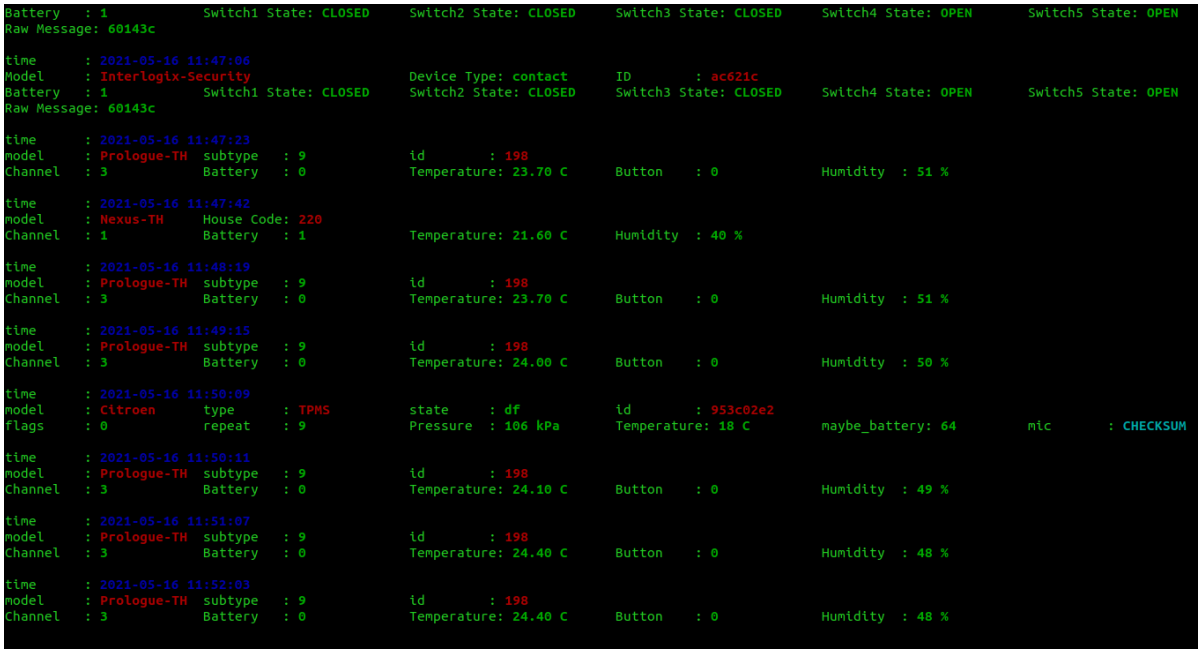

Figura 4.2: Captura de una escucha realizada por el software RTL-433, en la cual se ve que recibe un vehículo Citroën

4.2. Generación de la señal

Esta parte del proyecto tiene la finalidad de generar la trama de una señal TPMS para posteriormente en siguientes pasos poder transmitirla. Cada fabricante suele tener su propio protocolo que aunque cuentan con ciertas características similares, cabe remarcar que también cuenta con unas características propias y ampliamente diferenciadas de los protocolos de otras marcas, por ello se abordará cada protocolo por separado.

El objetivo es generar una señal falsa con unos parámetros dados que permitan activar la alarma del piloto de la presión de las ruedas de un vehículo, y con ello activar los protocolos de seguridad propios de cada modelo. En nuestra aplicación se generará una

de estas señales falsas pasándole el identificador de una rueda junto con el modelo del vehículo.

En el desarrollo de cada una de las tramas, la principal fuente de información fueron los diferentes archivos de código relativos a cada protocolo que es capaz de decodificar el software RTL 433, que se pueden encontrar en el repositorio de Github oficial de RTL 433[8].

Para este proceso se han creado diferentes algoritmos necesarios para la creación de las tramas TPMS. Para la codificación de las ondas se ha desarrollado un código para codificar tramas en Machester y Machester Diferencial. Estos algoritmos dados una secuencia de bits, en este caso la trama TPMS, los codifican con el algoritmo correspondiente. También ha sido necesario desarrollar algoritmos para realizar el checksum y el algoritmo crc8 de una secuencia de bits.

Una vez generada esta señal y antes de pasar al siguiente proceso hay que comprobar los diferentes algoritmos de generación de señales. Para ello utilizaremos el software de tx_tools [9]. Este software es capaz de transmitir y generar la onda de señales con una trama dada, utilizando la librería SoapySDR. Este software está basado en dos conocidas librerías rtl_sdr [10] y rx_sdr[11]. Lo que conseguimos con tx_tools es pasarle nuestra trama de la señal creada y generar con ella una señal RF. Esta señal RF ya sería posible enviarla por algunas antenas transmisoras del mercado, como HackRF One[12]. Para comprobar si el algoritmo ha generado bien la trama se hace uso del software rtl_433, leemos la señal RF y si este software la lee con los datos que se han pasado al generador de señales quiere decir que funciona correctamente.

4.2.1. Toyota

Este primer protocolo esta formado por un datagrama de 161 bits. Este está subdividido en diferentes campos de distintas longitudes. El primer campo, conocido como *Preamble* tiene una longitud de 14 bits, los cuales son utilizados para la sincronización de las comunicaciones. Le siguen 72 bits codificados utilizando Manchester dieferencial y finaliza con 4 bits, que actúan como *Final Trail*. En el siguiente cuadro se muestran los campos que forman la trama de Toyota.

Preamble	ID	Status1	Pressure1	Temperature	Status2	Pressure2	CRC	End of frame
14 bits	32 bits	1 bit	8 bits	8 bits	7 bits	8 bits	8 bits	3 bits

Cuadro 4.1: Trama de un sensor Toyota

Este *Preamble* o *preámbulo*, en el caso del protocolo Toyota, posee un valor definido y ya conocido, lo que permite que actúe como sincronización de la comunicación. Dicho valor es 01 0101 0100 1111.

A continuación se encuentran los diferentes campos que componen estos 72 bits mencionados previamente y que componen la parte principal de la trama.

El primer campo que se aprecia es el *ID*. Este campo tiene una longitud de 4 bytes y su función es identificar de manera única al sensor en cuestión desde el que se está transmitiendo.

Seguidamente se encuentra el *primer bit de estado*. El estado en este protocolo tiene una longitud de 8 bits, por lo que se ha de dividir en dos partes, una primera parte que se transmite aquí y que está compuesta únicamente por el primer bit, y una segunda parte que está compuesta por los siete bits restantes y que se transmite posteriormente en el segundo campo de estado.

A continuación se envían 8 bits correspondientes a la *presión* del neumático en cuestión. En este caso es importante tener en cuenta que la presión a codificar difiere de la presión real. Como bien se indica en el repositorio oficial de Github de RTL 433[8], la presión que se debe codificar se calcula como un cuarto de la presión real, habiendo sido desplazada esta en 7 unidades y medida siempre en PSI.

El siguiente campo que se encuentra es la *temperatura* que, al igual que el campo anterior, también posee una longitud de 8 bits. Para transmitirla es necesario tener en cuenta que tampoco se transmite el valor real, sino que en este caso la temperatura codificada se obtiene desplazando en 40 unidades el valor de la temperatura real, medida siempre en grados Celsius.

Seguidamente se transmite el *segundo campo de estado*. Este campo posee una longitud de 7 bits y, como bien se ha comentado antes, está compuesto por los 7 últimos bits del valor del estado.

A continuación se encuentra un *segundo campo de presión*. Este campo, al igual que el primero, tiene una longitud de 8 bits y se transmite el mismo valor de presión que se ha transmitido en el primer campo, pero en formato invertido. Este campo es utilizado como una forma más a la hora de realizar la comprobación de errores de la trama.

El último campo que encontramos dentro de estos 72 bits, sería el *CRC*, principal campo encargado de la comprobación de errores que posee una longitud de 8 bits. Para calcularlo se utiliza el CRC-8, utilizando como polinomio truncado el valor de 0x07 e inicializado con el valor de 0x80, tal y como se indica en el repositorio de RTL 433[8].

Finalmente, después de la trama codificada, se envían los 3 últimos bits, conocidos como *End of frame* que se calculan en función del resultado devuelto por la codificación Manchester diferencial, la cual devuelve como resultado la trama codificada y un bit de resultado. En función de este bit de resultado, se calculan los tres bits a transmitir como *End of frame*. En caso de que el bit resultado posea el valor de 1, el campo *End of frame* toma el valor de 000. En caso contrario, si el bit resultado posee el valor de 0, el campo *End of frame* toma el valor de 111.

4.2.2. Citroën

La trama generada por los dispositivos de los vehículos Citroën tiene una longitud de 200 bits. Esta señal esta compuesta por 10 campos de diferentes longitudes. Empieza con 2 bytes de sincronización conocidos como *Preamble* seguidos de 10 bytes codificados en Manchester y el final de la trama es 1 byte. En el cuadro 4.2 se puede observar la trama y los campos que la forman.

Preamble	State	ID	Flags	Repeat counter	Pressure	Temperature	Batery	Checksum	End of frame
16 bits	8 bits	32 bits	4 bits	4 bits	8 bits	8 bits	8 bits	8 bits	8 bits

Cuadro 4.2: Trama de un sensor Citroën

El *Preamble* o *preámbulo*, para el caso del protocolo Citroën, posee un valor definido y ya conocido que permite que actúe como sincronización de la comunicación. Este valor es 0101 0101 0101 0101 0101 0101 0101 0110.

A continuación se van a desglosar los diferentes campos que componen estos 10 bytes mencionados previamente y que componen la parte principal de la trama.

El primer campo que se observa es el campo *State* o *campo de estado*, encargado de informar sobre el estado del propio sensor que transmite la señal. El campo de estado en este protocolo posee una longitud de 8 bits.

A continuación, el siguiente campo que encontramos es el *ID*. Al igual que en el anterior protocolo, este campo tiene una longitud de 4 bytes y su función es identificar de manera única al sensor que está transmitiendo dicha señal.

Seguidamente se encuentran los *flags*, con una longitud de 4 bits y cuyos valores más habituales son, 0 en el 69.4 % de los casos, 1 en el 0.8 % de los casos, 6 en el 0.4 % de los casos, 8 en el 1.1 % de los casos, b en el 1.9 % de los casos, c en el 25.8 % de los casos y e en el 0.8 % de los casos.

El siguiente campo que se encuentra es el campo *Repeat* o *Repeat counter*. Se trata de un contador de repeticiones encargado de informar al receptor del número de veces que se ha enviado la trama. En este campo podemos encontrar los valores de 0, 1, 2 y 3.

A continuación, se trasmiten 8 bits correspondientes a la *presión* del neumático en cuestión. En este caso es importante tener en cuenta que la presión a codificar difiere de la presión real del neumático. Como bien se indica en el repositorio oficial de Github de RTL 433[8], la presión que se debe codificar se calcula aplicando un factor de escala de 1.364 pies sobre la presión medida en kPa, es decir, la presión a codificar será la presión real medida en kPa dividida de dicho factor de escala.

Seguidamente se encuentra el campo relativo a la *temperatura* que, al igual que el campo anterior, también posee una longitud de 8 bits y para trasmitirla es necesario tener en cuenta que tampoco se transmite el valor real. En este caso, la temperatura codificada se obtiene desplazando en 50 unidades el valor de la temperatura real, medida siempre en grados Celsius.

El siguiente campo que se encuentra es el campo *Battery* o *maybe_battery*. Cuenta con una longitud de 8 bits y la finalidad de este campo es informar sobre el estado de la batería del sensor.

El último campo que encontramos dentro de estos 10 bytes es el *Checksum*, principal campo encargado de la comprobación de errores, el cual posee una longitud de 8 bits y que se calcula dividiendo la trama actual en grupos de 8 bits y realizando la operación XOR entre cada grupo, obteniendo así como resultado los 8 bits de comprobación.

Finalmente, y después de la trama codificada en Manchester, se envía el último byte, conocido como *End of frame*. Este byte es utilizado para indicar al receptor que el envío de información ha finalizado y posee un valor previamente definido y conocido que es 0111 1110.

4.2.3. Renault

En el caso de Renault la señal consta de unos 180 bits y esta compuesto por 8 campos. Lo primeros 32 bits forman el *Preamble*, necesarios para la sincronización. Los 64 bits siguientes están codificados en Manchester. Termina con 4 bits conocidos como *Final Trail*. En el cuadro se pueden apreciar todos los campos que componen la trama de los sensores Renault:

Preamble	Flags	Pressure	Temperature	ID	Unknown	CRC	End of Frame
32 bits	6 bits	10 bit	8 bits	24 bits	16 bits	8 bits	4 bits

Cuadro 4.3: Trama de un sensor Renault

El campo *Preamble* o *preámbulo*, en el caso de Renault, posee un valor ya definido y conocido, lo que permite que actúe como sincronización de la comunicación. Dicho valor es 0101 0101 0101 0101 0101 0101 0101 0110.

A continuación se encuentran los diferentes campos que componen estos 64 bits mencionados previamente y que componen la parte principal de la trama.

El primer campo que se aprecia es el campo *Flags*. Este campo tiene una longitud de 6 bits y los valores más habituales que se pueden encontrar son, c0 en el 22 % de los casos, c8 en el 14 % de los casos, d0 en el 31 % de los casos y d8 en el 33 % de los casos.

El siguiente campo que se encuentra es el correspondiente a la *presión* del neumático en cuestión y posee una longitud de 10 bits. En este caso es importante tener en cuenta que la presión a codificar difiere de la presión real del neumático. Como bien se indica en el repositorio oficial de Github RTL 433[8], la presión que se debe codificar se calcula aplicando un factor de escala de 0.75 sobre la presión medida en kPa, es decir, la presión a codificar será la presión real medida en kPa dividida de dicho factor de escala.

Seguidamente se transmite el campo relativo a la *temperatura*, con una longitud de 8 bits. Al igual que en el caso anterior, es importante tener en cuenta que tampoco se transmite la temperatura real detectada por el sensor del vehículo, sino que en este caso la temperatura codificada se obtiene desplazando en 30 unidades al valor de la temperatura real, medida siempre en grados Celsius.

A continuación se envía el *ID*, con una longitud de 24 bits y cuya función, al igual que en protocolos anteriores, es la de identificar de manera única al sensor en cuestión desde el que se está transmitiendo. En este caso es muy importante destacar que, a diferencia de en el resto de campos, los bytes se escriben siguiendo un formato de "Little Endian", es decir, se ordenan los bytes del menos significativo al más significativo.

Los siguientes 16 bits de la trama son desconocidos, el autor del código desconoce cual es el motivo de estos, se le da un valor fijo de 0xffff.

El último campo que encontramos dentro de estos 64 bits, es el *CRC*, principal campo encargado de la comprobación de errores que posee una longitud de 8 bits. Para calcularlo se utiliza el CRC-8, utilizando como polinomio truncado el valor de 0x07 e inicializado con el valor de 0x00, tal y como se indica en el repositorio de RTL 433[8].

Finalmente, después de la trama codificada en Manchester, se envían los 4 últimos bits, conocidos como *End of Frame*. Estos bits son utilizados para indicar al receptor que

el envío de información ha finalizado, y posee un valor previamente definido y conocido que es 0000.

4.3. Transmisión de la señal

El objetivo de la transmisión de señales es conseguir enviar una señal generada o replicar una ya capturada. Estas transmisiones son muy utilizadas para diferentes ataques a vulnerabilidades de sistemas que se basan en la radiofrecuencia para su comunicación. Para realizar este paso es necesario tener una antena transmisora. Existen muchos tipos de estas antenas con diferentes cualidades y precios.

En nuestro proyecto esta parte es la culminación de todos los procesos anteriores. El objetivo después de recibir una señal y generar una señal falsa es transmitirla llegando a engañar a un vehículo con esta.

Para realizar esto en nuestro software, se ha conectado a una Raspberry Pi un transmisor Ti CC1101 mediante los puertos General Purpose Input/Output (GPIO), como se explicará en las subsecciones siguientes. Una vez conectado, para poder hacer uso de este y enviar señales, se utiliza un software de github, el cual programa el transceptor para enviar la señal con las características que se han configurado. Realizaremos el envío de la misma señal cuatro veces debido a las posibles interferencias que puedan producirse en la señal.

Al ser un dispositivo de tan bajo coste la reproducción de este paso ha sido bastante compleja. Finalmente hemos conseguido enviar una señal legible para el software de GnuRadio. La segunda prueba es que el propio software de RTL_433 la detecte y la lea correctamente. Ahora mismo este software detecta la señal enviada pero no es capaz de leerla correctamente. La última prueba se enfocaría a lanzar esta señal a un vehículo y que este encienda el piloto de alerta de TPMS. Esta prueba es más complicada de realizar por problemas de logística.

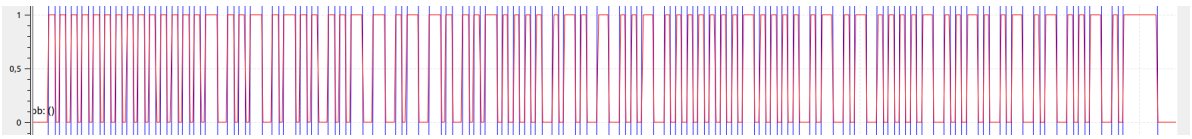

Figura 4.3: Recepción de una señal Citroën por el programa GNURadio

Otra posibilidad para realizar esta transmisión sería utilizar antenas más costosas, como la famosa antena HackRf One de Great Scott Gadgets. Este dispositivo ya está preparado totalmente para realizar este tipo de investigaciones y ataques. Realmente si a esta herramienta se le pasa un fichero de señal, como los que se han generado con la herramienta tx_tools para comprobar la señal, se podría realizar el ataque, ya que la antena lanzaría esta señal generada sin necesidad de configurar muchos parámetros. Comprobar este tipo de transmisiones es inviable debido al elevado precio del dispositivo.

4.3.1. Transceptor Ti CC1101

Para transmitir las señales TPMS se necesita un transmisor capaz de enviar las tramas con una codificación Manchester o Manchester Diferencial, modulación 2-FSK de la señal

y una antena que permita una frecuencia de operación de 433MHz. En nuestro caso se decidió utilizar un transceptor low-cost con el chip Ti CC1101 fabricado por la empresa Texas Instrument[13].

Este chip nos permite configurar bastantes parámetros del datagrama. Hasta un punto de que es posible enviar solamente nuestra trama y prescindir del preámbulo, Sync Word y el checksum CRC-16 del propio chip ajenos a la trama que se quiere transmitir. Con esto se consigue eliminar los parámetros opcionales del chip y así poder enviar nuestra trama, con una longitud máxima de 255 Bytes.

Para la comunicación con otros dispositivos, el chip en su PCB tiene unos pines preparados para el protocolo de comunicación Serial Peripheral Interface (SPI). Estos pines se comunican conectándose mediante cables hembra-hembra a los puertos GPIO de la Raspberry Pi 3 B+, conectando con los pines preparados para conectar dispositivos con SPI, utilizando los pines SPI0 como se explicará más adelante.

Figura 4.4: Transmisor Ti CC1101 conectado a Raspberry Pi 3 B+

4.3.2. Raspberry Pi

Las Raspberry Pi son pequeños ordenadores de un tamaño bastante reducido, similar al tamaño de una tarjeta de crédito, creada por la Raspberry Pi Foundation, una fundación de Reino Unido. Sacaron el primer modelo en el 2009 con el objetivo de llevar la digitalización y la computación a gente de todo el mundo, gracias a su bajo coste y al Software Libre.

Cada versión de las Raspberry Pi tiene diferentes especificaciones, ya que a lo largo de los años han ido mejorando sus tecnologías. En nuestro caso se decidió utilizar el modelo Raspberry Pi 3 B+. La elección no está basada en ningún criterio técnico, aunque recomendamos el uso de versiones Raspberry Pi 3 en adelante. Las versiones de Raspberry Pi Zero no servirían para este propósito ya que no tienen ningún puerto USB par poder conectar el dispositivo RTL.

Respecto a los softwares de terceros, no indican ninguna limitación para ninguna de las versiones a utilizar.

Especificaciones técnicas de la **Raspberry Pi 3 B+**:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @1.4GHz
- 1GB de RAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet 300Mbs
- 40 pines GPIO
- Puerto HDMI
- Cuatro puertos USB 2.0
- Puerto CSI para conectar una cámara
- Puerto DSI display para conectar una pantalla táctil
- Salida estéreo de 4 polos y puerto de vídeo compuesto
- Puerto micro SD
- Entrada de alimentación de 5V/2,5A

Existen otras placas, como las de Arduino, para poder utilizar en este proyecto, pero las configuraciones y el programa no alberga esta posibilidad. En nuestro caso se decidió utilizar la Raspberry Pi porque es un computador en sí mismo y puede realizar todas las tareas que requiere la aplicación.

Algo importante en este proyecto son los puertos GPIO para poder conectar el hardware con el software. De esta forma es posible tener una conexión del transmisor CC1101, del cual hablaremos en el siguiente apartado.

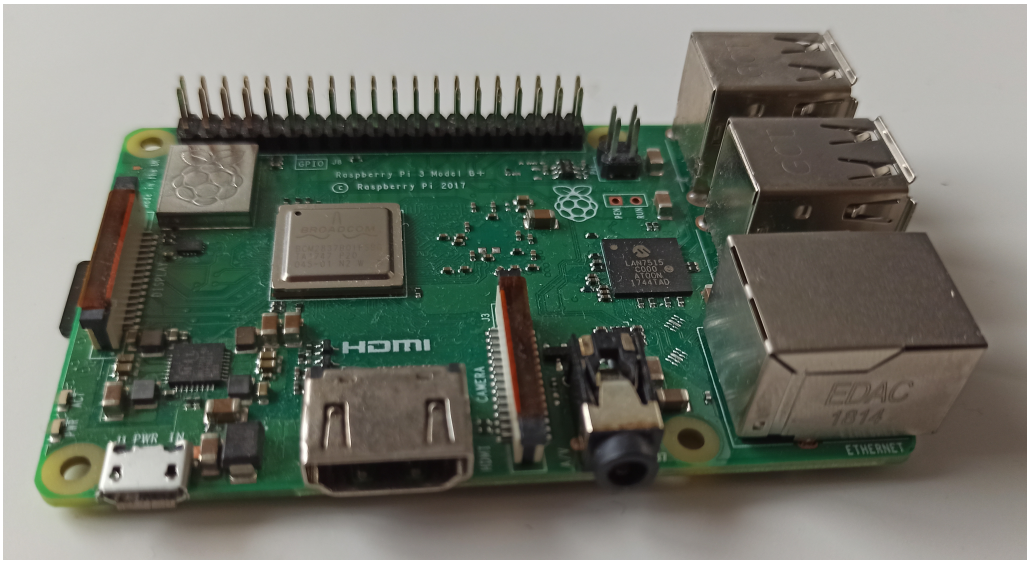

Figura 4.5: Raspberry Pi 3 B+

4.3.3. Python-cc1101

Es una librería[14] de Python y un herramienta para la Command Line Interface (CLI) para transmitir señales mediante transmisores que tengan el chip CC1101. Esta fue creada por Fabian Peter Hammerle, conocido en github como fphamerle. Es un repositorio bastante vivo ya que el creador intenta actualizar y arreglar los *bugs* muy continuadamente.

Esta herramienta permite utilizar un transmisor conectado a una Raspberry por los pines vinculados al protocolo SPI. Por defecto utilizará los pines del SPI0, por lo que si quieres configurarlo para otro bus SPI, deberás configurarlo a mano mediante las funciones de la librería.

Esta biblioteca permite configurar de una manera fácil y eficaz los parámetros para enviar las señales. Tiene diferentes funciones para configurar los campos opcionales del datagrama permitido por el propio chip.

En el mismo repositorio se pueden encontrar ejemplos que muestran la funcionalidad de la librería.

Este software está diseñado para funcionar en una Raspberry Pi con puertos GPIO. Para instalarlo, simplemente con el manejador de paquetes de Python 3 se puede descargar el paquete con el nombre cc1101. También es necesario una dependencia para manejar desde Python los buses SPI. Esta dependencia se instala con el manejador de paquetes "apt-get" con el respectivo python3-spidev.

4.4. Desarrollo de la interfaz

El objetivo de la interfaz es conectar todos los anteriores procesos en una aplicación manejable para el usuario para poder ejecutarlos con sencillez.

Uno de los principales objetivos de diseño era conseguir una interfaz sencilla, que fuera posible usarla incluso desde una pantalla táctil que probablemente se incorporara a nuestra Raspberry. Se buscó mostrar al usuario todos los datos relevantes de las señales detectadas, como son la temperatura, la presión, el modelo y el id de la rueda desde la que se emitió la señal captada.

Figura 4.6: Vista principal

Se ha decidido mostrar solo estos datos porque son los que se consideran imprescindibles, tanto para tratar de identificar el vehículo el cual estamos detectando, como para poder realizar el ataque a dicho vehículo, ya que, mediante la implementación realizada, los datos mínimos necesarios para realizar un ataque serían el id de la rueda en cuestión y el modelo, que determina una serie de condiciones en la codificación de la trama, como bien se ha explicado en el punto anterior de ”Generación de la señal”.

Dicha interfaz está claramente dividida en dos módulos o modos bien diferenciados. El primero es “Attack One”, donde se muestra una lista con las señales que se van detectando y donde se puede elegir una señal de la lista, lo que permite realizar un ataque a ese sensor en concreto. También es posible, en caso de conocer el id del sensor al que queremos atacar y el modelo, introducir estos valores manualmente y realizar un ataque a ese sensor en concreto, siempre y cuando se encuentre dentro de nuestro radio de acción.

Figura 4.7: Vista Attack One

El otro modo es el conocido como “Attack All”, en el cual, tras pulsar el botón de atacar, se van mostrando en la tabla todas las señales que han sido atacadas desde que se ha pulsado el botón, ya que se realiza un ataque masivo a todas las señales detectadas hasta ahora.

Figura 4.8: Vista Attack All

A continuación se procederá a explicar el software utilizado para desarrollar dicha interfaz.

4.4.1. GTK 3

Para el desarrollo de la interfaz gráfica se decidió utilizar el software Gnome Graphical Toolkit (GTK), el cual proporciona una biblioteca de componentes gráficos multiplataforma para desarrollar interfaces gráficas de usuario.

En nuestro caso optamos por utilizar la versión 3 de esta biblioteca, porque a pesar de haber disponible ya una versión 4, contábamos con una mayor documentación de la versión 3, lo que sería de gran ayuda ante cualquier problema que pudiera surgir en el desarrollo de nuestra interfaz.

Las principales ventajas por las que se decidió utilizar GTK para desarrollar nuestra interfaz son su amplia y completa colección de widgets que proporciona todo lo necesario para diseñar dicha interfaz, además de que consideramos relevante que sea un proyecto Open Source y que esté mantenido por GNOME.

Para instalar la librería de GTK 3 se hace uso del manejador de paquetes “apt-get” con el respectivo libgtk-3-dev.

4.5. Tecnologías descartadas

Como en todo desarrollo de un proyecto hay que realizar una toma de decisiones la cual implica aceptación de ideas pero sobre todo el descarte de algunas de ellas. Este proyecto

cumple esta premisa. Se ha tenido que decidir las diferentes tecnologías a utilizar, como sería la planificación del proyecto, los distintos lenguajes de programación, etc.

En casi todas estas controversias se ha intentando escoger la mejor opción para el proyecto. Para ello siempre se ha intentado documentar lo máximo posible sobre el tema a tratar para tomar la mejor decisión. En caso de no haber sabido como solventarse, siempre hemos intentando buscar la opinión de los directores de este TFG. Por otro lado, también es cierto que algunas de estas confrontaciones de ideas se han resuelto solas, bien porque la opción dejaba de ser válida o porque era la única que quedaba para el problema a ocupar.

Se intentará aclarar algunas de las tecnologías descartadas que consideramos más relevantes, explicando el por qué y la opción que se escogió finalmente.

Durante el desarrollo de este proyecto hemos utilizado dos diferentes lenguajes de programación. Para escoger el lenguaje principal de desarrollo se listaron varios con unas ciertas condiciones como que fuese conocido por el equipo, que permitiese desarrollo multiplataforma... Finalmente el lenguaje escogido y que engloba el 95 % del proyecto es el lenguaje C. La elección de este fue por una sugerencia de los directores y nuestra posterior valoración, en la cual sacamos en claro que cumplía con bastantes de nuestras exigencias.

En el caso de la elección de una antena que nos permitiese recibir las señales finalmente escogimos la antena de Nooelec. La cual cumple con los requisitos necesarios para el proyecto. A parte, el paquete viene con tres antenas diferentes para abarcar diferentes distancias de escucha. Destacable el precio, ya que no es un módulo muy caro y entraba en presupuesto del proyecto, intentando seguir una línea low-cost. Aun así había módulos receptores más baratos que cumplían las condiciones pero con las prestaciones mucho más recortadas. Por ello su descarte, ya que no se aseguraba 100 % que funcionase correctamente a la hora de realizar las escuchas de los vehículos.

También existen modelos con la capacidad de escuchar y transmitir señales, como la conocida herramienta de hacking, HackRF ONE. Estos módulos son muy potentes y serían las herramientas perfectas para el proyecto. Pero el precio desorbitado de estas romperían con la estética low-cost que se quiere conseguir. El objetivo es que con el mínimo coste posible llegar a realizar el ataque a un vehículo.

Por ello finalmente se decidió seguir con una antena receptora y para poder transmitir utilizar otra antena que permitiera esto.

Hay muchos transceptores en el mercado que encajan en el presupuesto. Pero muchos de ellos no cumplen las características necesarias para poder transmitir señales TPMS. Casi todos ellos cumplen la premisa de poder transmitir por la banda de frecuencia de 433Mhz, pero lo que no suelen cumplir es permitir la modulación FSK tan necesaria para el proyecto. Finalmente se escogió el chip Ti CC1101 que cumple la mayoría de las especificaciones requeridas.

Para hacer uso de todo este hardware de envío de señales es necesario un software para comunicar la antena con nuestra aplicación. La antena transmisora, como se indicó en puntos anteriores, se conecta a la Raspberry Pi mediante los puertos GPIO. Por ello para comunicar estos dos dispositivos hay que hacer uso de la interfaz de comunicación SPI.

Buscamos librerías que realizasen la comunicación y aprovecharan las máximas características posibles de la antena. Existen varias librerías de código abierto como CC1101[15] creada por el usuario *Space Teddy* o CC1101 MSP430 Energia Library v2[16] del usuario *abhra0897*. Ambas codificadas en C++, lenguaje que se puede fácilmente incorporar con código C como nuestra aplicación. Finalmente se descartaron estas ya que no permitían la modulación FSK de las señales enviadas por el transmisor.

Por ello seguimos buscando y encontramos la librería que hemos explicado en el punto de transmisión de señales, Python-CC1101. Como su propio nombre indica está desarrollada en Python, por ello para hacer uso de esta hay que realizar un pequeño código en este lenguaje y llamarlo desde la aplicación con los servicios que ofrece C.

Para controlar todas estas diferentes partes del proyecto se decidió hacer una interfaz que conectase todos los diferentes procesos y hacerle más simple la experiencia al usuario.

Se tomaron varias decisiones antes de desarrollar esta interfaz. La pregunta básica era el tipo de interfaz que haríamos, si una interfaz gráfica o una interfaz de consola. Descartamos la interfaz consola ya que lo que buscamos es que esta aplicación la pueda usar cualquier usuario y este tipo puede asustar a un usuario medio. Al decantarnos por una interfaz gráfica debatimos sobre que tecnologías utilizar para el desarrollo de la misma. Se pensó en utilizar tecnologías Web o directamente librerías propias de C para desarrollar este tipo de de interfaces. Descartamos el uso de tecnologías web ya que creíamos que la comunicación con el software en C sería demasiado complejo. Finalmente se escogió la librería GTK.

Todos estos puntos que se han explicado junto con el desarrollo e investigación sobre este proyecto han sido documentados en esta memoria.

A la hora de elegir herramientas para la elaboración de esta memoria hubo varias propuestas. Se pensó en las típicas herramientas de ofimática como Word, LibreOffice, Google Docs, etc. Todas muy válidas y prácticas para nuestro propósito. Pero finalmente las descartamos y elegimos una herramienta un poco más fuera de lo común pero muy potente, Overleaf.

La herramienta Overleaf es un editor colaborativo de LaTeX que se utiliza para escribir, editar y publicar documentos científicos. Este procesador de texto permite maquetar texto de una manera fácil, con unos resultados profesionales y de alta calidad.

Esta memoria se realizó utilizando este software ya que nos permitía seguir los estándares obligatorios para la realización de un TFG, haciendo uso de la plantilla “TeFloN X” desarrollada por David Pacios Izquierdo.

Capítulo 5

Validación del sistema

El objetivo de esta parte del proyecto es verificar el correcto funcionamiento de todos los protocolos desarrollados. Para ello se elaboró un exhaustivo plan de pruebas mediante el cual se pudiera comprobar de una manera sistemática que todos los protocolos que se habían desarrollado actuaban de la forma esperada y conseguíamos el objetivo final buscado.

Para la elaboración de este plan de pruebas, se optó por dividir el mismo en varias partes, al igual que se ha hecho con otros capítulos de esta memoria, por lo que contaremos con una parte en la que se analiza la recepción de la señal. Esta parte tiene un doble protagonismo en este plan de pruebas, ya que se tiene que realizar en dos ocasiones. Una primera vez al inicio, para poder decodificar la trama que se desea replicar y una segunda vez después de realizar la trasmisión, para verificar los datos enviados. Esta segunda vez también ha sido nombrada en alguna ocasión a lo largo de esta memoria como análisis de la señal. La segunda parte es la construcción de la señal, una vez decodificada, y después se encuentra una tercera etapa en la que se verifica la trasmisión de la señal.

A continuación, se detallan cada una de las partes de este plan de pruebas y posteriormente se realiza un análisis de cada protocolo, separando cada análisis en un caso de uso propio para cada uno.

En la parte de la recepción y/o análisis de la señal, el objetivo principal es validar los datos que se están recibiendo. Para ello, en primera instancia y como ya se ha indicado anteriormente en esta memoria, se hizo uso del Github oficial de RTL433 [8], con el que podemos observar los diferentes campos que deben aparecer una vez decodificado cada protocolo y la longitud que deben tener los mismos. Además, para algunos de los campos también indica los valores más frecuentes que se reciben. Teniendo este análisis teórico de cada protocolo, se procede a realizar una escucha utilizando nuestro receptor Nooelec NESDR SMArTee v2 Bundle y el software RTL433 filtrando las señales recibidas para captar únicamente los protocolos que realmente nos interesan.

Una vez recibida la señal deseada y esta ha sido decodificada por el software RTL433, se comprueba que todos los campos mostrados son correctos, validando tanto la longitud de los mismos, como los valores que estos poseen, comprobando así que no se han producido errores en el proceso de decodificación.

Para la segunda parte, en este caso la construcción de la trama, se validan todos los

campos que se han explicado en el capítulo anterior, verificando que se están siguiendo los parámetros establecidos para la correcta formulación de la señal, respetando siempre el tamaño de cada campo, la correcta codificación de los mismos y los valores conocidos que deben ser los mismos para cada trama.

En esta parte es muy importante tener en cuenta que ciertos valores de los campos son inmutables, como son el preámbulo, el fin de trama y, el más importante, el ID. Estos tres parámetros no pueden ser modificados, ya que imposibilitarían el establecimiento de la comunicación con el receptor deseado.

Con respecto al resto de parámetros que componen la trama, se deben verificar siguiendo lo indicado en el capítulo anterior, validando que todos los datos con los que estamos intentando construir la trama son correctos. Es importante seguir las especificaciones del capítulo anterior, porque como hemos visto hay varios datos que para transmitirlos se necesita aplicar una modificación sobre el valor real que detecta el sensor, por lo que será necesario comprobar que estamos realizando dichas modificaciones correctamente a la hora de construir nuestra trama, además de comprobar que esas modificaciones se han realizado adecuadamente en el proceso de decodificación de las tramas recibidas y los datos recibidos son coherentes.

También es necesario comprobar la correcta realización del proceso de comprobación de errores. Como se ha explicado en el capítulo anterior, estos protocolos cuentan con un campo específico mediante el cual se valida la corrección de la trama. Es importante validar que este cálculo, bien sea un CRC o un checksum, se está realizando correctamente para evitar que nuestra trama sea rechazada por el receptor en caso de que tenga algún error. Además, algunos protocolos también cuentan con ciertos campos que contribuyen en este proceso de validación de errores. Estos campos también podrían ocasionar este rechazo de la trama por parte del receptor por lo que son revisados exhaustivamente para evitar que se pueda producir cualquier rechazo una vez hayamos trasmitido nuestra trama modificada.

Nuestro plan de pruebas continúa examinando el proceso de transmisión de la señal. Esta es sin duda la parte más compleja de verificar ya que hay que hacer uso de varios software adicionales, tanto para la realización de la transmisión como para la verificación de la misma.

En esta parte es necesario comprobar que estamos aplicando la correcta codificación a la señal que queremos transmitir. En el caso de este proyecto se utilizan únicamente las codificaciones Manchester y Manchester diferencial, por lo que necesitaremos un software para captar nuestras transmisiones que sea capaz de decodificar estas dos codificaciones. Por tanto, haciendo uso de la propia codificación que nos ofrece el transmisor elegido, se envía la trama codificada y se verifica que la codificación se realiza de manera correcta, captándola con el receptor y haciendo uso de un software especializado que permita realizar la decodificación de la misma. De esta forma se verifica que los datos recibidos coinciden con los datos enviados y que no se ha producido ningún error ni en la codificación, ni en la transmisión, además de que no han aparecido interferencias que hayan podido ocasionar algún error en la recepción del mensaje.

Además de la codificación, también es necesario verificar que se está aplicando correctamente una modulación 2-FSK, para la cual también se hará uso de la opción que nos brinda el propio transmisor y, al igual que en el punto anterior, necesitaremos que el

software encargado de la recepción y decodificación de la señal soporte dicha modulación 2-FSK.

Una vez explicados todos los aspectos a tener en cuenta dentro de nuestro plan de pruebas, se va a documentar de manera independiente, utilizando un caso de uso para cada protocolo desarrollado, las diferentes comprobaciones y validaciones realizadas para cada módulo que compone nuestro desarrollo.

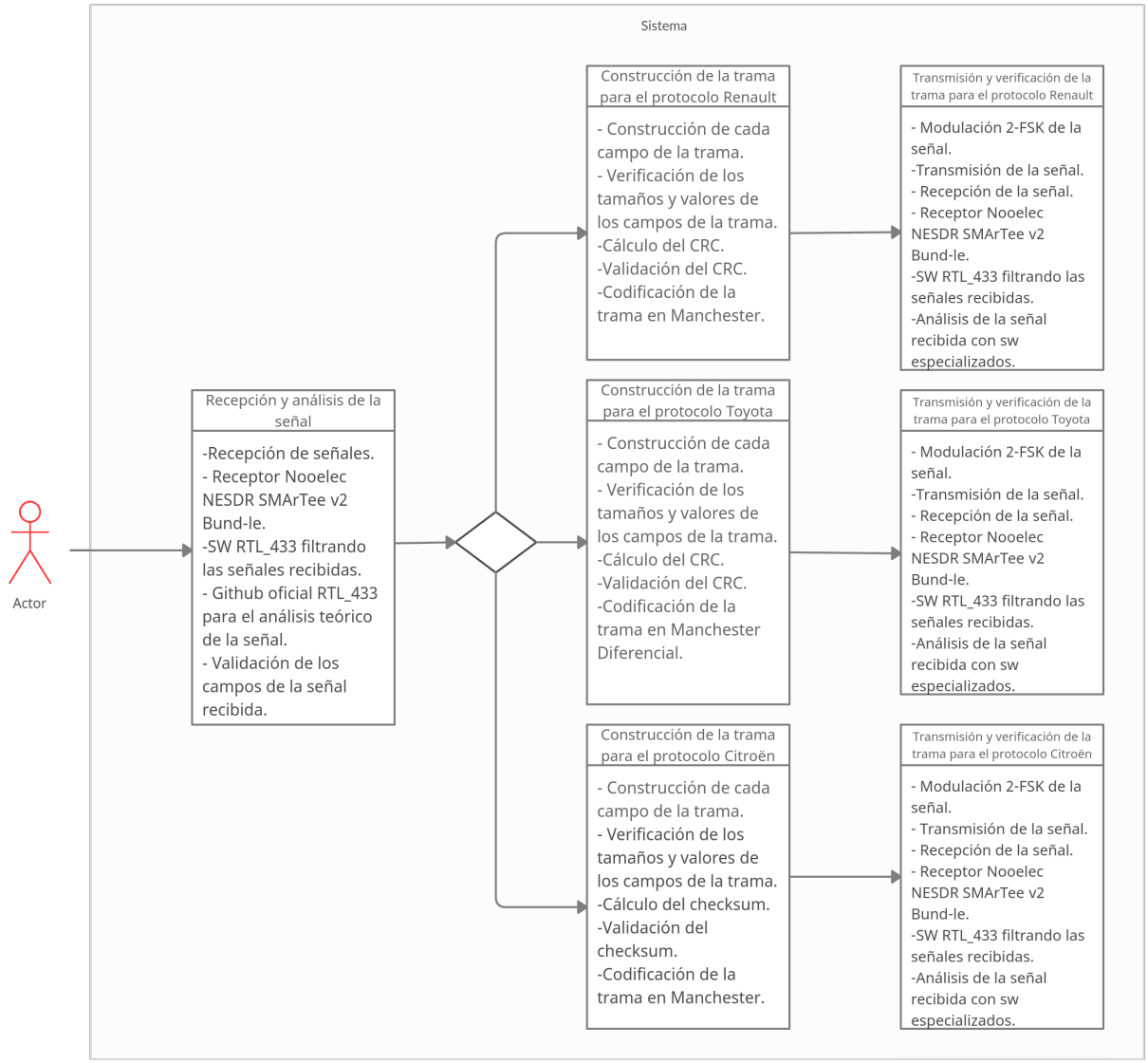

Figura 5.1: Diagrama del proceso de validación

5.1. Caso de uso del protocolo Citroën

A lo largo de este punto se desarrolla un caso práctico en el que se recibe una señal correspondiente a un sensor Citroën, se construye una señal con datos falsos y seguidamente se envía una señal falsa con el fin de engañar a la ECU.

Este proceso comienza con el arranque de la aplicación creada. Una vez iniciada la

aplicación, la recepción de posibles señales comenzará una vez se haga click en el botón "Attack One".

Figura 5.2: Vista principal

5.1.1. Recepción, decodificación y análisis de la señal

Una vez iniciada la recepción de señales en la vista "Attack One", se recibe una señal de un dispositivo Citroën, mostrando en la aplicación información sobre el modelo, en este caso Citroën, el id del dispositivo y la presión y la temperatura correspondiente a la rueda en la que se encuentra este sensor.

Figura 5.3: Vista recepción sensor Citroën

Como se puede observar en la imagen, se ha recibido una señal de un sensor correspondiente a un modelo Citroën, cuyo ID es 8a58f9a2 y que marca una temperatura de 15 grados Celsius y una presión de 242.792 kPa. Estos datos se obtienen al recibir y decodificar la señal mediante RTL433.

Una vez recibida y decodificada se seleccionan los datos que se desean mostrar en la aplicación ya que hay información que no es relevante mostrar, como es el "state", los "flags", el "repeat" o la "maybe_battery". Para poder comenzar la construcción de la trama falsa, simplemente hay que seleccionar en la tabla la señal, en este caso la que aparece de Citroën, y seguidamente pulsar en "Attack".

5.1.2. Construcción de la trama

Para comenzar a construir la señal con datos falsos, en este caso asociar a la presión y a la temperatura valores anormales que pudieran hacer saltar la alerta, se ejecuta una serie de funciones para convertir los datos a binario. En este caso contamos con una señal con los siguientes valores:

- **"model":** Citroen
- **"state":** 13
- **"id":** 8a58f9a2
- **"flags":** 0
- **"repeat":** 1
- **"pressure_kPa":** 242.792
- **"temperature_C":** 15.000
- **"maybe_battery":** 56

Todos estos datos se mantendrán con el mismo valor, a excepción de la temperatura y la presión que valdrán 0 y 1 respectivamente. Es importante tener en cuenta que hay que aplicar las correspondientes operaciones a la presión y a la temperatura relacionadas con sus respectivas constantes, como se explica en el capítulo 4.2 Generación de la señal, en concreto el punto 4.2.2 que se centra en Citroën, donde también se puede encontrar el formato de la trama.

El primer paso es convertir a binario cada campo, con lo que obtenemos los siguientes valores:

- **"model":** Citroen
- **"state":** 00010011
- **"id":** 10001010010110001111100110100010
- **"flags":** 0000
- **"repeat":** 0001
- **"pressure_kPa":** 00000001
- **"temperature_C":** 00110010

- **“maybe_battery”:** 00111000

El siguiente paso consiste en concatenar todas las partes siguiendo el orden y la estructura de la trama. De esta forma obtenemos una trama como la siguiente:

Trama: 000100111000101001011000111110011010001000000001000000010011001000111000

Llegados a este punto, el siguiente paso es fundamental, ya que se trata de calcular correctamente el checksum de la trama. Para ello, y como se explica en el punto 4.2.2, se realizan las XOR entre los correspondientes grupos de 8 bits. Con esto obtenemos el siguiente resultado:

Checksum: 10000011

Seguidamente, el checksum se concatena a la trama, obteniendo así la trama completa para poder realizar la codificación, en el caso de Citroën en Manchester. Una vez codificada la trama completa se obtiene la siguiente trama codificada:

Trama en Manchester: 0101011001011010100101011001100101100110100101011010101010010111010011001010110010101010101010101100101010101010101100101101001011001010110100101010110010101010101011010

Una vez obtenida la trama codificada en Manchester, hay que concatenar al principio de la trama el preámbulo, el cual tiene el valor 01010101010101010101010101010110 y al final de la trama el “End of frame” con el valor 01111110, esto está explicado en el punto 4.2.2.

Con esto último realizado, la trama completa ya está lista para pasar a la parte de la transmisión.

5.1.3. Transmisión de la señal

Una vez se tiene construida la trama completa y esta ya ha sido codificada, es el momento de empezar el proceso de transmisión. El objetivo es enviar la señal que se ha generado en anteriores pasos.

Para enviar la señal se hace uso del transceptor Ti CC1101. Para comunicarse con este dispositivo por los puertos SPI se utiliza la librería python-cc1101.

Todo este proceso significará la culminación de los procesos anteriores, consiguiendo transmitir la señal falsa generada anteriormente. Para el usuario la mayor parte del proceso va a ser invisible. Simplemente para indicar que se está atacando se mostrará una respuesta de pantalla, es decir, se mostrará un GIF junto a la deshabilitación del botón de ataque para evitar la saturación del programa mientras está realizando el ataque.

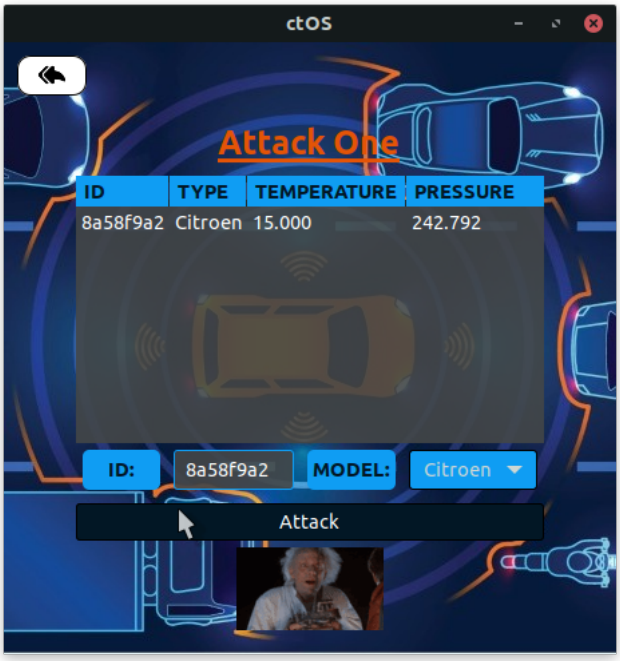

Figura 5.4: Transmisión de la señal de un Citroën

5.1.4. Recepción y decodificación de la señal trasmitida

Este paso de validación se resume en analizar la señal que se recibe tras realizar el envío y solo es necesario para depurar las señales construidas. Para estas comprobaciones utilizamos dos softwares de escucha diferentes, URH y GNU Radio. Para comprobar si la señal recibida es correcta simplemente capturamos la señal, la demodulamos y realizamos el análisis de esta comprobando si los bits corresponden con la señal transmitida.

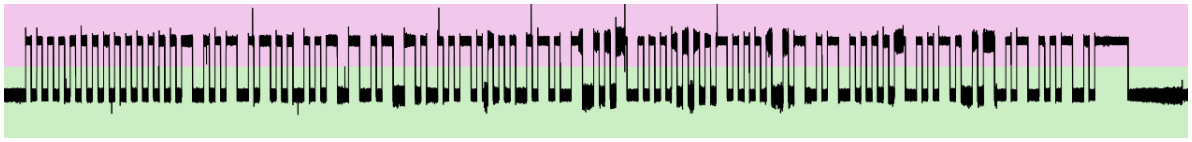

Figura 5.5: Recepción de la señal de Citroën con el software de URH

5.2. Caso de uso del protocolo Toyota

En esta parte se desarrolla un caso de uso en el que se recibe una señal de un vehículo Toyota, se genera una señal falsa siguiendo el datagrama del fabricante y finalmente se envía esta señal falsa para engañar a la ECU.

Este caso comienza en la pantalla inicial de nuestra aplicación, como en el anterior caso de uso explicado. Para empezar la interacción con las señales RF hay que hacer click en el botón “Attack One”.

Figura 5.6: Vista principal

5.2.1. Recepción, decodificación y análisis de la señal

Una vez iniciada la recepción de señales en la vista "Attack One", se recibe una señal de un dispositivo Toyota, mostrando en la aplicación información sobre el modelo, en este caso Toyota, el id del dispositivo y la presión y la temperatura correspondiente a la rueda en la que se encuentra este sensor.

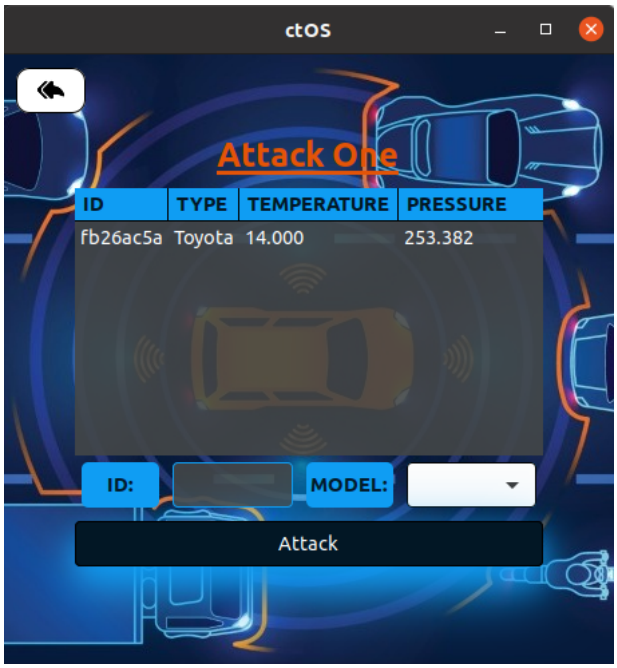

Figura 5.7: Vista recepción sensor Toyota

Como se puede observar en la imagen, se ha recibido una señal de un sensor correspondiente a un modelo Toyota, cuyo ID es fb26ac5a y que marca una temperatura de

14 grados Celsius y una presión de 253.382 kPa. Estos datos se obtienen al recibir y decodificar la señal mediante RTL433.

Una vez recibida y decodificada se seleccionan los datos que se desean mostrar en la aplicación ya que hay información que no es relevante mostrar, como es el "status". Seguidamente se selecciona en la tabla la señal, en este caso la que aparece de Toyota, y seguidamente pulsar en "Attack", para así poder comenzar la construcción de la trama falsa.

5.2.2. Construcción de la trama

Para construir la señal con datos falsos se siguen los mismos pasos que en el caso de uso anterior. En este caso contamos con una señal con los siguientes valores:

- **"model":** Toyota
- **"id":** fb26ac5a
- **"status":** 128
- **"pressure_kPa":** 253.382
- **"temperature_C":** 14.000

La temperatura y la presión pasarán a valer 0 y 1 respectivamente, teniendo en cuenta que hay que aplicar las correspondientes operaciones a la presión y a la temperatura relacionadas con sus respectivas constantes.

El primer paso es convertir a binario cada campo, con lo que obtenemos los siguientes valores:

- **"model":** Toyota
- **"id":** 11111011001001101010110001011010
- **"status":** 10000000
- **"pressure_kPa":** 00100000
- **"temperature_C":** 00101000

A parte de estos campos, en la trama Toyota y como se puede ver en la estructura en el punto 4.2.1, el status se divide en dos partes: la primera formada por el primer bit y la segunda formada por los 7 bits restantes. Además de esto también se puede observar otro campo que es la presión invertida. De esta forma, y observando la estructura correspondiente a Toyota, obtenemos una trama como la siguiente:

Trama: 11111011001001101010110001011010100100000001010000000000011011111

El siguiente paso es esencial, ya que se trata de calcular correctamente el crc de la trama, como se explica más en detalle en el punto 4.2.1. Con esto obtenemos el siguiente resultado:

CRC: 00010010

A continuación se concatena el CRC a la trama para poder realizar la codificación en Manchester diferencial. Una vez codificada la trama completa se obtiene la siguiente trama codificada:

Trama en Manchester diferencial: 0011001100101100101011010100110100101101001101010100101100101110100101010101010100101101010101010101001101001100101010110101001011

El último paso, al igual que en el caso de uso anterior, consiste en concatenar al principio de la trama codificada en Manchester diferencial el preámbulo, el cual tiene el valor 01010101001111, y al final de la trama el "End of frame" con el valor 111 en este caso, esto está explicado en el punto 4.2.1.

Con esto último realizado, la trama completa ya está lista para pasar a la parte de la transmisión.

5.2.3. Transmisión de la señal

Después de haber realizado todos los pasos anteriores, habiendo generado una señal falsa siguiendo el datagrama de un vehículo Toyota correctamente, se podrá empezar el proceso de transmisión de una señal Toyota.

A nivel de la aplicación, la transmisión comenzará tras haber completado el proceso de construcción de la trama. Para el usuario se mostrará un GIF para indicar que el proceso está en ejecución.

Para enviar la señal se hace uso del transceptor Ti CC1101. Para comunicarse con esta antena por los puertos SPI se utiliza la librería python-cc1101.

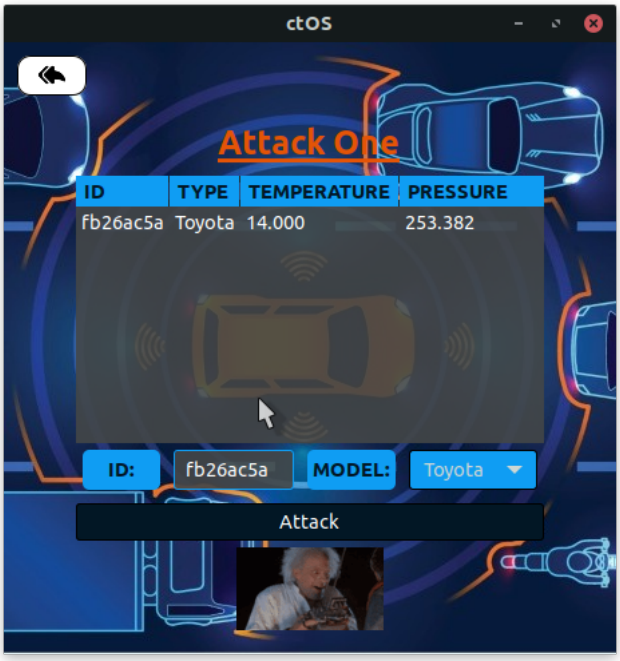

Figura 5.8: Transmisión de la señal de un Toyota

5.2.4. Recepción y decodificación de la señal trasmitida

Una vez transmitida la señal, para validar el proceso capturamos la señal para ver si se está realizando correctamente el proceso. Una vez recibida, simplemente la demodulamos con GNU Radio y/o con URH(como se puede ver en la imagen). Una vez demodulada podremos comprobar si la señal está bien capturada.

Este paso es simplemente para comprobar que todo el proceso de transmisión se realiza correctamente. A nivel de la aplicación esta funcionalidad no está contemplada.

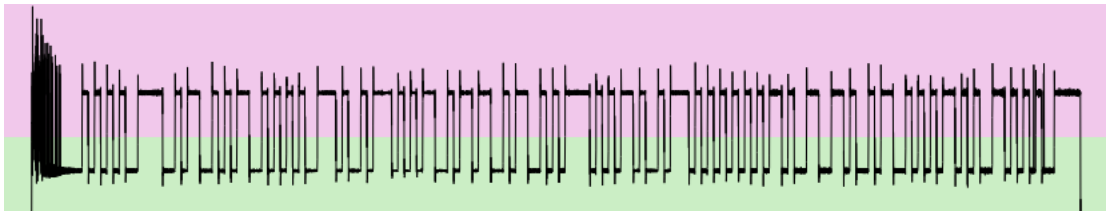

Figura 5.9: Recepción de la señal de Toyota con el software de URH

5.3. Caso de uso del protocolo Renault

A lo largo de este punto se desarrolla un caso práctico en el que se recibe una señal correspondiente a un sensor Renault, se construye una señal con datos falsos y seguidamente se envía una señal falsa con el fin de engañar a la ECU.

Este proceso comienza igual que los anteriores, es decir, una vez se haga click en el botón “Attack One”.

Figura 5.10: Vista principal

5.3.1. Recepción, decodificación y análisis de la señal

Se recibe una señal de un dispositivo Renault, mostrando en la aplicación información sobre el modelo, en este caso Renault, el id del dispositivo y la presión y la temperatura correspondiente a la rueda en la que se encuentra este sensor.

Figura 5.11: Vista recepción sensor Renault

Como se puede observar en la imagen, se ha recibido una señal de un sensor correspondiente a un modelo Renault, cuyo ID es 87f293 y que marca una temperatura de 25 grados Celsius y una presión de 202.5 kPa. Estos datos se obtienen al recibir y decodificar la señal mediante rtl 433.

Una vez recibida y decodificada se seleccionan los datos que se desean mostrar ya que hay información que no es relevante mostrar, como son los "flags" o el "unknown". Se selecciona en la tabla la señal, en este caso la que aparece de Renault, y pulsando en "Attack" comenzará la construcción de la señal falsa.

5.3.2. Construcción de la trama

Para comenzar a construir la señal con datos falsos se siguen los mismos pasos en los casos de uso anteriores. En este caso contamos con una señal con los siguientes valores:

- **"model":** Renault
- **"flags":** 34
- **"pressure_kPa":** 202.500
- **"temperature_C":** 25.000
- **"ID":** 87f293

- **“unknown”:** 0xffff

La temperatura y la presión pasarán a tener el valor de 0 y 1 respectivamente, aplicando posteriormente las correspondientes operaciones a la presión y a la temperatura relacionadas con sus respectivas constantes. Además el ID pasará a escribirse siguiendo un formato “Little Endian”.

El primer paso es convertir a binario cada campo, con lo que obtenemos los siguientes valores:

- **“model”:** Renault
- **“flags”:** 110100
- **“pressure_kPa”:** 0000000001
- **“temperature_C”:** 00011110
- **“ID”:** 1001001111100101000011
- **“unknown”:** 1111111111111111

El siguiente paso consiste en concatenar todas las partes siguiendo el orden y la estructura de la trama. De esta forma obtenemos una trama como la siguiente:

Trama: 1101000000000001000111101001001111110010100001111111111111111111

El siguiente paso es fundamental consiste en calcular correctamente el CRC de la trama, como se explica en el punto 4.2.3. Así obtenemos el siguiente resultado:

CRC: 00100100

Una vez obtenido el CRC se concatena a la trama, al igual que en el caso de uso anterior, obteniendo así la trama completa a codificar en Manchester. Una vez codificada la trama completa se obtiene la siguiente trama codificada:

Trama en Manchester: 1010011001010101010101010101011001010110101010011001
0110010110101010101001011001100101010111010101010101010101010101010101010101010101
01101010010110

Al igual que en los anteriores casos de uso, hay que concatenar al principio de la trama codificada el preámbulo, el cual tiene el valor 01010101010101010101010101010110 y al final de la trama el “End of frame” con el valor 0000, esto está explicado en el punto 4.2.3.

Con esto último realizado, la trama completa ya está lista para pasar a la parte de la transmisión.

5.3.3. Transmisión de la señal

La transmisión es la culminación de los procesos anteriores. Habiendo generado ya satisfactoriamente la señal de Renault simplemente transmitimos este datagrama.

Para poder transmitir una señal desde la aplicación es necesario haber construido correctamente la señal falsa. La aplicación iniciará el proceso y mostrará un indicativo visual (GIF) de que la transmisión se esta ejecutando.

Para enviar la señal se hace uso del transceptor Ti CC1101. Para comunicarse con esta antena por los puertos SPI se utiliza la librería python-cc1101.

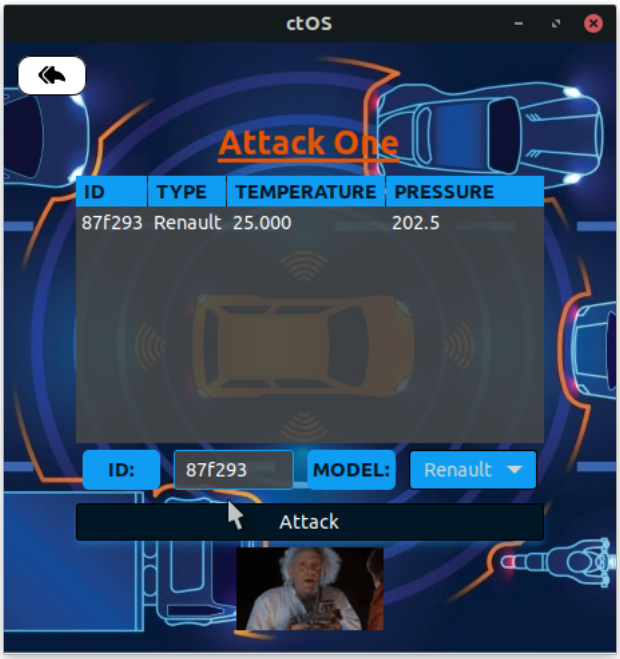

Figura 5.12: Transmisión de la señal de un Renault

5.3.4. Recepción y decodificación de la señal trasmitida

Para validar que la señal que hemos transmitido en el proceso anterior se ha transmitido correctamente, es necesario capturar la señal enviada con los sofwares de GNU Radio o URH. Es necesario capturar y demodular la señal captada para comprobar a nivel de bits si esta se ha transmitido correctamente.

Este proceso no es un servicio de nuestra aplicación, solamente es necesario para depurar las señales.

Capítulo 6

Discusión, conclusiones y trabajo futuro

Una vez llegado a este punto en el desarrollo del proyecto y habiéndose comprobado que es posible replicar las señales TPMS de los vehículos cuyo modelo sea Citroën, Toyota o Renault, en este capítulo se va a realizar un análisis del contexto de estas tecnologías empleadas para realizar la réplica de la señal.

En la primera parte del capítulo se abordan los posibles inconvenientes o problemas con los que se puede encontrar un usuario en el momento de tratar de replicar la señal siguiendo el desarrollo explicado en este proyecto. A su vez, también se proponen análisis a estos problemas tratando de buscar una solución para los mismos.

En esta primera parte también se exponen posibles situaciones donde sería útil el uso de esta tecnología y cual es la mejor forma de abordarlas teniendo en cuenta los inconvenientes detallados en esta parte.

En la segunda parte del capítulo se exponen las conclusiones que se han obtenido teniendo en cuenta todo lo mencionado en el punto anterior y valorando a su vez las distintas pruebas realizadas durante el desarrollo del proyecto. También se proponen posibles alternativas para evitar este tipo de ataques que hemos expuesto durante el desarrollo de este proyecto.

En la última parte del capítulo se habla de las posibles mejoras o desarrollos que se podrían añadir a este proyecto, en caso de que se decidiese continuar con el desarrollo del mismo. Esta serie de mejoras ampliarían aún más los desarrollos logrados en este proyecto consiguiendo una aplicación más completa y versátil.

6.1. Discusión

Para comenzar, se detallan algunos aspectos relevantes a tener en cuenta a la hora de recibir la señal, lo que a priori es la parte más sencilla de este proyecto.

En primer lugar se debe tener en cuenta el rango de detección de la antena con la que estemos trabajando, como bien se indica en el estudio “Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study” [4], la recepción de mensajes provenientes de sensores TPMS puede ser correctamente

recibida hasta una distancia de 10 metros con respecto al vehículo que los está emitiendo haciendo uso de una antena de bajo coste. Además, esta distancia puede aumentarse hasta los 40 metros incorporando al sistema un amplificador básico de bajo ruido. Por lo tanto, en el caso de que se buscase aumentar aún más el rango de detección, sería necesario optar por antenas que contasen con un mayor alcance y por ende antenas con un coste superior, o hacer uso antenas direccionales o incluso contar con un mayor número de antenas omnidireccionales. También cabría la posibilidad de añadir otro tipo de amplificadores con unas características superiores que permitieran alcanzar los rangos de distancia deseados.

Estos aspectos relativos al rango de detección son también aplicables al rango de emisión donde se deberían tener en cuenta todos estos detalles para conseguir que el vehículo deseado reciba la señal que se le está tratando de enviar.

Una vez se ha valorado la distancia a la cual puede encontrarse el vehículo para que se permita la detección de la señal, es importante tener en cuenta la frecuencia de emisión de la misma. En el estudio anterior encontramos que según las regulaciones estipuladas por la NHTSA estos sensores transmiten la información de presión cada 60 a 90 segundos, sin embargo, realizando pruebas de detección hemos comprobado que algunos sensores llegan a transmitir incluso cada 30 segundos, por lo que consideraremos un abanico que abarque desde los 30 a los 90 segundos.

Este intervalo de transmisión es muy importante, ya que considerando que el receptor se encuentre en un punto fijo y el vehículo se encuentre en movimiento sería importante tener en cuenta este intervalo de transmisión así como la velocidad a la que circula el vehículo, ya que debe circular a una velocidad superior a 40 km/h para que el sensor comience a transmitir. Además no hay que olvidarse de la distancia a la que nos encontramos del mismo para poder garantizar la recepción de la señal.

Otro aspecto a tener en cuenta a la hora de recibir una señal serían las diferentes variaciones de los protocolos que se pueden encontrar a la hora de recibir una señal TPMS. Atendiendo a lo observado en diversos estudios se puede concluir que se pueden encontrar más de 147 variaciones de protocolos, por lo que sería necesario ampliar los protocolos que este proyecto es capaz de detectar para así poder cubrir un mayor número de vehículos detectados.

En relación con dichas variaciones, se quiere destacar los posibles preámbulos que se pueden detectar, ya que consideramos que es un campo muy importante porque se encarga de establecer la conexión. Entre los preámbulos más comunes encontramos el 55 55 55 56 o aa aa aa aa a9, en función de la polaridad del montaje del sensor, pero luego también se pueden encontrar preámbulos menos comunes que nos dificultan el desarrollo de nuestro proyecto como pueden ser el 11111 10 perteneciente a Pacific PMV-107J usado por Toyota USA o el 01010101001111 00110011 perteneciente a Pacific PMV-C210 empleado en las versiones europeas de Toyota, incluso preámbulos como el 0001111110 perteneciente a Pacific TPMS.

Ahora bien, sabiendo que contamos con tantas variaciones vamos a centrarnos en los tres protocolos desarrollados en este proyecto y destacar otro aspecto relevante sobre los mismos. El primer protocolo desarrollado es Toyota, pero es importante remarcar que no solo los vehículos Toyota llevan este sistema TPMS, también podemos encontrar este sistema en otros modelos como puede ser Mazda, ya que los sensores Toyota son fabricados

por Pacific Industrial Corp. y por TRW Automotive, empresas que no fabrican dichos sensores en exclusiva para Toyota por lo que pueden ser utilizados en otras marcas.

Con respecto al segundo protocolo estudiado, en este caso Citroën, ocurre algo similar. No solo encontramos sensores Citroën en los vehículos de la marca, sino que otras marcas como pueden ser Peugeot, Fiat o Mitsubishi también emplean este modelo de sensor en sus vehículos, por lo que todo esto nos dificulta enormemente saber a qué vehículo pertenece la señal recibida en caso de que nos encontremos en un espacio donde haya varios vehículos susceptibles de poseer el mismo modelo de sensor.

Con respecto al tercer protocolo estudiado, en este caso Renault, ocurre algo similar a lo observado en los dos protocolos anteriores. Además de encontrar este tipo de sensores en vehículos de la marca Renault, también han sido observados en algunos modelos de la marca Dacia, por lo que una vez más se nos dificulta el hecho de conocer de que vehículo proviene la señal recibida.

El último aspecto a destacar sería relativo a la emisión de la señal con los valores alterados para buscar engañar a la ECU. No bastaría con enviar una única señal con los valores alterados, sino que sería necesario enviar una señal periódica, ya que en caso de enviar una única señal, la ECU recibiría dicha señal y alertaría al conductor de la presión deficiente en dicho neumático. Sin embargo, en caso de proseguir el vehículo en movimiento, se volverían a recibir las señales con la presión real y adecuada, haciendo que nuestro intento de engañar a la ECU no consiguiese el resultado esperado, cosa que no pasaría con una emisión periódica ya que se seguiría detectando esa presión deficiente y conseguiríamos nuestro objetivo.

En relación a esto, en caso de realizar una emisión única también podría darse el caso de que la alerta de presión deficiente se mantuviese, pudiendo considerar así que hemos logrado el objetivo de este proyecto. Sin embargo, en este caso, el conductor del vehículo podría detenerse y comprobar la presión de la ruedas, detectando así que la presión real es correcta y quedando este aviso de presión deficiente como un posible fallo en la medición.

Con respecto al mínimo número de transmisiones necesarias para lograr que el vehículo muestre la alerta de presión deficiente, de acuerdo con lo expuesto en el articulo “Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study” [4] y tras realizar un exhaustivo análisis por nuestra parte, se puede concluir que no es suficiente con la realización de una transmisión única, ya que se ha comprobado que cuando se realiza una única transmisión buscando engañar a la ECU, ésta responde al paquete enviado devolviendo dos señales de activación, provocando de esta manera que el sensor real del vehículo transmita la presión real del neumático, imposibilitando la consecución del objetivo buscado.

Por lo tanto, de acuerdo con las investigaciones realizadas en el artículo citado anteriormente, se puede concluir que serían necesarias 4 transmisiones con una presión deficiente para conseguir producir una alerta por parte de la ECU. También se observa que estas transmisiones pueden ser realizadas en un mínimo de 1 segundo por lo que se puede concluir que cada transmisión debe distanciarse de la anterior en tan solo 225 milisegundos. Además, se observa que si se aumenta la distancia entre transmisiones superando los 4 segundos entre ellas no sería posible conseguir engañar a la ECU y producir esa alerta.

Esto se puede interpretar como que los sensores TPMS poseen una ventana para la detección de presiones deficientes de 240 milisegundos, es decir, 225 milisegundos entre transmisiones más 15 milisegundos que es el tiempo que se tarda en transmitir un paquete TPMS de acuerdo a lo observado en investigaciones anteriores. Teniendo esto en cuenta se puede concluir que para la iluminación de la alerta luminosa por parte de la ECU, se establecen 4 ventanas de 240 milisegundos, las cuales deben ser todas positivas, es decir, haber recibido un paquete con la presión deficiente, independientemente del número de paquetes recibidos con una presión correcta, dichas ventanas se restablecerán en caso de que no se reciba ningún paquete con una señal deficiente transcurridos 4 segundos.

También se puede afirmar que aumentar la velocidad de transmisión de los paquetes, transmitiéndolos con una diferencia menor a 240 milisegundos, continuaría produciendo el efecto deseado manteniendo la señal de alerta encendida. Sin embargo, en caso de aumentar esa diferencia a un valor superior, siendo siempre inferior a 4 segundos, produciría que la luz de alerta aparezca y desaparezca de manera intermitente, no logrando así el objetivo buscado.

A pesar de esto, es importante destacar que si una vez se ha conseguido encender la alerta por parte de la ECU, esta generalmente desaparecerá después de aproximadamente 6 segundos desde el momento en que dejemos de transmitir los mensajes falsos, lo cual confirma lo que mencionábamos anteriormente de la necesidad de un envío periódico de paquetes para poder así lograr unos objetivos más robustos.

6.2. Conclusiones

Una vez llegado a este punto se va a tratar de valorar si se ha conseguido el objetivo inicial de este proyecto. Dicho objetivo consistía en conseguir engañar a la ECU de un vehículo transmitiendo una señal falsa desde un dispositivo externo, simulando la señal que emiten los propios sensores TPMS de dicho vehículo.

Según lo explicado en la sección previa de este capítulo, y verificándolo además con lo expuesto en el capítulo de Arquitectura del sistema y más explícitamente en el capítulo de Validación del sistema, se puede concluir que sería posible lograr el objetivo buscado y conseguir engañar a la ECU.

Ahora que tenemos claro que sería posible engañar a la ECU, se va a tratar de analizar cuán perjudicial puede llegar a ser esta vulnerabilidad de los sistemas TPMS en caso de ser explotada con fines maliciosos.

Lo primero que se debe tener en cuenta es la complejidad de lograr completar dicho engaño, puesto que como ya se ha explicado a lo largo del proyecto, aparecen ciertas limitaciones al tratar de utilizar herramientas de bajo coste. Estas limitaciones van desde el rango de detección de las señales, que como hemos visto se puede ampliar fácilmente hasta los 40 metros con herramientas de bajo coste como serían una antena y un amplificador de bajo ruido, pero que se convierte en un proceso más costoso a la hora de querer aumentar esta distancia haciendo necesario adquirir otro tipo de antenas más potentes o con unas características superiores, aumentando así el coste del desarrollo del proyecto.

También es importante tener en cuenta, como se indicó en la discusión, la gran cantidad

de posibles variaciones de los protocolos, hecho que dificulta enormemente el desarrollo de nuevos protocolos, ya que hay que realizar el proceso de decodificación y aplicar ingeniería inversa para desglosar los diferentes campos que componen la trama, en un proceso largo y tedioso.

Algo a remarcar y que ha supuesto un verdadero reto, el cual no se ha conseguido superar del todo, es el tema de la tasa de bits, ya que si el emisor transmite una cierta cantidad de bits por segundo y el receptor espera recibir menos bits en ese mismo tiempo, el receptor no será capaz de decodificar la información que se está transmitiendo. Esto es algo bastante complejo de depurar debido principalmente a la falta de información, ya que también los diferentes protocolos que hay pueden tener tasas de bits distintas, lo que complica aún más la situación. Este ha sido sin duda el mayor problema que hemos encontrado a la hora de transmitir e intentar que RTL_433 lo decodificara como una señal generada por un TPMS real.

Lo último que se va a valorar, y a priori una de las partes más importantes para conseguir el objetivo, es el hecho de que no basta con enviar una sola trama sino que sería necesario enviar una trama periódica. Esto dificulta enormemente lograr nuestros objetivos porque significa que debemos estar transmitiendo durante un mayor tiempo, por lo que en una hipotética situación real en la que nos encontráramos en un punto fijo, el vehículo al que tratemos de atacar a medida que avance el tiempo pasará a estar cada vez más lejos de nuestra posición, volviendo una vez más al primer problema mencionado, el rango de detección y transmisión, por lo que volveríamos a tener la necesidad de aumentar los costes del proyecto para contar con herramientas más potentes u optar por una posible solución alternativa que consistiría en no ubicarnos en un punto fijo, sino que estuviéramos ubicados en algún dispositivo móvil, permitiendo así una vez comenzado el ataque poder seguir al vehículo y lograr mantener nuestro objetivo en el tiempo.

Una vez se valorados estos principales inconvenientes, surge el debate de si es viable y rentable o no este tipo de ataques. Para responder a esta pregunta se van a valorar las posibles consecuencias que tendría un posible ataque como este.

Como bien sabemos, en la actualidad la mayoría de vehículos solo emiten una alerta luminosa cuando reciben una señal de presión deficiente en alguno de los neumáticos, por lo que si se valora el ratio esfuerzo frente a resultados, parece obvio que no resulta rentable invertir ni tiempo ni esfuerzo ni dinero en este tipo de ataques. Sin embargo la cosa cambia según vamos analizando posibles vehículos de alta gama y posibles consecuencias que tendrá en un futuro cada vez menos lejano.

Se ha observado que ciertos vehículos de alta gama ya incorporan ciertos controles en relación a la presión de los neumáticos, por lo que ya no solo se produciría la alerta luminosa como en el resto de vehículos sino que también en algunos de estos vehículos se produce una limitación de la velocidad máxima. Esto produce que el interés en desarrollar este tipo de ataques aumente y por ende comience a parecer más rentable la investigación sobre ellos.

Llegamos al punto donde posiblemente encontremos el mayor interés en la investigación sobre este tipo de ataque, como bien se expone en uno de los estudios que se han expuesto a lo largo de esta memoria, en un futuro, los vehículos tendrán la capacidad de, en función de las señales recibidas por parte de los sensores TPMS, variar la velocidad de giro de las ruedas en función de la presión detectada para así mantener la trayectoria

correcta del vehículo. Por lo que es aquí donde este tipo de ataques logra mayor relevancia convirtiéndose en un problema realmente importante y en el cual es necesario trabajar para solucionar puesto que si dichos vehículos variasen la velocidad de giro atendiendo a los valores detectados mediante la transmisión de una señal falsa podrían producirse situaciones de grave peligro para la seguridad.

Sin embargo, también hay que destacar que no todo sería negativo en cuanto al aprovechamiento de esta vulnerabilidad, ya que nos permitiría detener un vehículo en caso de producirse una situación peligrosa que lo requiriese, disminuyendo progresivamente la velocidad de giro de la ruedas hasta conseguir detenerlo. Hecho que podría ser utilizado, por ejemplo, por agentes de la autoridad, como podrían ser la policía, reduciendo enormemente los riesgos que conlleva una posible persecución policial.

Un último aspecto relativo a las vulnerabilidades de estos sistemas TPMS viene derivada por el hecho de que estos sistemas transmiten sus mensajes en claro, por lo que sabiendo que los ID que poseen los sensores del vehículo son inmutables se puede realizar una especie de rastreo o geolocalización haciendo uso de estas señales permitiendo así conocer la ubicación de un vehículo en cualquier momento siempre y cuando conozcamos los ID de los sensores de sus neumáticos.

Por tanto, se puede concluir que con vistas a medio largo plazo, resulta enormemente interesante trabajar sobre este tipo de ataques tratando de solventar la vulnerabilidad, así como tratar de encontrar posibles usos beneficiosos de la misma.

Una vez analizados los posibles intereses en investigar sobre este tipo de ataques así como los posibles peligros que podrían ocasionar, se proponen varias posibles alternativas para tratar de proteger estos sistemas.

La principal mejora que se debería realizar para contrarrestar estos posibles ataques es el hecho de que los sensores transmitan en claro. Actualmente estos sensores no poseen ningún tipo de cifrado ni de mecanismo alternativo para proteger las comunicaciones, por lo que se debería implementar algún tipo de cifrado, encriptado o alguno de los distintos mecanismos expuestos en capítulos anteriores de esta memoria.

Otra mejora que se propone, y a priori de las más sencillas, sería sustituir los checksum que poseen algunos protocolos como mecanismo de control de errores por códigos Cyclic Redundancy Check (CRC), que ya se están utilizando en diversos protocolos y que, como bien es conocido, son bastante más seguros que los anteriormente comentados checksum permitiéndonos minimizar los posibles errores de transmisión que pudiesen producirse.

Ahora bien, es importante tener en cuenta que para abordar esta vulnerabilidad sería necesario no solo comenzar a implementar estas mejoras en los nuevos sistemas TPMS desarrollados e incorporarlos en los nuevos vehículos, sino que también sería necesario reemplazar o reparar todos los sistemas TPMS existentes en los vehículos a día de hoy, lo que supone un enorme esfuerzo y un enorme gasto que tendría que ser asumido bien por las empresas desarrolladoras de este tipo de sensores, las cuales asumieran los costes de reemplazar estos sensores, o bien deberían ser asumidos por el consumidor final, quien debería acudir a un taller o concesionario para reparar los sistemas TPMS de su vehículo y hacerse cargo del gasto que ello conlleva. Por lo que en vistas a lo analizado anteriormente y volviendo a hacer incapié en los posibles riesgos que a día de hoy supone esta

vulnerabilidad, consideramos que lo más rentable de acuerdo con el ratio coste frente a beneficio sería realizar estas mejoras en los nuevos sensores y comenzar a implantarlos en los nuevos vehículos, con el objetivo de reducir los posibles problemas que se podrían producir en el futuro cuando el vehículo sea capaz de realizar acciones en base a los datos recibidos mediante estos sensores, evitando así lo que sí supondría un grave riesgo.

6.3. Trabajo futuro

Tras haber realizado este trabajo de investigación, nos damos cuenta de que este proyecto tiene potencial para ser ampliado en multitud de direcciones, haciendo que abarque más posibilidades y que solvente algunas de las situaciones en las que podría haber algún que otro problema.

Para comenzar, la primera opción y la más obvia, es ampliar el proyecto aumentando el número de protocolos para los que puede funcionar este proyecto. Actualmente está pensado para los protocolos de Toyota, Citroën y Renault, sin embargo, y como se puede observar en la página de manual de RTL 433, hay sistemas TPMS de otros vehículos que también pueden ser captados en esta frecuencia, como puede ser Ford. Por lo tanto, si se realiza un estudio sobre la estructura de la trama de estos protocolos, podría ampliarse el abanico de protocolos para los cuales se puede realizar un ataque enviando una señal falsa.

La siguiente opción consiste en realizar una segunda versión de este prototipo funcional del proyecto, que sea más compacta y manejable. Hasta este punto consta de una raspberry, el receptor y el emisor, además de haber realizado una interfaz gráfica para poder manejar todo con facilidad. Ahora bien, ¿cómo se puede ampliar esto?

Lo primero sería encontrar una caja en la que se pueda acoplar nuestra raspberry, además de incluir una pantalla táctil con la que poder gestionar todo directamente desde ahí. Por tanto, tras buscar en diversos sitios, encontramos algunas opciones en Amazon que incluían una caja para nuestro modelo de raspberry y una pantalla táctil por un precio entre 30 y 40 euros. Sin embargo, existe la posibilidad de realizar tu propio diseño mediante una impresora 3D y adquirir solamente la pantalla por 15-20 euros.

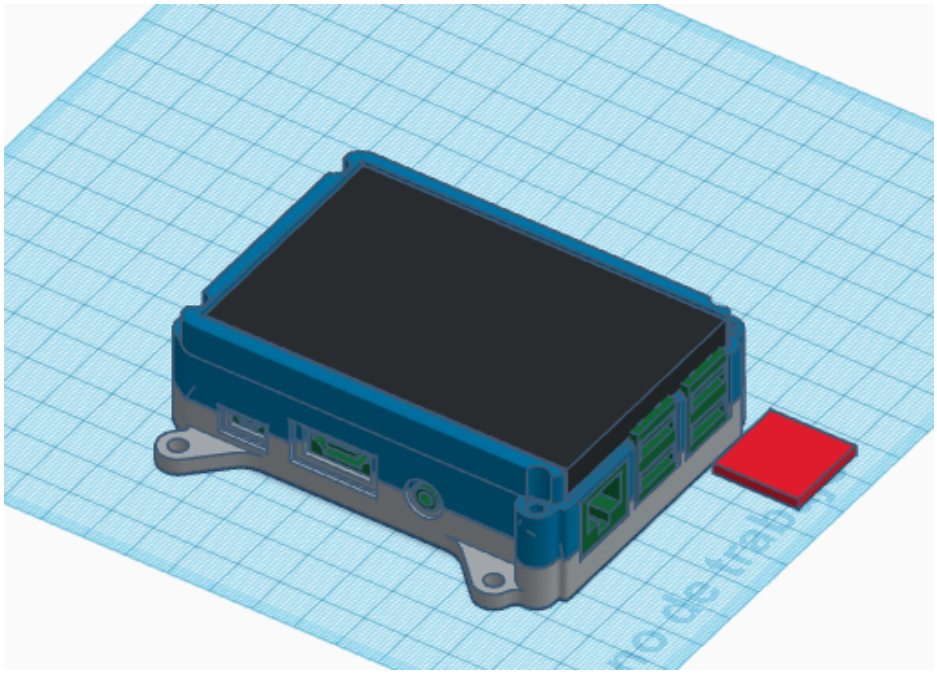

Figura 6.1: Ejemplo de caja con pantalla y raspberry diseñado con Tinkercad

A esto se le podría añadir una caja mayor en la que poder acoplar este diseño y la antena de una manera más cómoda. También existe la posibilidad de adquirir una pantalla más grande y cambiar el diseño por completo, pero esto supondría un coste mayor.

Continuando con las opciones de ampliar el proyecto, se encuentra la idea de poder saber dónde está el vehículo del que estamos recibiendo la señal. Esta idea surge del problema que plantea la siguiente situación: tenemos la necesidad de mandar una señal falsa a un vehículo Toyota, con el fin de que haga al coche reducir la velocidad o algo por el estilo, entonces alrededor se encuentran otros vehículos Toyota, por tanto al recibir las señales no sabremos a qué vehículo pertenece cada señal, lo que supone un problema. Como solución surge la idea de que mediante algún sistema de medición de la intensidad de la señal recibida se pueda saber a qué vehículo corresponde cada señal, o quizá de alguna otra manera que desconocemos.

Siguiendo este hilo, surgió otra dificultad que se podría solucionar de la misma manera. La cual se da cuando se reciben varias señales del mismo vehículo debido a que las ruedas tienen diferentes ID. Como al recibir la señal no hay nada que indique a que rueda pertenece dicha señal, existiría alguna forma de que, en función de como se pusiera la antena y mediante una medición de la intensidad de la señal, pudiéramos saber de que rueda proviene esta señal, haciendo de esta forma que el ataque sea más preciso.

Otra posibilidad que existe, aunque no está relacionada con los vehículos directamente, es el poder abrir una puerta de garaje ya que estas señales también se transmiten por la frecuencia de 433MHz. Sin embargo, la forma de conseguir un ataque con éxito no es tanto de conseguir generar una señal con valores falsos, sino que consiste en recibir la señal que transmite la llave y posteriormente enviar la señal que hemos recibido para así lograr abrir dicha puerta. Esto mismo se puede llegar a aplicar a las llaves de los coches, ya que funcionan de una manera similar. Por ejemplo en el caso de Ford en el Github de RTL 433 se puede encontrar un fichero que aporta información sobre cómo identificar

dichas señales, sin embargo se menciona que puede que no esté actualizado.

Capítulo 7

Discussion, conclusions and future work

Once reached that point in the development of the project and has proved that is possible to replicate the TPMS signals of the vehicles whose models are Citroën, Toyota or Renault, in this chapter, an analysis of the context of the used technologies will be developed to make the replicate of the signal.

The possible problems that a user may find at the moment when is trying to replicate the signal following the development process that is explained in this project, are going to be tackled in this first part of the chapter. Besides, we will find an analysis of these problems to find a solution for them.

In addition, in this part are shown possible situations where could be useful the use oh this technology and what is the best way of approach them thinking about the problem which is been explained in this part.

On the second part of the chapter, conclusions that have been obtained following all that have been mention in the previous point are present and assess the different tests that have been made during the development of this project. Moreover, we will find some possible alternatives to avoid these attacks that are shown during the development of the project.

The last part of the chapter is shown possible improvements or developments that could be added to this project, to continue with the development of it. This series of improvements may extend the developments achieved in this project looking for a more complete and diverse application.

7.1. Discussion

To start, some key aspects that are necessary to keep in mind at the time to receive the signal are detailed, beforehand this is the most simple part of the project.

First of all, is necessary to keep in mind the detection range of the antenna that are working with, as is explained in the paper “Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study” [4], the message detection that provides from a TPMS sensor can be correctly received up to a distance of

10 meters from the vehicle that is sending, using a low-cost antenna, besides this distance can be increased up to 40 meters with the integration of a basic low noise amplifier.

Therefore, in case that we look to increase the detection range is necessary to opt for antennas that have a large range consequently antennas that have a bigger price, or use directional antennas or even use a greater number of omnidirectional antennas. As well as it is possible to add another type of amplifier with better characteristics that allow us to reach the distance ranges desired.

These relative aspects about range detection can be applied to the transmission range where is necessary to keep in mind all of these details to achieve that the desired vehicle receives the signal which is trying to transmit.

Once the distance in which is possible stay be the vehicle to could be possible the signal detection have been appreciated, it is important to keep in mind the transmission frequency of the signal, in the previous paper find that according to the stipulated regulation by the NHTSA these sensors transmit the information of the pressure every 60 to 90 seconds, but testing the detection we have verified that some sensors even transmit every 30 seconds, so we consider a range that encompasses from 30 to 90 seconds.

This transmission interval is so important because considering that the receptor is located at a fixed point and the vehicle is in circulation it is important to keep in mind this transmission interval and the velocity at the vehicle is circulating because it is necessary to circulate at a speed greater than 40 km/h to the sensor begin to transmit. Besides, do not forget the distance at which we are situated from them to ensure the reception of the signal.

Another aspect to keep in mind at the time of receiving the signal is the different variations of the protocols that we can be found when are trying to receive a TPMS signal. According to the observations of multiple pieces of research, it is possible to conclude that can be found more than 147 protocol variations, so it is necessary to increase the protocols that this project can detect to reach a bigger number of detected vehicles.

Concerning these variations, it is necessary to highlight the possibles preambles that can be detected, because it is considered an important field because it is responsible for establishing the connection. Among the most usual preambles we find the 55 55 55 56 or the aa aa aa aa a9, depending on the polarity of the assembly of the sensor, but then can be found lees usual preambles that difficult us the development of our project how can be 1111 10 belonging to Pacific PMV-107J which is used for Toyota USA or 01010101001111 00110011 belonging to Pacific PMV-C210 which is used in the European versions of Toyota, even preambles like the 0001111110 belonging to Pacific TPMS.

Having said that, knowing that have a huge variety of protocols we will focus on the two protocols that are developed in this project and highlight another relevant aspect about them. The first developed protocol is Toyota, but it is important to emphasize that not only the Toyota vehicles have this TPMS system because the Toyota sensors are manufacture by Pacific Industrial Corp. and by TRW Automotive, companies that do not manufacture these sensors exclusively for Toyota so it can be used by others brands.

Regarding the second studied protocol, which is Citroën, occurs something similar. Not only find Citroën sensors in the vehicles of the brand, others brands like can be Peugeot, Fiat or Mitsubishi use this sensor model in their vehicles too, so this makes so

difficult to identify at which vehicle belongs the received signal in case that we are in a place where there are lots of liable vehicles that can have the same sensor model.

Regarding the third studied protocol, in this case is Renault, occurs something similar with the observed in the previous protocols. We can find this type of sensors in vehicles of Renault brand, however, it has been observed in some models of Dacia brand too. So other time difficult us to know which vehicle the signal comes from.

The last aspect that will be highlighted is relative to the transmission of the signal with the modified values searching to cheat the ECU. It is not enough to send a unique signal with the modified values, it is necessary to send a periodical signal because, in case of sending an only signal, the ECU receives this signal and alerts the driver that the insufficient pressure on this tire. However, in case of the vehicle continues in movement, the signals with the real and appropriate pressure would be received again, making that our attempt to cheat the ECU does not reach the expected result, a thing that does not occur with a periodical transmission because it continues detecting this insufficient pressure and we reach our goal.

Regarding this, in the case of sending a unique transmission could also be the case that the insufficient pressure alert keeps, being able to consider that we have reached the goal of the project. However, in this case, the driver of the vehicle could stop and verify the pressure of the tires, detecting that the real pressure is appropriate and remaining this insufficient pressure alert like a possible measuring mistake.

Regarding the minimum number of necessary transmissions to reach that the vehicle shows the insufficient pressure alert, according to the explained in the article Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study [4] and after making an exhaustive analysis by us, can be concluded that is not enough with the transmission of a unique signal, because it has been demonstrated that when we realize a unique transmission trying to cheat the ECU, it responses to the transmitted package returning two activation signals, causing by this way that the real sensor of the vehicle transmits the real pressure of the tire, making impossible to reach the searched goal.

Therefore, according to the investigations that were made in the previously mentioned article, it is possible to conclude that 4 transmissions with insufficient pressure are necessary to reach produce the alert by the ECU. Also has been observed that these transmissions can be made in a minimum of 1 second so we can conclude that each transmission should be a distance to the previous in just 225 milliseconds. Besides, has been observed that if the distance of the transmission is increased exceeding 4 seconds between them, could not be able to cheat the ECU and produce the alert.

This can be interpreted how the TPMS sensors have a detection window for insufficient pressures of 240 milliseconds, that is, 225 milliseconds between transmissions plus 15 milliseconds that is the time that takes the transmission of a TPMS package according to the shown in the previous investigations. Keeping this in mind it is possible to conclude that for the illumination of the light alert by the ECU, have been established 4 windows of 240 milliseconds, that all of them have to be positive, that is, have received a package with insufficient pressure, independently of the number of received packages with the correct pressure, these windows will be restored in case of it have not received any package with a deficient signal after 4 seconds.

Also can be affirmed that increase the transmission velocity of the packages, and transmit them with a difference smaller than 240 milliseconds, continues making the same desired effect keeping the alert signal active. However, in case of increase de difference to a value, always being smaller than 4 seconds, it will produce that the alert light appears and disappears intermittently, so do not reach the searched goal.

Despite this, it is important to highlight that if once have reached to turn on the alert by the ECU, generally will disappear after around 6 seconds from the moment in which stop transmitting false messages, which confirms that we have previous mentioned the necessity of a periodical package transmission to achieve more robust goals.

7.2. Conclusions

Once we have reached this point we will try to appreciate if the initial goal of this project has been reached. This goal consisted of achieve to cheat the ECU of a vehicle by the transmission of a false signal from an external device, simulating the signal that the own TPMS sensors of the vehicle transmit.

According to that have been explained in the previous section of the chapter, and verifying with the explanation which is in the System Architecture chapter and more explicitly in the System Validation chapter, it is possible to conclude that the researched goal can be achieved and it is possible to cheat the ECU.

Now that we are clear that could be possible to cheat the ECU, it is going to try to analyze how much detrimental would be this vulnerability of the TPMS systems in case of being used for spiteful purposes.

The first thing to keep in mind is the complexity of reach this cheat because how it has been explained throughout the project, appears some limitations when we are trying to use low-cost tools. These limitations go from the signal detection range, which can be increased up to 40 meters using low-cost tools like an antenna and a low noise amplifier, but it makes into a more difficult process at the time of increase this distance making it necessary to acquire another types os antennas more powerful or with better characteristics, increasing by this way the cost of the development of the project.

Besides it is important to keep in mind, how was explained in the discussion, the huge quantity of possible protocols variations difficult extremely the development of new protocols because it is necessary to make the decoding process and apply reverse engineering to itemize the different fields that formed the frame by a large and tedious process.

Something that is important to remark and that it has supposed a true challenge is the fact of the bits rate and it has not been completely overcome, because if the transmitter uses a certain quantity of bits per second and de receiver expect to receive fewer bits in the same time, the receiver will not be able to decode the information that is being transmitted. This is something that is a lot complicated to debug, mainly due to lack of information, because the different protocols that exist can have different bit rates, so the situation becomes even more difficult. This has been the major problem that we have found at the time to transmit and try to decode the signal using RTL_433 like a signal generated by a real TPMS sensor.

The last thing to be appreciated and beforehand one of the most important parts

to reach the goal is the thing that is not enough to transmit a unique frame but it is necessary to transmit a periodical frame. This thing complicate extremely to achieve our goals because it means that it is necessary to be transmitted during a greater time, so in a hypothetical real situation in which we are in a fixed point, the vehicle that we are trying to attack at a time that progresses the time, it will be placed further near our position, returning other time to the first problem that has been mentioned, the detection and transmission range, so we will have the necessity again of increase the cost of the project to have more powerful tools or opt for a possible alternative solution that consists of do not be in a fixed point, so we should be in a mobile device, allowing by this way to once the attack have begun to follow the vehicle and achieve to keep our goal in time.

Once have been appreciated the principal problems, the debate appears about if viable and worthwhile or not this kind of attack. To answer this question, the possible consequences that can produce this kind of attack are going to be appreciated.

As well known, nowadays the major part of the vehicles only show a light alert when they receive an insufficient alert pressure of some tire, so if we appreciate the effort opposite results ratio, seems obvious that it is not worthwhile to invest time nor effort nor money in this kind of attacks. However, this changes when we analyze possible high-end vehicles and the possible consequences that would produce in the future less and less distant.

Have been observed that some high-end vehicles already include some controls about the tire pressure, so already it not only produces the light alert like the rest of vehicles but in some of these vehicles it also produces a maximum speed limit. This produces that the interest in developing this kind of attack increase and begin to seem more worthwhile the investigation of them.

We reach the point where possibly find the most interest of the investigation about this kind of attack, as well as shown in one of the studies that have been explained throughout this memory, in the future, the vehicles will have the ability to depend on the received signal from the TPMS sensors, they will be able to change the turn speed of the tires according to the detected pressure to maintain the correct trajectory of the vehicle. So it is here where this kind of attack achieves its maximum relevance and turn into a really important problem in which is necessary to work to solve it because if these vehicles change the turn speed according to the detected values of a false signal transmitted would produce very dangerous security situations.

However, it is necessary to highlight that not all the relative to the exploitation of this vulnerability is negative, it allows us to stop a vehicle in case of a dangerous situation that requires it, reducing gradually the turn speed of the tires until stop it. The fact that could be used, for example, by authority agents like the police, reducing hugely the risk of a possible police chase.

The last aspect relative to the vulnerabilities of the TPMS systems comes from the fact that these systems transmit their messages in plaintext, so knowing that the IDs of the sensors of the vehicle are immutable it is possible to realize some kind of search or geolocation using this signals allowing us to know the location of the vehicle anytime as long as we know the IDs of the sensors of his tires.

Therefore it can be concluded that if looks to medium long term, it turns out to be

hugely interesting to work about this kind of attacks trying to resolve this vulnerability and try to find possible beneficial uses of it.

Once have been analyzed the possible interests in investigating this kind of attack and the possible risks that can be produced, it will be proposed some alternatives to protect these systems.

The main improvement that should be realized to counteract the possible attacks is the fact that the sensors transmit in plaintext. Nowadays these sensors do not have any cipher type nor any alternative mechanism to protect the communications, so should be implemented any type of cipher or encryption or anyone of the different mechanisms that have been explained in previous chapters of this memory.

Another improvement that is proposed and could be one of the most simple is that the checksum that has some protocols acting like error handling mechanism should be replaced by CRC codes, that are being used by various protocols yet and as well known, it is much safer than the checksums that were previously been mentioned allowing us to minimize the possible transmission errors that could produce.

Having said that, it is important to keep in mind that for tackle this vulnerability would be necessary not only to began to implement these improvements in the new developed TPMS systems and incorporate them into the new vehicles, but it is necessary to replace or repair all the TPMS system that already exists in the vehicles nowadays, so this suppose a huge effort and a huge cost that should be assumed by the developer companies of this type of sensors, which should assume the cost of the replacement of the sensors or should be assumed by the final consumer, who should go to a garage or an authorized dealer to repair the TPMS systems of his vehicle and should take care of the cost that it will bring. So if we look at the previously analyzed and re-emphasizing the possible risks that nowadays signify this vulnerability, we consider that the most worthwhile according to the ratio cost opposite benefit should be to realize these improvements in the new sensors and begin to implant them in the new vehicles with the goal of reducing the possible problems that would be produced in the future when the vehicle would be able to take actions based on received data from these sensors, avoiding what would be a serious risk.

7.3. Future work

After having done this investigation work, we realize that this project has the potential to be extended in multiple directions, making that encompass more possibilities and resolve some situations in which could have another problem.

To start, the first option and the most obvious is to extend the project increasing the number of protocols for which this project can works. Actually, it is thought for the Toyota, Citroën and Renault protocols, however, and how can be observed in the manual page of RTL 433, there are TPMS systems of other vehicles that can be received too in this frequency, like can be Ford. So, if some studies are realized about the structure of the frame of these protocols, could increase the variety of protocols for which can realize an attack transmitting a false signal.

The next option consist on realize the second version of this functional prototype of

the project, which may be more compact and easy to use. Up to this point, it consists of a raspberry, a receiver, and a transmitter, besides having done a graphical interface to can manage easily all of this. Now well, how can this be extended?.

The first thing will be to find a box where our raspberry can be fit together, moreover, it should include a touch screen on which we can manage everything directly from there. So after searching in various places, we find some options on Amazon which include a box for our raspberry model and a touch screen with a price between 30 to 40 euros. However, exists the possibility of making your own design using a 3D printer and add only the touch screen for 15 to 20 euros.

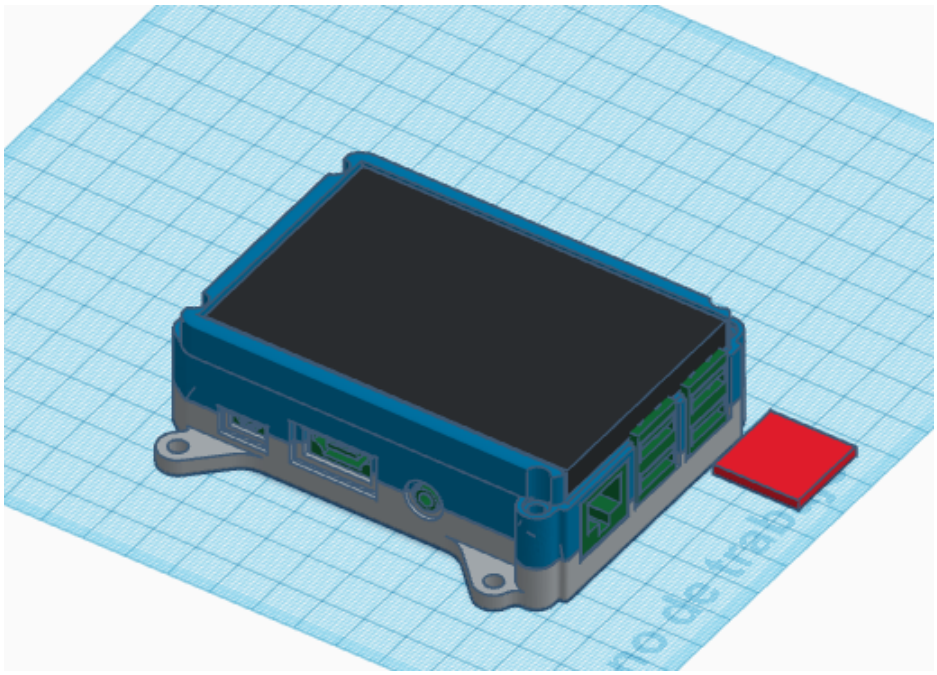

Figura 7.1: Example of a Tinkercar designed box whit a screen and a Raspberry

To this can be added a bigger box where can be fit together this design and the antenna in a more comfortable way. Besides exists the possibility to acquire a bigger screen and change the design completely, but it will suppose a bigger cost.

Continue with the options of extending the project, we find the idea of can know where is the vehicle of which we are receiving the signal. This idea comes up from the problem that shows the following situation: we need to transmit a false signal to a Toyota vehicle, to make the car slow down its velocity or something like this, so near they are other Toyota vehicles therefore at the time to receive the signals we would not know which vehicle each signal belongs to, so it is a problem. How solution comes up with the idea of using some measuring system of the intensity of the received signal could be known to which vehicle each signal belongs to, or by another way that we do not know.

Following this thread, appears another difficulty that could be solved in the same way. That occurs when we receive some signals from the same vehicle because all tires have different IDs. How when we receive the signal there is nothing that indicates from which tire is each signal, may exist any may to, in the function of how could be placed the antenna and by the measuring of the signal intensity would be known to which tire the

signal comes from, making by this way that the attack would be more precise.

Another possibility that exists, although it is not directly related to the vehicles, is the ability to open a garage door because these signals are transmitted too by the 433MHz frequency. However, the way to reach this attack successfully is not related to generating a signal with some false values, but it is about to receive the signal that is transmitted by the key and afterward transmit the signal that has been received to achieve opening this door. The same thing could be applied to the car keys because it works similarly. For example, in the case of Ford, in the RTL 433 Github, we can find a file that provides some information on how to identify these signals, however, it is mentioned that could be not updated.

Capítulo 8

Distribución del trabajo

8.1. Metodología

La metodología a seguir a la hora de realizar un proyecto es fundamental, ya que dos proyectos con los mismos recursos y con distintas metodologías difieren en tiempo y resultado. En un principio se desconocía el alcance o la viabilidad del proyecto, por lo que la metodología a seguir debía ser una que permitiera darle flexibilidad al proyecto, para así poder gestionarlo eficazmente. Debido a esto, se optó por seguir la metodología ágil.

La metodología ágil se caracteriza por la flexibilidad y capacidad de modificar el producto a lo largo del proyecto, lo que permite adaptar el trabajo al entorno para que se obtengan beneficios, reduciendo los costos y maximizando la producción. Utiliza el trabajo en equipo para ofrecer mejoras constantes. Dentro de la metodología ágil se pueden diferenciar varios tipos o marcos, como "Scrum", "XP", "Kanban", etc.

Entre todos los tipos, el que más se ajustaba a nuestra forma de trabajar era la metodología Kanban. Kanban es una palabra japonesa formada por Kan, que quiere decir visual, y Ban, que significa tarjeta. Por lo tanto, Kanban hace referencia a las tarjetas visuales.

Esta metodología es muy sencilla, se puede actualizar y los equipos de trabajo la pueden asumir sin problema. Al ser un método visual permite que a golpe de vista se conozca el estado de los proyectos, además de asignar nuevas tareas de manera muy efectiva. Para aplicarlo, es necesario un tablero de tareas con el que poder mejorar el trabajo y tener un ritmo sostenible.

A diferencia de otros tipos de metodologías ágiles, en Kanban las tareas pueden tener una fecha límite y no es tan estricto el periodo determinado en un inicio para cada tarea. El hecho de trabajar en pequeños bloques que son las tarjetas, permite ser mucho más resolutivos y reducir los tiempos de desarrollo.

Para gestionar correctamente las tareas se ha utilizado la herramienta Trello, de la que explicamos los detalles en el punto 8.1.2.

Durante el desarrollo del TFG se realizaron diversas actualizaciones del proyecto para así llevar un control de versiones que permitiera gestionar el trabajo realizado eficien-

temente. La herramienta escogida para la gestión del control de versiones es GitHub, explicado más en detalle en el punto 8.1.1.

Una práctica habitual desde el principio del desarrollo son las constantes reuniones realizadas entre los miembros del equipo, así como alguna reunión cada cierto tiempo con los tutores. Todo esto se explica en detalle en el punto 8.1.3.

8.1.1. Control de versiones

Como ya se comentó anteriormente, con el fin de gestionar adecuadamente todo el código del proyecto, se decidió utilizar la herramienta GitHub[1]. Esta herramienta permite alojar el código del proyecto, además de poder administrarlo mediante el sistema de control de versiones, ordenando el código de cada una de las nuevas versiones para evitar confusiones. Así, al tener copias de cada una de las versiones del proyecto, no se perderán los estados anteriores cuando se va a actualizar. También registra quién realizó los cambios en las versiones.

La razón por la que se decidió utilizar esta herramienta es que ofrece las mejores características de este tipo de servicios sin perder la simplicidad, y es una de las más utilizadas del mundo por los desarrolladores. Además, a pesar de que GitHub es una de las principales herramientas enfocadas a crear proyectos abiertos de aplicaciones y herramientas caracterizándose así por sus funciones colaborativas, permite crear proyectos privados en los que solo tienen acceso aquellas personas que se inviten. De esta forma podemos trabajar sin que nadie externo pueda modificar el código.

Comentar que a la finalización del TFG, decidimos cambiar el proyecto a público para que así cualquiera que lo necesite pueda consultar el código. El enlace al código se encuentra en los anexos de la memoria.

8.1.2. Trello

Durante el desarrollo del TFG, se utilizó la herramienta Trello[2] con el fin del implementar el tablero de tareas de la metodología Kanban. Trello es un software que cuenta con interfaz web y con aplicación tanto para iOS como android que permite crear tableros formados por diversas listas a las que se pueden añadir tareas.

Estas tareas se pueden mover de una lista a otra, además de añadir comentarios en ellas o asignar dichas tareas a un miembro del equipo. También permite asignar colores en función del tipo de la tarjeta. Todo esto hace posible visualizar las tareas y el estado en el que se encuentran de una manera clara, sencilla y actualizada, lo que permite llevar una buena organización de las tareas durante el desarrollo del proyecto.

Para este proyecto se creó un tablero con el nombre “TFG” en el que se incluyen todas las tareas que fueron surgiendo durante el desarrollo. Para llevar una buena organización de dichas tareas, se crearon las siguientes listas:

- **Dudas:** Incluye las dudas que iban surgiendo para preguntar a los tutores en la siguiente reunión.

[1] GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git
[2] Dirección URL: https://trello.com

- **Lista de tareas:** Todas las tareas que van surgiendo para el desarrollo del proyecto y que no son dudas.
- **En proceso:** Lista de tareas que están en proceso de desarrollo. Al principio estas tareas se encuentran en la lista de tareas y cuando va a comenzar su desarrollo se asignan a uno o varios miembros y se desplazan a esta lista, además de poner una fecha límite para terminar su desarrollo.
- **En pruebas\Revisión:** Incluye todas las tareas que en un principio han terminado la etapa de desarrollo y pasan a las pruebas o a la revisión. En caso de que la tarea se haya desarrollado correctamente, pasaría a la última lista. Si la tarea presenta fallos, volvería a la lista anterior.
- **Hecho:** En esta última lista se incluyen todas las tareas finalizadas de forma correcta.

Como se mencionó anteriormente, Trello cuenta con la posibilidad de asociar colores a las tareas, lo que permite asignar un color a un tipo de tarea y así poder diferenciarlas de una manera rápida y visual. Los colores utilizados son:

- **Rojo:** Representa las tareas relacionadas con el desarrollo de la memoria.
- **Amarillo:** Se utiliza para representar tareas relacionadas con el hardware.
- **Verde:** Tareas relacionadas con el desarrollo del código referente a los modelos Citroën.
- **Azul oscuro:** Tareas relacionas con el desarrollo del código referente a los modelos Toyota.
- **Azul claro:** Tareas relacionadas con el desarrollo del código referente a los modelos Renault.
- **Naranja:** Se asigna a tareas del desarrollo de la interfaz para seleccionar el vehículo.
- **Violeta:** Hace referencia a las tareas más generales o que sirven para conectar varias partes del proyecto.
- **Gris oscuro:** Se utiliza para representar las tareas enfocadas al emisor, ya sea conectarlo, configurarlo, etc.
- **Rosa:** Hace referencia a las tareas relacionadas con las reuniones con los tutores.
- **Gris claro:** Se utiliza para representar las tareas relacionadas con la gestión del repositorio Github.

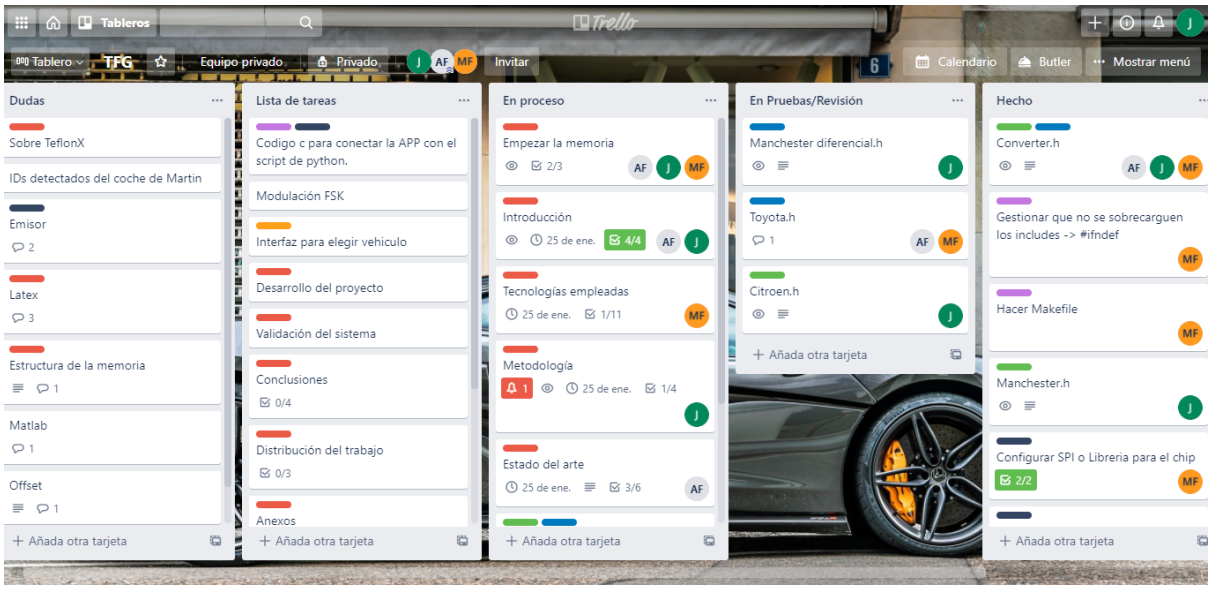

Figura 8.1: Tablero del proyecto en Trello

8.1.3. Reuniones

Desde el inicio del desarrollo del TGF se convirtió en práctica habitual realizar reuniones frecuentes entre los miembros del equipo para comentar los progresos, las dudas o los problemas que iban surgiendo. Por otro lado, estas reuniones servían para poner en común ideas para el desarrollo del proyecto, así como poder asignar las tareas pendientes a los distintos miembros, intentando siempre que el peso de trabajo fuera más o menos el mismo para todos. Estas reuniones, por lo general, se hacían a través de Discord[3], una herramienta que permite crear servidores privados para hablar por chat o por voz con los miembros, además de poder compartir todos pantalla simultáneamente.

Cada cierto tiempo se realizaban reuniones con los tutores para comentar el estado del proyecto, así como las dudas que habían surgido y que, en un primer momento, no se pudieron resolver en las reuniones entre los miembros del equipo.

Para concretar una de estas reuniones se mandaba un email a los tutores, comentando sobre qué tema se quería tratar en dicha reunión, ya fueran dudas, revisar la memoria, etc. Estas reuniones se llevaban a cabo mediante la plataforma Google Meet, donde los tutores creaban una sesión para un día y a una hora acordadas y enviaban el enlace para poder reunirnos.

Hay que comentar que debido a la situación que estamos viviendo causada por la COVID-19, prácticamente todas las reuniones han sido telemáticas, ya que viviendo en diferentes comunidades autónomas era muy complicado reunirse con frecuencia. Esto supuso que en algunas ocasiones se dificultara un poco el trabajo en grupo, pero con las herramientas que existen hoy en día se pudo sacar el proyecto hacia adelante.

[3]Dirección de descarga: https://discord.com/

8.2. Contribuciones

En esta sección se exponen las principales contribuciones que ha realizado cada miembro del equipo durante el desarrollo de este proyecto.

En primer lugar se van a destacar una serie de tareas en las que han participado todos los miembros del equipo, en mayor o menor medida en función de la tarea, posteriormente se exponen tareas mas concretas que han sido desempeñadas por cada miembro específico, siempre siendo ayudado y apoyado por el resto de integrantes del equipo.

Estas tareas que han sido desempeñadas en conjunto son:

- Investigación inicial tratando de recabar toda la información necesaria para comenzar a desarrollar el proyecto. Esta investigación inicial abarcó ámbitos desde lo puramente informativo cuyo objetivo era el desarrollo de esta memoria, como aspectos más técnicos cuyos fines eran la posibilidad de implementar los protocolos que se querían desarrollar, así como investigar sobre las diferentes tecnologías que era necesario conocer para permitir el correcto desarrollo del proyecto.
- Diseño del concepto de la aplicación. Se realizaron una serie de sesiones de trabajo conjuntas con el objetivo de esclarecer las principales características que debía tener la aplicación que se iba a desarrollar, centrándonos principalmente en las funcionalidades que se debían cumplir para lograr alcanzar el objetivo buscado en este proyecto.
- Desarrollo del código. Al tratarse de un proyecto que nos permitía trabajar en paralelo, bien sea desarrollando distintos protocolos a la vez, cada uno por uno de los diferentes miembros del equipo o bien centrarnos en otras partes como podían ser la detección o la transmisión, se buscó repartir de manera equitativa el desarrollo del código entre los diferentes miembros del equipo.
- Diseño y desarrollo de la interfaz. Se realizaron una serie de sesiones de trabajo conjuntas en las que se fueron definiendo unos bocetos de como se buscaba que fuese la interfaz y a partir de los cuales se fue desarrollando un primer prototipo inicial, el cuál se fue refinando hasta lograr la interfaz actual.
- Testing y validación. Se realizaron las distintas comprobaciones y verificaciones para asegurar que el código desarrollado funcionaba de manera correcta, así como que cada módulo que compone la aplicación desempeña su labor de manera correcta.
- Desarrollo de la memoria. Se estableció un reparto de los distintos capítulos y secciones que componen la memoria entre los distintos miembros del equipo, siendo estos revisados por el resto de miembros del equipo una vez habían sido completados, tratando así de corregir posibles erratas, así como añadir o eliminar cierta información que se considerase que era necesaria, o al contrario, que no debería estar en la memoria.
- Labores organizativas. Se buscó, a pesar de las dificultades que la actual crisis de la COVID-19 nos suponía, establecer la mejor forma de estructurar el trabajo así como las distintas reuniones desarrolladas y vías de comunicación entre todos los miembros del equipo, brindando así el apoyo que pudiera necesitar cualquier miembro del equipo en cualquier momento.

A continuación se van a destacar algunas de las tareas desarrolladas mas concretamente por cada miembro.

8.2.1. Jorge María Martín

Comenzó, en colaboración con Alberto, desarrollando una investigación sobre las posibles tecnologías útiles para el desarrollo del proyecto, evaluando posibles pros y contras de cada una de ellas para así decidir cuáles se usarían y cuáles serían descartadas. A continuación, su investigación se centró más concretamente en el lenguaje que se usaría posteriormente para la elaboración de la interfaz. En este caso el elegido fue GTK.

En cuanto a los distintos protocolos desarrollados en este proyecto, Jorge se centró en Renault, realizando la investigación previa y el posterior desarrollo del mismo, así como la validación y prueba de la trama desarrollada. Además de centrarse en la elaboración de este protocolo, también colaboró con el resto de sus compañeros en el desarrollo de todos los demás protocolos.

Participó también en la implementación de los distintos modos de ataque de los que se compone la aplicación, colaborando en la elaboración tanto del modo de ataque único como el modo de ataque masivo. En el desarrollo de estos diferentes tipos de ataques también participaron el resto de miembros del equipo, centrándose cada miembro en uno específico.

Relativo a la memoria se encargó del capítulo del resumen y palabras clave, así como su correspondiente traducción al inglés. Relativo al capítulo del estado del arte, desarrolló la primera parte del capítulo, que englobaría las secciones de sistemas de seguridad, radiofrecuencia y telecomunicación. Además se encargó dentro del capítulo de validación del sistema del correspondiente caso de uso del protocolo que había desarrollado, en este caso Renault. Con respecto al capítulo de la arquitectura del sistema se encargó de la elaboración de las secciones de recepción de la señal y desarrollo de la interfaz. Dentro de el capítulo de discusión, conclusiones y trabajo futuro, desarrolló la segunda parte de conclusiones. También elaboró dentro del capítulo relativo a la distribución del trabajo, la primera parte de metodología.

8.2.2. Alberto Rodríguez Fuentes

Como ya se ha mencionado anteriormente, colaboró junto con Jorge en la investigación sobre qué tecnologías se usarían y cuáles no en el desarrollo del proyecto. Posteriormente centró su investigación en las tecnologías encargadas de la recepción de la señal, más concretamente en el software RTL-433. Además, su investigación también se centró en recopilar el mayor número de información actual que pudiese ser útil para el desarrollo del proyecto, así como también información que fuese útil para la elaboración de la memoria.

En este caso, se centró en la investigación y elaboración del protocolo Citroën, realizando la correspondiente validación y pruebas para comprobar que la trama se generaba de manera correcta. Al igual que en el caso anterior, también colaboró con sus compañeros en la elaboración del resto de protocolos y la realización de las pruebas de los mismos.

En cuanto a los distintos modos que componen la aplicación, se centró en la elaboración del modo de ataque único, junto con la colaboración de su compañero Jorge.

En referencia a la memoria, desarrolló la segunda parte del capítulo del estado del arte, que englobaría las secciones de seguridad en la transmisión de señales, vulnerabilidades y ejemplos y trabajo previo, así como también se encargó dentro del capítulo de validación del sistema del correspondiente caso de uso del protocolo que había desarrollado, en este caso Citroën. Con respecto al capítulo de arquitectura del sistema se encargó de la sección de tecnologías descartadas. Además elaboró en el capítulo de discusión, conclusiones y trabajo futuro, esta primera parte de discusión, así como la correspondiente traducción al inglés del capítulo completo. También elaboró dentro del capítulo relativo a la distribución del trabajo, la segunda parte de contribuciones. Finalmente elaboró dentro del manual de usuario correspondiente al anexo I, la segunda parte de uso de la aplicación.

8.2.3. Martín García Fuentes

En este caso su investigación inicial se centró más en la parte de la comunicación, con una visión más específica en todo los relacionado con la Raspberry en vistas a que debíamos utilizar una para implementar la transmisión de la señal, así como distintas librerías que podíamos utilizar para facilitarnos esta transmisión.

Martín a su vez, se centró en la investigación y elaboración del protocolo Toyota, y colaboró de igual manera con el resto de miembros del equipo en la elaboración de los distintos protocolos. También se encargó de las pertinentes pruebas y validaciones para comprobar la correcta formación de la trama creada.

En cuanto a los distintos modos de juego que componen la aplicación, se centró en el modo de ataque masivo, en colaboración con Jorge como bien se ha indicado anteriormente.

También fue el principal responsable de la implementación de la transmisión, utilizando las librerías sobre las que había investigado en un primer momento. Además se encargó de las correspondientes validaciones y pruebas relativas a la transmisión de la señal y a las diferentes señales detectadas, tanto las recibidas en primera instancia como las enviadas por nosotros para verificar el correcto desempeño de las mismas.

En cuanto al repositorio de Github donde será publicado el código final del proyecto, Martín fue el principal encargado de las gestiones relativas al mismo, encargándose de que el código estuviese diferenciado claramente entre la parte que se estaba desarrollando y la parte que ya estaba correcta, así como la elaboración de los correspondientes "markdowns".

Con respecto a la memoria se encargó de la elaboración del capítulo correspondiente a la introducción además de su correspondiente traducción al inglés. Además, en el capítulo de arquitectura del sistema, se encargó de la elaboración de las secciones de generación de la señal y de transmisión de la señal. También se encargó dentro del capítulo de validación del sistema del correspondiente caso de uso del protocolo que había desarrollado, en este caso Toyota. Desarrolló también, en el capítulo de discusión, conclusiones y trabajo futuro, la última parte de trabajo futuro. Finalmente elaboró dentro del manual de usuario correspondiente al anexo I, la primera parte de instalación.

Bibliografía

[1] NHTSA, "Tires." https://www.nhtsa.gov/equipment/tires.

[2] R. A. C. de España, "El agarre del neumático al asfalto y la seguridad vial." https://www.race.es/agarre-neumatico-asfalto-seguridad.

[3] F. Wang, H. Chen, L. Guo, and Y. Hu, "Predictive safety control for road vehicles after a tire blowout," *Science China Information Sciences*, vol. 61, 07 2018.

[4] I. Rouf, R. D. Miller, H. A. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study.," in *USENIX Security Symposium*, vol. 10, 2010.

[5] A. Caballero Gámez, "Estudio de la seguridad del protocolo tpms." Trabajo de Fin de Grado en Ingeniería de Computadores, Facultad de Informática UCM, Departamento de Arquitectura de Computadores y Automática, Curso 2019/2020. https://eprints.ucm.es/id/eprint/62887/, 2020.

[6] P. Cancel, "Spoofing tire pressure monitor system." https://www.s4ur0n.com/assets/media/hc0n2k.pdf, 2020.

[7] BOE, "Boe-a-2020-8286, cuadro nacional de atribución de frecuencias," *BOE*, 21/07/2020.

[8] B. Larsson, "rtl_433." https://github.com/merbanan/rtl_433, 2013.

[9] triq org, "tx_tools." https://github.com/triq-org/tx_tools.

[10] librtlsdr, "rtl_sdr." https://github.com/librtlsdr/librtlsdr.

[11] rxseger, "rx_sdr." https://github.com/rxseger/rx_tools.

[12] "Hackrf one great scotss gadgets." https://greatscottgadgets.com/hackrf/one/.

[13] T. Instruments, "Data-sheet cc1101, low-power sub-1 ghz rf transceiver," 2012.

[14] F. P. Hammerle, "python-cc1101." https://github.com/fphammerle/python-cc1101, 2020.

[15] S. Teddy, "Cc1101." https://github.com/SpaceTeddy/CC1101.

[16] abhra0897, "Cc1101 msp430 energia library v2." https://github.com/abhra0897/msp430_cc1101_energia_v2#cc1101-msp430-energia-library-v2.

Anexo I: Manual de usuario

En este primer anexo se va a tratar de explicar de una manera breve y concisa como un usuario final debería utilizar la aplicación desarrollada durante el proyecto.

Esta explicación comienza con una primera parte donde se detallan las distintas librerías y programas que se deben instalar la primera vez que se trate de usa la aplicación para que esta pueda ejecutarse de manera correcta. Es importante remarcar que esta aplicación está diseñada para ser usada en un entorno con sistema operativo GNU/Linux por lo que no podrá ser utilizada en entornos con otro sistema operativo. En la segunda parte se detalla cómo cualquier usuario final debería usar la aplicación y los distintos modos que se ofrecen dentro de la misma, los cuales son, un modo de ataque único, conocido como "Attack One", el cual le permitiría al usuario realizar un ataque a un vehículo concreto, y un segundo modo de ataque masivo, conocido como "Attack All", el cual le permitiría al usuario realizar un ataque masivo a todos los vehículos que se vayan detectando.

Instalación

Lo primero que se debería tener instalado es el compilador GCC(GNU Compiler Collection) que nos permita compilar el programa. En caso de no tenerlo instalado, podremos instalarlo accediendo a la terminal de nuestro equipo y ejecutando el siguiente comando:

$ sudo apt-get install gcc

Una vez tengamos el compilador instalado se procederá a instalar el software necesario para utilizar el dispositivo RTL-SDR mediante el cual trataremos de recibir las señales TPMS. En nuestro caso hacemos uso de RTL-SDR SMArt v4 que tiene su propio manual de instalación [4] que recomendamos seguir paso a paso.

A continuación, se procederá a instalar el software RTL 433, que será el utilizado para decodificar las señales recibidas, podremos instalarlo accediendo a la terminal de nuestro equipo y ejecutando el siguiente comando:

$ sudo apt-get install rtl_433

Si se desea consultar más información acerca de este proyecto pueden consultar su repositorio [5].

[4] https://www.nooelec.com/store/downloads/dl/file/id/72/product/294/nesdr_installation_manual_for_ubuntu.pdf

[5] https://github.com/merbanan/rtl_433

Finalmente es necesario instalar la librería python-cc1101 que será utilizada para implementar la conexión con el chip y realizar la transmisión de la señal, de igual manera que antes podremos instalarlo accediendo a la terminal de nuestro equipo y ejecutando el siguiente comando:

$ pip install cc1101-python

Si se desea una mayor información es posible visitar el repositorio o la página del proyecto en PyPi[6].

Una vez tenemos todo esto instalado procederemos a compilar el proyecto haciendo uso del MAKEFILE, ejecutando el siguiente comando:

$ make all

Si en algún momento decidimos eliminarlo solo necesitamos ejecutar el comando:

$ make clean

Uso general

Una vez se ha iniciado la aplicación, el usuario se encontrará con una pantalla tal y como la que se puede observar en la figura 8.2:

Figura 8.2: Vista inicial

Tal y como vemos se trata de una interfaz muy simple donde se pueden apreciar únicamente tres botones, dos botones de ataque, uno para cada modo y un botón para salir de la aplicación.

Se va a comenzar el primer modo de ataque, al cual podemos acceder mediante el botón situado mas a la izquierda y que seria el ataque único. En este caso accederemos

[6]https://pypi.org/project/cc1101-python/

mediante el botón “Attack One”, el cual nos conducirá a otra ventana como la mostrada en la figura 8.3:

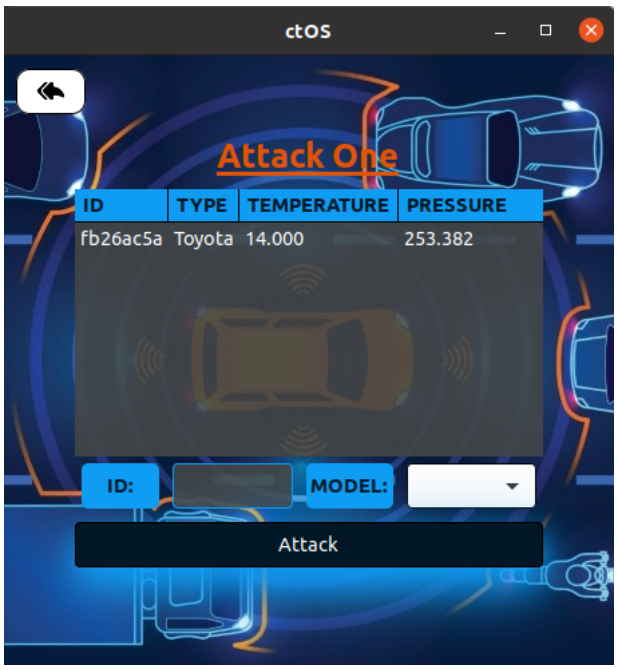

Figura 8.3: Vista Attack One

Tal y como se puede observar en la figura, a medida que vayamos detectando vehículos se irán añadiendo a la lista mostrada en pantalla, permitiendo al usuario seleccionarlo de la lista y realizar un ataque, de una manera rápida y sencilla, simplemente haciendo clic en el botón “Attack”.

Esta sería la manera más simple de realizar un ataque único, sin embargo, dentro de este modo de ataque único existe otra posible forma de realizar dicho ataque. Esta segunda forma consistiría en que, siempre y cuando se conozca el ID del sensor que queremos atacar y el modelo del mismo, podemos introducir estos valores manualmente utilizando los campos habilitados para tal efecto como son el campo de texto “ID” y el desplegable “Model” y pulsar en el botón “Attack” para realizar el ataque a ese vehículo del cual ya conocíamos alguno de los IDs de sus neumáticos.

Para la realización del segundo tipo de ataque, el ataque masivo, se accederá a través del botón situado mas a la derecha en la interfaz principal, el botón de “Attack All”, mediante este botón accederemos a una pantalla como la mostrada en la siguiente imagen:

Figura 8.4: Vista Attack All

En primera instancia cuando accedamos a este modo de ataque nos aparecerá la lista vacía, será entonces cuando una vez iniciado el ataque pulsando en el botón “Attack” comenzarán a aparecer todos los datos de los sensores que están siendo atacados, tal y como se puede ver en la imagen anterior. Se realizará por tanto un ataque masivo a todos los sensores que se vayan detectando hasta el momento en que decidamos parar nuestro ataque.

En resumen, estos serían los dos modos disponibles en la aplicación, un primer modo, “Attack One”, que consiste en un ataque único y que se puede realizar de las dos formas indicadas anteriormente, bien seleccionando el sensor de la lista de sensores detectados o bien introduciendo manualmente el ID y el modelo, y un segundo modo, “Attack All”, que consiste en la realización de un ataque masivo a todos los sensores que se van detectando.

Anexo II: Repositorio del proyecto

Enlace al respositorio del proyecto:

https://github.com/Martingf56/ctOS-TPMS

Sobre TEFLONX

Teflon X(cc0 1.0(documentación) MIT(código))es una plantilla de LaTeX creada por David Pacios Izquierdo con fecha de Enero de 2018. Con atribuciones de uso CC0.

Esta plantilla fue desarrollada para facilitar la creación de documentación profesional para Trabajos de Fin de Grado, Trabajos de Fin de Máster o Doctorados. La versión usada es la X

V:X Overleaf V2 with XeLaTeX, margin 1in, bib

Contacto
Autor: David Pacios Izquiero
Correo: dpacios@ucm.es
ASCII: ascii@ucm.es
Despacho 110 - Facultad de Informática

Jorge María Martín, Alberto Rodríguez Fuentes y Martín García Fuentes
01 de junio del 2021
Ult. actualización 15 de junio de 2021
LaTeX lic. LPPL & powered by TEFLON CC-ZERO