Title: Eavesdropping an EFC transaction using SDR

Student: Javier Felipe Blanco

Problem description:

In Norway, as well as in many other European countries, Electronic Fee Collection (EFC) is used to charge drivers for using toll roads. This technology is based on the communication between a so-called On-Board Unit (OBU), installed in the vehicles, and a Road Side Equipment (RSE) device placed in the toll road, by means of a wireless link. Proper mechanisms against the increasing security threats are fundamental nowadays for every communication system. This is particularly relevant when the system performs sensitive transactions with user data, such as EFC. Many vulnerabilities have been detected in this system in the last years. Weak authentication and access control mechanisms, based in DES algorithm, and no encryption in the wireless link, make possible performing an attack against the system.

This project will take a step further in the analysis of the EFC system security, recreating a real transaction by using commercial RSE and OBUs in a lab environment. This communication will be later eavesdropped using a Software Defined Radio (SDR) device, and the received signal will be processed, aiming to read and store the exchanged EFC data.

Responsible professor: Stig Frode Mjølsnes, ITEM

Supervisor: Tord Ingolf Reistad, Statens Vegvesen

Abstract

The use of Electronic Fee Collection (EFC) system is widespread in Europe, to charge drivers for the use of toll roads. It is based on a wireless high-frequency communication between an On Board Unit (OBU) in the cars and a Road Side Equipment (RSE). The data is exchanged using the Dedicated Short Range Protocol (DSRC), with a working frequency of 5.8 GHz.

Several projects and research have shown that EFC lacks good security features. A weak encryption mechanism (Data Encryption Algorithm) is used to provide authentication features, and the link is not encrypted. Therefore, the system is susceptible to suffer an attack, which could bring undesirable consequences for both the road users and the service providers.

In this thesis, the feasibility of building a device capable of eavesdropping the communication in EFC has been explored. For this, Universal Software Radio Peripherals (USRP) and the GNU Radio framework have been used. These tools are part of the Software Defined Radio (SDR) scope, and they are publicly available and cheap. To establish a DSRC link, commercial RSE and OBU devices have been utilized.

Although an error-free signal receiver was not achieved, the developed program is able to obtain a high amount of information from the system. It makes possible to visualize the downlink and uplink signals, both in time and frequency domain. This shows that, without an encryption mechanism, it can be possible to eavesdrop the communication and potentially read private data from the users.

Additionally, some other tests have been performed with the described tools. One of them shows that, the signal emission of other wireless systems in the 5.8 GHz band, could cause an interference and interrupt the EFC communication. The risk of suffering a replay attack is also explored.

Preface

This Master's Thesis has been carried out at the Department of Information Security and Communication Technology (IIK), at Norwegian University of Science and Technology (NTNU), as part of my Erasmus+ exchange program.

I would like to thank both my professor, Stig F. Mjølsnes, and my supervisor, Tord I. Reistad, for the feedback and guidance throughout the year.

Trondheim, June 2018

Javier Felipe Blanco

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Intelligent Transportation Systems and EFC

EFC is one of the applications within the scope of Intelligent Transportation Systems (ITS). These involve both Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communications, to provide services such as traffic management, traffic information broadcast, or even collision avoidance. While most of these potential applications have not been widely implemented yet, EFC has a significant presence in most of the developed countries in the world.

EFC is based on a wireless communication between two devices, a Road Side Equipment (RSE) and an On Board Unit (OBU). The OBU shall identify the road user when reaching a toll point, and the RSE shall be installed in these areas and read the information contained in the OBUs. Besides, the RSE will be connected to a centralized system, containing all the necessary information to complete the payment.

In most EFC systems in Europe, DSRC is the protocol used for the wireless data exchange between the OBU and the RSE [Com15]. The Norwegian implementation of EFC is called AutoPASS, and it is based on the European standards for DSRC and EFC [Adm14].

1.2 Motivation

It is very important to take security issues into consideration in EFC. The data exchange between the RSE and the OBU contains private information about the users, and should not be accessed by non-authorized parties. Furthermore, it should be granted that nobody can build fake devices for illegitimate purposes, such as using the toll roads and then make another user pay for that.

EFC implements both authentication and access control mechanisms [CEN11].

More details will be given in the next chapter. The Data Encryption Standard (DES) cipher is used in both mechanisms. Nowadays, DES is not considered as a secure encryption method, mainly because of the short key, which is only 56-bit long. With the computation capabilities available nowadays, this makes DES vulnerable against brute-force key search attacks [des].

In addition to the weaknesses of DES, there is another point to remark. The packets in the DSRC communication between the OBU and the RSE are not encrypted, leaving open the possibility of sniffing the DSRC link and obtaining private user information. Even more, it could also be possible to perform a replay attack against the system, using the captured information and building a DSRC transmitter.

All the characteristics described above justify the need of making a further study of the EFC system. Some previous work has been conducted on this area by other Norwegian University of Science and Technology (NTNU) students, and it will be described in the following chapters.

1.3 Objectives

This thesis will continue the analysis of the toll road system security, focusing on building a DSRC radio device which is able to receive and decode the signals involved in an EFC transaction. The no-encryption of the link should allow an attacker to eavesdrop the DSRC link and read the protocol messages and the user data, by building a radio receiver. The main objective of the project is to prove whether it is feasible to build such a device, using publicly available tools and free signal processing software.

Additionally, the possibility of replaying the received radio signals against one of the legitimate devices will be explored, and other tests might be performed. The EFC system itself will be studied.

1.4 Methodology

First of all, a DSRC link containing EFC data will be established. This will be done using a commercial RSE device, and several OBU devices used in the AutoPASS system. The RSE needs to be configured by running a controlling software, that must also implement all the necessary logic in order to complete a transaction with an OBU. This program will be developed using a driver provided by Kapsch, the manufacturer of the RSE.

After this, the DSRC receiver will be developed using Software Defined Radio (SDR) tools, with the goal of eavesdropping that communication. For this, a Universal

Software Radio Peripheral (USRP) will be used, with antennas designed for DSRC 5.8 GHz applications. In addition, the free-licensed framework GNU Radio will be used to develop a Digital Signal Processing (DSP) software. This framework is based on the interconnection of signal processing blocks to build SDR applications. All the blocks are created by contributors around the world, and some of them are included in the GNU Radio installation package. However, it's possible to install new blocks, and also to write new ones.

Apart from building a receiver, the possibility of recording and replaying the DSRC signals, and some other additional test, will be performed during this thesis, with the goal of contributing to the security analysis of the EFC system.

1.5 Content

- **Chapter 1** gives an overview on the project, and summarizes the motivation and goals.
- **Chapter 2** explains the most important features of EFC, focusing on the security of the system.
- **Chapter 3** goes through the physical and link layers of DSRC, describing the most relevant information needed to build the SDR receiver.
- **Chapter 4** describes the configuration process of the RSE unit used in this project, and the communication with an OBU.
- **Chapter 5** is based on the whole process of developing the receiver, including the fundamental signal processing theory behind it, and the obtained results.
- **Chapter 6** describes additional tests performed with the SDR tools, regarding to signal replay and interference.
- **Chapter 7** gives some research conclusions, and proposes some guidelines for further work.

Chapter 2
Electronic Fee Collection and security issues

In this section, an overview about the EFC system using DSRC will be given. Particularly, some possible attacks and threats against the system will be presented and discussed.

2.1 Electronic Fee Collection with DSRC

2.1.1 Background on EFC

In several countries around the world, electronic toll collection systems are used to charge the users of the toll roads. In many of these countries, manual payment systems are also implemented. This electronic collection systems are not always interoperable, since each country developed them independently. Therefore, many times, a single user needs several different OBUs to use the electronic toll collection system in different countries.

Back in 2004, the European Commission published the Directive 2004/52/EC, which some years later would result in the Decision 2009/750/EC. These publications define an European Electronic Toll System (EETS), which has not been implemented yet, but gives some valuable guidelines for the countries in the EU to follow for their fee collection systems. The wireless technologies within the scope of these rules are Satellite Positioning (GNSS), GSM and DSRC [Com15].

Even if there is not complete interoperability in Europe right now, almost all the currently implemented systems are compliant with the Decision published by the European Commission. Most of them use DSRC at 5.8 GHz. In addition, there are some providers that have developed joint solutions, thus enabling the customers to access the tolling networks of the countries involved using a single OBU. An example of this is the EasyGo system between the Scandinavian countries.

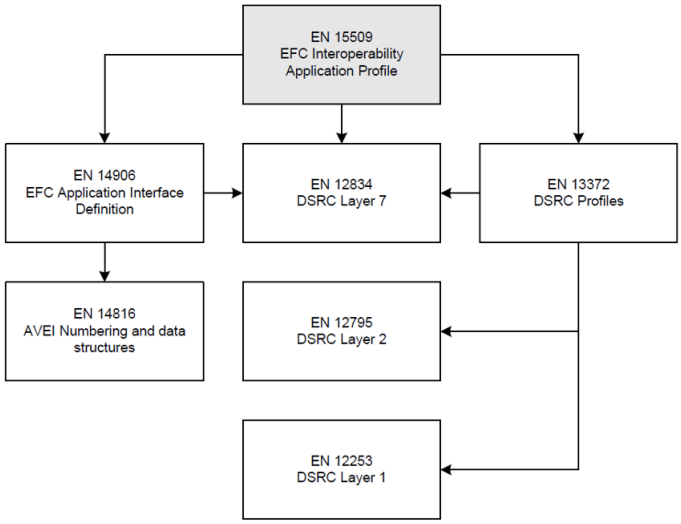

Figure 2.1: Relations between DSRC standards. Source: [CEN07]

2.1.2 EFC-DSRC standards

The European Committee of Standardization (CEN) has approved and published several standards regarding EFC with DSRC. These standards are the base for many of the toll collection systems in Europe, AutoPASS system among them.

There are separate standards for each protocol layer of the EFC-DSRC stack, and they are related as shown in the Picture 2.1. The documents include the specifications of the DSRC physical layer (EN 12253), link layer (EN 12795) and application layer (EN 12834).

EN 13372 defines sets of values for DSRC parameters in the three layers, called DSRC profiles. EN 14906 provides specifications for EFC attributes, application functions and a EFC transaction model. Finally, EN 15509 specifies the implementation of EFC using the previously mentioned standards, aiming to be a reference for interoperable EFC systems based in DSRC.

2.1.3 The EFC-DSRC protocol stack

As mentioned in the previous section, the DSRC protocol in Europe is standardized by CEN, and three layers can be distinguished, the physical, link-layer and application layer. In top of the application layer, the actual EFC applications (like AutoPASS) will run.

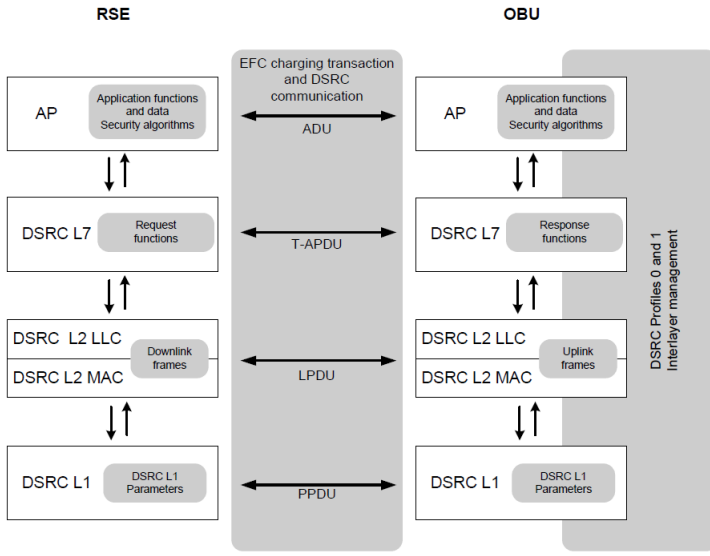

Figure 2.2: DSRC protocol stack. Source: [CEN07]

The DSRC application layer (or L7) defines some services, which shall be used by the application running on top of it. In this case, the EFC application builds some functions using the underlying L7 services. The most important services are:

- GET: to obtain data.
- SET: to modify data.
- ACTION: an action is performed. In EFC, this service is used by the GET-STAMPED function, which will be explained later.
- INITIALIZATION: this service is used when the RSE and the OBU start the communication.
- EVENT-REPORT: in EFC, this service is used to close the transaction.

On the other hand, all the data contained in the OBU units (vehicle plate number, information about the contract, ...) is specified as EFC attributes in the EN 15509 document. This data is accessed or modified by the RSE using the services mentioned above.

2.1.4 DSRC physical layer

In order to eavesdrop an EFC transaction, it's necessary to study in depth the physical layer of the DSRC stack. Therefore, the EN 12253 standard [CEN04a] has a very high relevance on this thesis.

This document describes all the aspects related to both the downlink and uplink signals in DSRC at 5.8 GHz. All this information is necessary to develop a proper receiver later on. A greater description of this physical layer is given in the next chapter.

2.1.5 Data exchange between the RSE and the OBU

The communication between the two entities which exchange EFC data, namely the EFC and the OBU, consists on an exchange of data frames using the 5.8 GHz link. The EFC, when active, sends periodically Beacon Service Table (BST) messages. This beacons contain information about the available applications in top of DSRC (in our case EFC, but it could be a different one, or even more than one). Communication parameters such as information about the sub-carrier frequency (more details on this are given in the next chapter) are also included. The BSTs also have a Beacon ID, in order to identify which RSE unit sent the beacon message.

In order to start an EFC transaction, an initialization process is performed. When the OBU receives one of the BSTs, it makes a check with the last received BST. If the Beacon ID of the newly received BST is different from the previous one, or the time between the reception of those two BSTs is higher than 255 seconds, the OBU goes ahead with the process. If neither of those conditions is fulfilled, the BST is discarded.

If the BST is accepted, the next step is checking whether there is a match between the applications included in the BST and the ones that supports the OBU. If so, a Vehicle Service Table (VST) is built and sent back to the RSE. This message includes information about the contract between the user and the toll road service provider, such as the contract provider and type of contract. The frame containing the VST is identified by a LID (Link Identifier) number, randomly selected by the OBU.

Upon reception of the VST, the LID contained in it is stored by the RSE, and associated to the correspondent OBU. After this point, the RSE can send GET and SET messages in order to complete the EFC transaction. After finishing it, the transaction is closed and the LID associated with the OBU deleted from the RSE.

2.2 Security features in EFC

In a system like EFC, where payment transactions are included, some security requirements should be granted. The DSRC-EFC standards define two security features: authentication and access control. The first one implies that the OBU should be recognized as valid in order to access the tolling services, and the second

one requires the road-side unit to authenticate before accessing or modifying the parameters in the OBU.

The authentication of the OBU is performed using a Message Authentication Code (MAC), whereas the RSE performs a calculation of a parameter called Access Credentials to gain access to the OBU. These two mechanisms are described in more detail below.

2.2.1 Security levels

The EN 15509 standard defines two different security-levels for EFC. The security level 0 requires only authentication of the OBU, while the security level 1 (the one implemented by AutoPASS) requires additionally access control to the data in the OBU. Therefore, the RSE also has to proof its legitimacy. When working with only the security level 0, a fake RSE could communicate with OBU devices and access its data.

2.2.2 Access Control

OBUs store an 8-octet Access Key (AcK), derived from a 16-octet Master AcK. Details about the key derivation in EFC can be found in EN 15509, Annex B. When receiving a valid BST, the OBU sends in the VST message both an Access Credential reference (AC-CR-KeyReference), which includes a reference to the key generation, and a 4-octet random number. The RSE then computes the Access Credentials value (AC-CR), using the random number sent by the OBU as input and the AcK as key. The DES algorithm is used to compute AC-CR. Afterwards, the AC-CR is sent to the OBU, which verifies if the value is correct. If yes, the RSE is authenticated. (See Figure 2.3).

Note that this procedure is only performed when the system implements Security Level 1. With Security Level 0, the RSE doesn't have to calculate any credentials, and therefore doesn't need to authenticate before using a GET or SET service.

2.2.3 Authentication

As stated above, the authentication of the OBU is based on the calculation of a MAC. When the RSE request attributes from the OBU, a GET-STAMPED message (which uses the ACTION DSRC-L7 service) is sent. This message contains the identifier for the requested attributes (AttributeIDList), and a random number (RandomRSE). The random number and the values of the requested attributes (AttributeList) are concatenated and used as the input for the MAC calculation, by means of the Cipher Block Chaining Message Authentication Code (CBC-MAC) method. An Authentication Key (AuK) and DES encryption are used. The AuK (8 octets) is

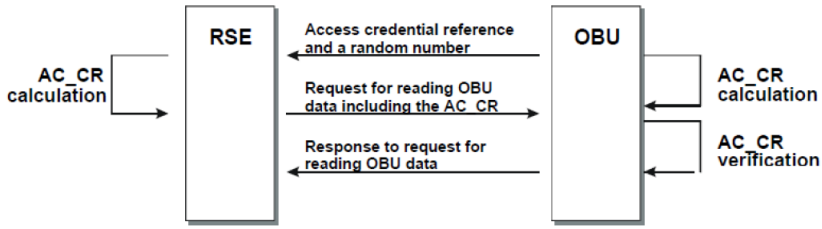

Figure 2.3: Access control in EFC. Source: [CEN07]

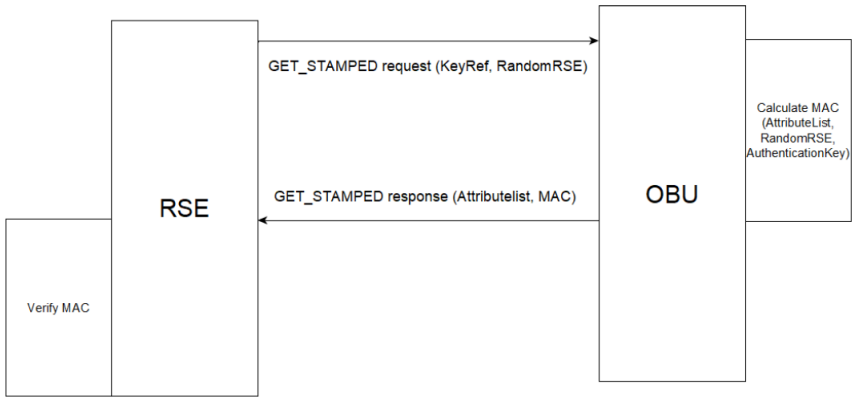

Figure 2.4: Authentication of the OBU in EFC.

derived from a Master AuK (16 octets), using values known both for the RSE and the OBU (Payment Means and Contract Provider attributes). The OBU stores 8 different AuK from 8 different key generations, so the RSE specifies which AuK to use in the GET-STAMPED message (KeyRef value). When the OBU answers the RSE, the MAC is included in the message, and then verified by the RSE to authenticate the OBU.

2.3 Security threats

EFC implements an authentication mechanism for both the RSE and the OBU. However, this doesn't imply that the EFC systems are secure. Indeed, there are many arguments that support the lack of security in EFC.

First, both the authentication and access control mechanisms are based in the

DES standard. Therefore, both for the calculation of the MAC codes by the OBU and the calculation of the access credentials value by the RSE, 56-bit long temporal keys are used. This key length is very short to be considered secure nowadays, as it would be feasible to recover them by performing a brute-force attack.

Another important point to consider is that the DSRC link is not encrypted. Therefore, it could be possible to eavesdrop the communication by using a customized DSRC receiver and get private information. In addition, it also makes easier to perform a replay attack against the system, capturing the data transmitted from one OBU to the RSE, and later re-transmit that information to the RSE.

2.3.1 EFC security assessment

Back in 2015, CEN published a technical report [CEN15] analyzing the security threats for EFC. Many topics were discussed in this document, and some concrete attacks and possible solutions were presented.

The report focused mainly in the weakness of the authentications and access control mechanisms, due to the short keys. Some of the most relevant attack scenarios are discussed below.

Brute force attack against the DES keys

Both the Access Control and Authentication keys are subject to brute force attack. When an OBU receives a BST, it includes in the VST message two values (Access Credentials reference and Random Number) used to compute the AC-CR value by the RSE. If a customized OBU unit is built, a chosen-plaintext attack could be performed, obtaining AC-CR values from an RSE, which could be used to retrieve the Access Credentials key.

In a similar way, if a customized RSE is built, a chosen-plaintext attack could be performed against an OBU, and obtaining MAC values retrieve the AuK. In this case, the GET-STAMPED message (which includes part of the input for the calculation of the MAC value by the OBU) must also include the Access Credentials value calculated by the RSE, so that the OBU verifies the authenticity of the RSE. Therefore, it is necessary that either the RSE knows the value of the AuK of the OBU (which could be obtained from the previously mentioned attack), or that the Security Level 0 is used, as this scenario doesn't require the calculation of the access control credentials.

An attacker with in possession of these keys could clone and build fake OBU units, copying or faking the attribute set of valid OBUs.

Integrity attack against the VST data

The VST message is not protected by a MAC code, so an attacker could modify the parameters sent in this message. Some of these parameters are used as the input for the calculation of the Access Credentials by the RSE, so if an attacker manages to change these values, the access control mechanism would fail.

Replay attack against the RSE

Although this attack is not specifically mentioned in [CEN15], a DSRC-capable transceiver could intercept the messages involved in the communication between a RSE and a certain OBU, and later replay the uplink messages against the RSE. This could involve a security breach in the system.

Eavesdropping the communication

As mentioned before, the DSRC link is not encrypted at any point. This means that, building a DSRC receiver, an attacker could read the data exchanged in the EFC transactions. Apart from reading user-related data, some other parameters (as the ones used as input to generate the Access Control and MAC codes) could be read. This could also be an advantage for an attacker aiming to retrieve the system keys.

Obtaining Master keys

After obtaining authentication and access control keys, an attacker could try to obtain the master keys from which the previously stolen keys have been derived. However, this is considered as unfeasible nowadays, as the key derivation algorithm (3DES) is considered to be secure.

2.4 Related work

In the previous years, some other NTNU master students studied the security features and weaknesses in the EFC system. In 2016, Jonathan Hansen studied in his Master's Thesis [Han16] the possibility of building an DSRC transceiver, and also presented some concrete brute-force attacks which could be performed to break the authentication and access control mechanisms.

Another Master's Degree student, Sverre Turter Sandvold, focused one year later [San17] on obtaining MAC codes from OBUs, which could be later used to retrieve the authentication key used by the OBU to generate those MACs. Using a professional DSRC transceiver, and customizing the GET-STAMPED message, he managed to obtain MAC codes from OBUs with known keys (to bypass the access control mechanism). He also presented a method based on rainbow tables to brute-force the authentication key.

Chapter 3
The DSRC Physical and Link Layer

As stated before, the physical layer of the DSRC stack is the most relevant one for the work performed during this thesis, as the knowledge of the physical parameters of the signal (modulation, bit rate and so on) is necessary in order to build a transceiver. Therefore, an in-depth study of this layer has been performed, using the [CEN04a] standard as a guideline. In this chapter, the most relevant aspects of the link layer and the frame structure will be also presented.

3.1 DSRC at 5.8 GHz

In Europe, the use of DSRC in the 5.8 GHz ISM band was standardized by CEN. As this is a non-licensed band, other applications (not necessarily involving a radiocommunication) could emit power in the same band, and therefore cause interference. But since all the applications that make use of this band (including EFC) involve short-range signals, this is more difficult to happen.

The CEN standard defines 4 possible channels within the 5.8 GHz band, where the downlink carrier is allocated: 5.7975 GHz, 5.8025 GHz, 5.8075 GHz and 5.8125 GHz. AutoPASS uses the 5.7975 channel. A signal with enough power emitted in this frequency should wake up the OBUs within the coverage range (the OBU devices are not connected to any power source).

In the following sections, both the downlink and the uplink will be described in more detail. As one of the goals in this thesis is to build a device capable to read the information involved in the communication between the RSE and the OBU, both links must be studied.

3.2 Downlink

The RSE units in AutoPASS transmit information using a carrier at 5.7975 GHz, as indicated above. The signal is polarized hand-left (this is relevant in order to choose

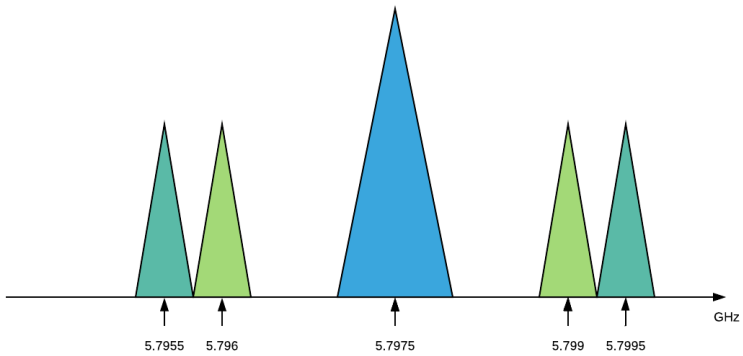

Figure 3.1: The uplink and downlink channels in DSRC, with downlink carrier at 5.7975 GHz.

an optimal antenna).

Baseband signal and modulation

The modulating signal, i. e. the pulse based baseband signal carrying the binary information, has a 500 kbit/s bit rate. The used data coding is FM0. In this scheme, the "1" bits are coded with a transition in the beginning and in the end of the bit interval, while the "0" bits have an additional transition in the middle of the bit interval. The modulation used in the downlink is a two level amplitude modulation. In this type of modulation, the sinusoidal carrier signal changes its amplitude between a high and a low signal level, according to the shape of the baseband signal.

It was a complex task to find information about the baseband bandwidth of the signal, even in the standards. However, after performing some signal spectrum recordings (described in the Chapter 4, it's estimated that the downlink baseband bandwidth is 500 KHz for the downlink, and 250 KHz for the uplink.

When the information is transmitted via the modulated radiofrequency signal, the amount of spectrum occupied is the double of the baseband signal bandwidth. So, the radiofrequency signal carrying the information requires 1 MHz bandwidth for the downlink, and 500 KHz for the downlink, which can be slightly higher in the real transmission.

Preamble

Another important parameter to take into consideration is the preamble of the frame. As the transmission is not continuous, but burst-based (there are intervals where

Table 3.1: Comparison between downlink and uplink physical layer in DSRC

Parameter	Downlink	Uplink
Carrier modulation	2-level Amplitude Modulation	BPSK
Carrier central freq.	5.7975 GHz	1.5 MHz or 2 MHz separation from downlink carrier
Data coding	FM0	NRZI
Bit rate	500 Kbit/s	250 Kbit/s
Preamble	16 high-low level pulse alternating sequence (32 µs)	32 to 36 µs modulated with subcarrier, and 8 NRZI-coded bits

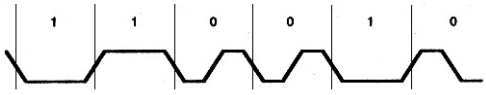

Figure 3.2: The FM0 baseband signal

no information is sent), the receiver needs to know somehow when the data frames start. In the downlink signal, the preamble consists on an alternating sequence of high and low level pulses (2 µs of duration each), and has a length of 16 bits (with a tolerance of 1 bit).

3.3 Uplink

The OBUs make use of a subcarrier (another signal with a certain separation in relation to the main carrier frequency) to send the uplink data to the RSE. In this case, the standard specifies two possibilities: a subcarrier at 1.5 MHz or at 2 MHz from the carrier frequency. The standard [CEN04b] recommends the use of the 1.5 MHz subcarrier whenever possible, for interoperability with legacy installations. The OBU units, however, must support both subcarriers, and it's the RSE in the BST messages which specifies the subcarrier frequency to use.

The modulation used in the DSRC uplink is Binary Phase Shift Keying (BPSK). With Phase Shift Keying (PSK) modulation, the carrier is modulating by changing its phase depending on the symbol value. BPSK is the simplest PSK modulation, as the carrier signal phase changes between only two values: 0 and 180 degrees. Therefore, BPSK also carries one bit per symbol.

There are also other significant changes in the baseband signal: the employed data coding is Non Return to Zero Inverted (NRZI), instead of FM0. The reason of

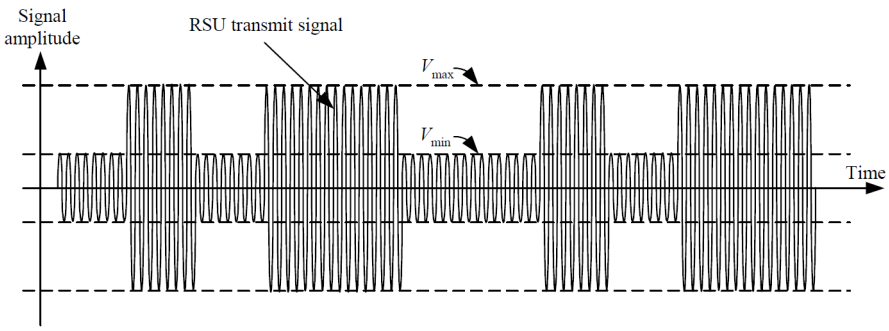

Figure 3.3: The modulated downlink signal. The bit sequence here is: 01101. Source: [ETS04]

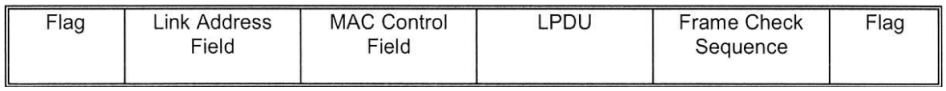

Figure 3.4: Structure of the DSRC link layer frame. Source: [CEN03a]

employing the data coding, both in the uplink and in the downlink, is to make the bit information independent on the value on the symbol itself, but on the symbol transition. This way, if due to the transmission the symbol values are inverted (low level transmitted symbols are received as high level, and vice versa), the information will still be correct.

The binary rate is also lower, 250 Kbit/s. The uplink signal also includes a preamble.

3.4 DSRC link layer

When either the OBU or the RSE transmit data on the wireless link, they send a frame with a specified structure (shown in Figure 3.3). This frame contains a Link Identifier (LID), which can be either private (to address the communication between the RSE and a particular OBU device) or a broadcast LID. It can contain (or not) data corresponding to the application layer, which is included in the Link layer Protocol Data Unit (LPDU) field. A Frame Check Sequence field is added after the LPDU. Also, a flag is included at the beginning and at the end of the frame. This frame is sent after the physical level preamble, described in the previous section.

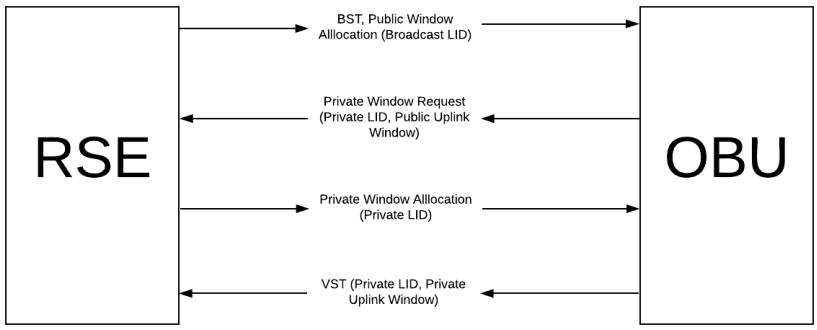

Figure 3.5: BST-VST exchange with private window allocation.

The MAC sublayer[1] within the DSRC link layer is responsible for the access control to the wireless interface, in order collisions not to happen. The resources are allocated following the time division multiple access (TDMA) technique, which means that different time slots or windows are allocated for different frames, both in the downlink and the uplink. The RSE has the function to allocate uplink windows to the OBU units that want to send a frame. These windows can be either public or private. When the downlink frame includes a private LID, previously assigned for the communication between the RSE and a certain OBU, a private uplink window is allocated for the uplink frame answering the downlink frame. On the other hand, when a frame with a broadcast LID is sendt by the RSE, several public uplink windows are allocated one after each other. These may be used by any OBU device in the coverage area.

During the transmission of BSTs by the RSE, the broadcast LID is used, as the beacons are intended to be received by any valid OBU within the coverage area. When an OBU receives one of the beacons, it has to answer with a VST message. However, at this point no private LID has been assigned yet. This means the OBU device can only use an uplink public window to communicate with the RSE. These windows are smaller than the private ones, and are not large enough to support the VST sending. Therefore, the OBU has to request a private window allocation before sending the VST. Upon reception of the BST, the OBU application layer generates a private LID, which is sent in the private window allocation request. This request is processed by the RSE, and afterwards a private window allocation frame is sent back to the OBU, that now has a private window available to transmit the VST and the subsequent frames involved in the transaction.

[1]Here, MAC makes reference to the protocol layer with that name, not to Message Authentication Code

Chapter 4
Setting up the DSRC communication

In this chapter, the steps followed to perform a successful DSRC communication between the OBU and the RSE will be described. The data exchange involved in this communication will be later used as input for the DSRC receiver, built with an USRP and GNU Radio.

4.1 The OBU Programming Station

To fulfill an EFC transaction with an OBU, it is necessary to have an active RSE device, which sends beacon messages periodically and handles the responses from the OBU. For this purpose, the OPS-1955 (OBU Programming Station, manufactured by Kapsch) will be used.

This device is used by toll road service providers (or similar entities) to program the OBU, before delivering it to the user of the road. This step, also known as OBU personalization, includes the configuration of all the customer specific data in the OBU, so that the user can access the services provided by the toll road, generally both access to the road and payment. As the OPS-1955 implements the EFC and DSRC protocol stack, it can be used in a lab environment as a RSE, in order to recreate the communication that takes place between a RSE and an OBU in a real toll road.

To carry out the communication between the RSE and the OBU, it is necessary to develop and run a software that configures the RSE with all the parameters needed to initialize the radio link. This software is also responsible for handling the bidirectional RSE-OBU communication, and will be run in another computer connected to the OPS-1955 (host computer). The connection will be done with an USB cable, that will also be used to power the OPS-1955, which is an USB-powered device.

4.2 Development of the RSE controlling software

Kapsch provides an API called Layer7, which acts as a interface between the OBS-1955, and the software that controls this device (host application). When the RSE receives a frame from the OBU in the DSRC link, it will be notified to the host application via the Layer7 Protocol, which implements some predefined messages for the communication between the OPS-1955 and the host application. The Layer7 library implements also a set of functions, which make it easier to develop the host application. The driver will encapsulate the messages sent between the OPS-1955 and the host application, and will handle the communication between then. The application is only responsible for handling the data contained in the Layer7 messages.

First, the software package containing the Layer7 driver is installed, following the steps indicated in [AG14a]. After completing the installation, the driver will run as a service and the host computer can communicate with the OPS-1955. Upon successful communication, the LED in the OPS-1955 will turn on with blue colour.

Now that the driver is installed and the host computer and the RSE can communicate, it's possible to develop and run a host application, using the provided Layer7 API functions. This software is written in C, as that's the language the driver supports for function calls from an application. Kapsch provides, together with the driver installation and the libraries, a small demo RSE controlling software. This software implements many of the functions that are necessary to configure the RSE and handle the messages from the OBU, and also many structures related to the data attributes defined in the standards. Thus, this code is very useful for the development of the controlling software.

In order to configure the RSE, the main task is to write a main function that uses the code mentioned above, as well as modifying some of the functions that are called. The developed host application implements the BST sending functionality, after configuring the DSRC link. In order to build a full EFC transaction, more functionality has to be implemented in the controlling software. In this project, the RSE will just send beacon frames, but will not answer to the received VST in order to request more OBU attributes. However, the BST-VST exchange is sufficient to attempt the link eavesdropping and test the receiver, which will be described in the following chapter.

Before starting coding the application, the programmer's manual provided by Kapsch [AG14b] has been read, with the aim of gaining better understanding of the Layer7 API. This manual provides a description of the Layer7 communication and all the possible messages and parameters, and gives a guideline about how the OPS-1955 should be initialized. This information has been also very helpful for the coding process.

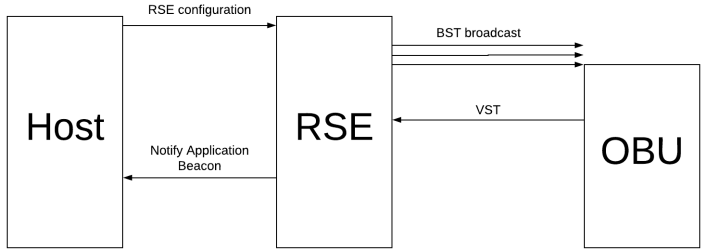

Figure 4.1: OPS-1955 configuration and communication with an OBU.

The executable file used to control the OPS-1955 is added as an attachment. Note that this file requires to be run in the same directory as the Layer7 Driver header file, and this is proprietary software from Kapsch.

4.3 Communication between the OPS-1955 and an OBU

After running the application software, the RSE starts sending beacon frames periodically, with a frequency configured by the previously mentioned software. When an OBU is placed within the coverage range of the RSE, a VST is sent in the uplink, and it's received by the RSE, that notifies this event to the controlling software, via a Notify Application Beacon message. This process is described in the Figure 4.1.

This Notify Application Message, send from the RSE to the host running the controlling software, includes the parameters sent with the VST. This way, the software can process that information and perform the subsequent phases of the transaction.

A printing function was defined in the software, so that a message would pop un in the console upon BST reception. In addition, one of the OBUs available has two leds: one yellow and one red. When the yellow one is light on, indicates that the OBU has waken up to a signal in the desired frequency range. When the red light is also on, the OBU has also identified a correct BST. Here, both led lights go on for a short amount of time, when the OBU detects a BST.

Visualizing the signal spectrum

After doing the setup on the RSE and communicating it with an OBU, several recordings with the tool GQRX were performed. This tool is built on GNU Radio, and provides a GUI where it's possible to record the signal and see its frequency spectrum and how it changes with the time. This was very useful in order to gain better understanding of the characteristics of both the uplink and downlink signals.

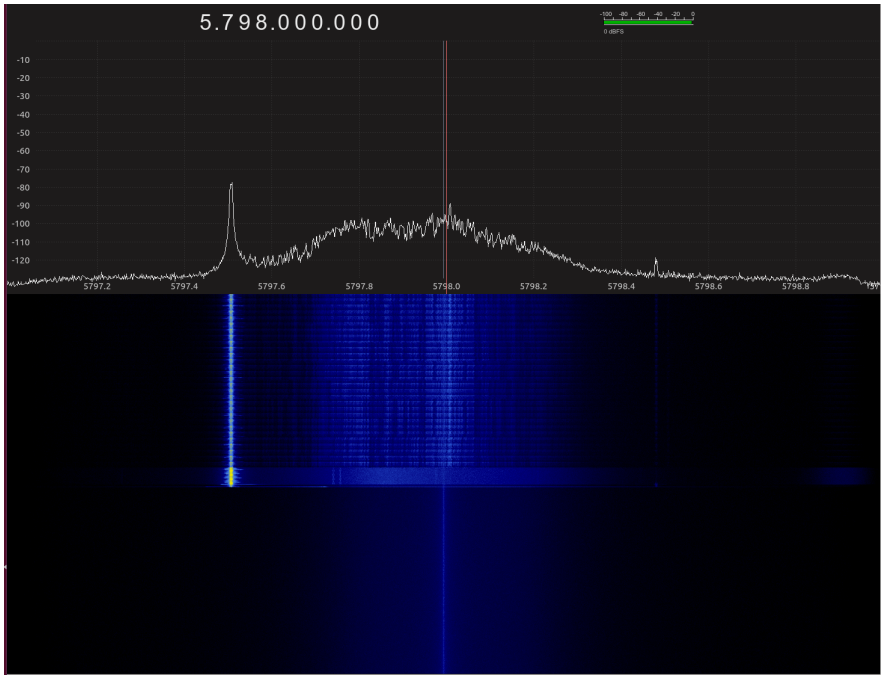

Figure 4.2: Frequency spectrum centered at 5.798 GHz.

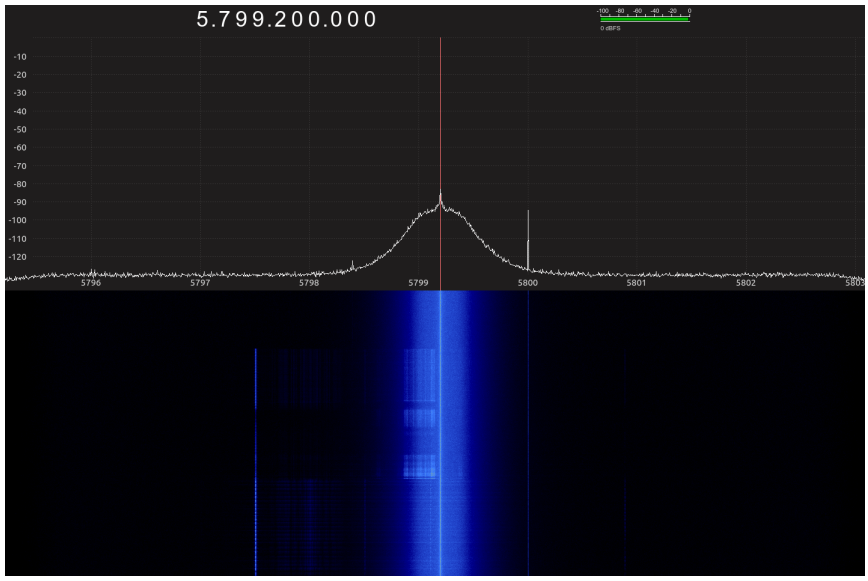

Figure 4.3: Frequency spectrum centered at 5.7992 GHz.

In the Figure 4.2, where the receiver is tuned close to the downlink carrier signal, the BST broadcast can be seen in the frequency spectrum. The higher power component, centered at downlink carrier frequency (5.7975 GHz), can be seen, together with the peaks corresponding to the BSTs and expanded along all the bandwidth of the downlink. Although it can't be seen in the picture, because the receiver is tuned at 5.798 GHz, the signal also has power in the frequency spectrum below the carrier frequency.

In the Figure 4.3, the uplink VST messages can be seen, centered at 5.799 GHz. As the RSE only sends BSTs and the controlling application doesn't perform any further function upon reception of the VST, the OBU continues sending VSTs during certain time. This is, most likely, because the OBU interprets that the VST is lost in the wireless link, as no response is received.

Chapter 5
Developing a DSRC burst receiver

This chapter will focus in the work performed to eavesdrop the DSRC link. The tools used will be described, as well as the approach followed. Finally, the obtained results will be presented and discussed.

To perform this, first a literature review has been carried out. This has included a study of the DSRC standards (specially the one related to the physical layer), a review of previous works in the area of signal decoding with USRPs and GNU Radio, and also a study on several digital signal processing concepts. The tutorials provided by GNU Radio, specially , have been very useful.

5.1 Universal Software Radio Peripherals (USRP)

For this project, a USRP (model B200 Mini) has been used. This radio devices (manufactured by Ettus Research) permit tuning the radio signal by choosing the center frequency, and deliver the I/Q baseband data after performing the necessary processing stages. This device can be connected to a computer by USB, for further processing of the data using a software. The frequency range of the device goes from 70 MHz to 6 GHz, which makes it suitable for DSRC applications at 5.8 GHz [Resa].

The basic architecture of the USRP is shown in Figure 5.1. The RF signal is received using an antenna connected to one of the ports. As there are two ports, full-duplex operation is possible using two antennas. At reception, the high-frequency radio signal is then mixed with a local oscillator, downconverting the signal to a so-called Intermediate Frequency. Later the analog signal is converted to a digital one, in order to process it in software.

A circular-polarized 5,8 GHz antenna by SpiroNET is used for signal reception. This antenna was also used by Sandvold in his Master's Thesis [San17], as it fulfills the physical layer specification for DSRC.

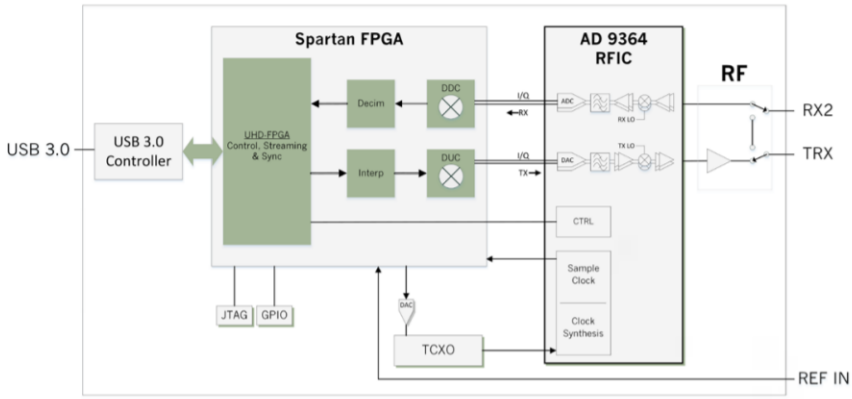

Figure 5.1: The USRP block architecture. Source: [Resa]

Figure 5.2: Setup of the experiment. The USRP is placed next to the RSE and the OBU, in order to receive a good signal power level.

The LO will be tuned as closely as possible to the RF frequency, but there will be a small offset between the LO frequency and the RF frequency. Therefore, a second stage of tuning is performed with the already digital signal. The FPGA block of the USRP accounts for this task [Resa]. After this, a baseband complex signal is delivered via USB connection for further processing. In this project, GNU Radio is used, a digital signal processing tool that will be described in detail in the next section.

The USRP open-software driver is called UHD, and it enables to deploy applications that can address the USRP and use the signal data for further processing. USRP devices are widely used with GNU Radio.

5.2 Signal Processing with GNU Radio

From getting the complex baseband signal, to actually obtain the data contained there, there are still some stages that must be performed. For this, the SDR framework called GNU Radio will be used. This is a free-software tool that permits using predefined blocks (previously created by the user community), and also customized ones, to build a software that performs the desired processing.

GNU Radio also includes a GUI called GNU Radio Companion (GRC). This tool makes a lot easier to develop SDR applications, adding the existing or own blocks to the flowgraph.

GNU Radio blocks can be developed both in Python and C++. When the performance of the block is relevant, C++ is recommended [Rad]. Therefore, most of the signal processing predefined blocks in GNU Radio are written in C++. The most relevant ones include also some documentation, so the integration in the application of interest becomes easier.

One of the advantages of working with GNU Radio to develop signal processing software, is that there is a big community contributing with new developments and creating new blocks for different functions. This is very valuable when working in a project like this one. It is very easy to download and install blocks created by a third party, and also modifying the code of these blocks if necessary. Once installed, they can be added to the GRC flowgraph. Most of the blocks take some parameters as an input, in order to perform their function. For example, a filter will take a cutoff frequency/frequencies value as a parameter, among others. These must be indicated before running the program.

Installing new blocks

Throughout the development of the receiver, some third party blocks have been installed and added to the flowgraph. This is a very easy process in GNU Radio. When a new block is created, a so-called out-of-tree module is generated. This consists mainly on a folder structure with many different files, including the code corresponding to the block logic itself, either in Python or C++. Many blocks can be contained in a single module.

In order to add these third party block or blocks, contained in a module, it's necessary to install the module. The guidelines and commands for this are given in the GNU Radio tutorials [Rad]. Another option is to generate a new out-of-tree module and write brand new blocks. Some guidance about how to write these blocks is also given in the tutorials, along with some flowgraph examples. All this

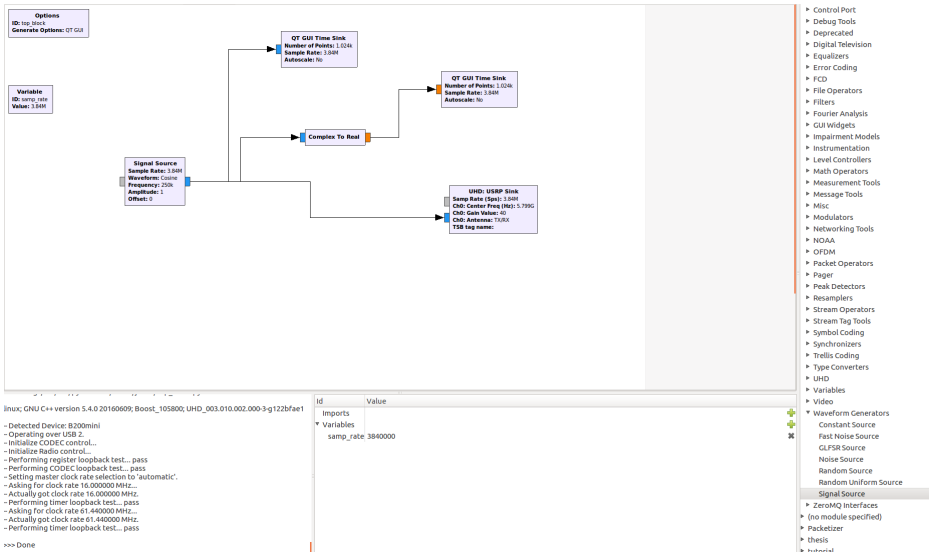

Figure 5.3: A simple flowgraph in GRC. In the right side, a list with the installed blocks. Below the flowgraph, a console providing some output, and a window indicating the parameters of the selected block.

has resulted very useful to gain better understanding about GNU Radio and the possibilities it offers.

5.3 Related Work

As mentioned in the Chapter 2, some previous Master's Thesis at NTNU researched EFC and DSRC, and some part of that work had the focus on building a DSRC transceiver.

Einar Thorsrud worked in his thesis on the implementation of a DSRC link [Tho09], using GNU Radio and creating several signal-processing modules, aiming to build a transceiver and successfully establish a DSRC data exchange, similar to the one performed in the EFC system.

This work was continued later in 2016 by Jonathan Hansen, who aimed to recreate the communication between a RSE and an OBU, using an USRP device and GNU Radio working as the RSE unit, and commercial OBU devices [Han16]. Hansen took many of the blocks used by Thorsrud and edited some of them, with the goal of building a working transceiver software.

One year later, Sandvold reused Hansen's software in his thesis, making also

additional modifications. As stated in the second chapter, his goal was obtaining MAC codes from OBU devices, and he aimed to achieve this using the DSRC transceiver as a RSE unit, in the same way as Hansen did [San17].

Unfortunately, after these projects the DSRC transceiver was not working properly. One of the problems these students were facing, was that they could not obtain a correct EFC signal from an OBU to test the receiver. As their transmitter was not working properly, no answer from the commercial OBU devices was obtained, and the receiver could not be tested with anything but noise. In this thesis, a real DSRC communication is achieved between commercial RSE and OBU devices, so the DSRC receiver can be developed and tested without developing a transmitter. ’

Burst reception

Additionally, many people have built and designed GNU Radio software focusing specifically in reception and transmission of bursty signals. These type of signals, opposite to continuous signals, have time intervals with data transmission (which could correspond to a frame or a packet exchanged by the entities involved in the communication) and intervals where no data is transmitted. During these intervals, the receiver only gets noise and no valuable information is received.

In order to build a DSRC receiver, it’s very relevant to take this into account. The receiver should be able to trigger the time slots with data transmission, and decode the information correctly, following some steps that will be described in the next section. As many of the blocks in GNU Radio are designed for continuous transmissions, many projects have focused on developing burst receivers, reusing some of the predefined modules, modifying them and also creating new ones.

5.4 Building the radio receiver

In this section, the process involving the development of the DSRC receiver will be described. The different steps involved in the signal reception will be described, and the implementation for a receiver in GNU Radio will be presented.

5.4.1 Signal processing in reception

The uplink signal in DSRC is modulated using PSK, a modulation based in shifting the phase of the carrier signal in order to code the different symbols. In DSRC, BPSK is used, where the signal phase shifts between 0 and 180 degrees. This is the most robust PSK modulation, but at the same time in only allows the encoding of one bit per symbol, which makes the bit rate lower than in higher order PSK modulations. However, as the carrier frequency is high, the bit rate achieved is sufficient for the DSRC applications.

To get the bit information from the carrier signal, several steps are involved. As defined previously, the USRP delivers the baseband samples to the GNU Radio software. This baseband signal has to be further processed. Following, the actions that need to be performed, and which will be implemented in GNU Radio, will be described. They are valid both for the uplink, modulated with BPSK, and the downlink, modulated with a 2-level Amplitude Modulation.

Filtering

The first step to perform is filtering of the baseband signal, in order to remove all the frequency components outside the signal bandwidth. This can be achieved by means of a low-pass filter. In this way, much of the white noise associated with the transmission will be removed, as well as another signals whose center frequency is close to the desired one.

Synchronization

This step is very relevant in order to achieve a successful signal reception. As the transmitter and the receiver are different devices, they are not synchronized, and some offsets between the transmitted and the received signal may happen. These must be removed before decoding the signal and obtaining the bitrate. This is achieved using loop blocks that, based on the previous samples of the received signal, can minimize the offset. The synchronization task can be split in two: timing recovery and carrier recovery.

Timing recovery

It's well known that the transmitter and the receiver use different clock signals. This causes that, at reception, the signal symbols are not sampled at the optimal point, therefore increasing the chance of making a wrong decision in the decoder. The goal of the time recovery process is to align the clock in the receiver with the clock in the transmitter, so that the symbols are sampled at its maximum signal point.

Carrier recovery

There are several elements that cause an offset between the transmitted and received signal, in terms of both frequency and phase. These offsets can cause the receiver not to perform a correct demodulation. Among them, we can find the differences between the local oscillators in transmission and reception (same nominal frequency, but some small differences in reality), and also some Doppler-shift effects due to the movement of the OBU [MR]. A carrier recovery phase is necessary to correct this.

Symbol decision

After performing filtering and synchronization, the remaining task is getting the bit information from the baseband signal. In our order 2 modulations (just one bit encoded per symbol), this is simply determining whether they symbol corresponds to a high level (bit 1) or a low level (bit 0). After this, the bit stream can be obtained. However, as in transmission FM0 (downlink) or NRZI (uplink) coding is used, an additional decoding function is necessary to obtain the original data, which can be saved to a binary file.

5.4.2 Implementation in GNU Radio

The implementation of a DSRC receiver in GNU Radio will be described here. Note that there are many different possibilities when it comes to which blocks use, as new blocks are created or modified constantly. In fact, during this project several different configurations have been tested. More details will be given in the next section.

Two flowgraphs have been created, one for the uplink and another one for the downlink. However, the only one outputting bit information is the one corresponding to the uplink. The downlink decoding is slightly more complex, especially due to the use of FM0, so a complete receiver development was not achieved. It was chosen to spend more time working on completing a correct uplink decoding, and solving the problems related to the bursty signal reception. However, the downlink flowgraph is useful to visualize the baseband signal, and the guidelines for building the blocks needed for decoding are given later.

Below, the blocks used for the uplink receiver will be described, together with the values of the most relevant parameters. The differences with the downlink receiver flowgraph will be described afterwards.

USRP Source

This first block is necessary to use the USRP as the signal source for the signal processing software. Here, some important parameters such as the clock rate of the device and the sampling rate must be configured. The selected clock rate is 30.72 MHz, and the sample rate 3.84 MS/s. Ettus recommends choosing a integer even ratio between the clock rate and the sample rate [Resb]. This is because decimation is performed to get the desired sample rate from the master clock rate.

It's also necessary to configure the frequency of the local oscillator, that will be the central frequency of the carrier signal (5.799 GHz for the uplink). Another important value is the gain. After testing several different values, a 40 dB gain is set. As the USRP B200 mini has 2 antenna ports, the one in use has to be specified as well.

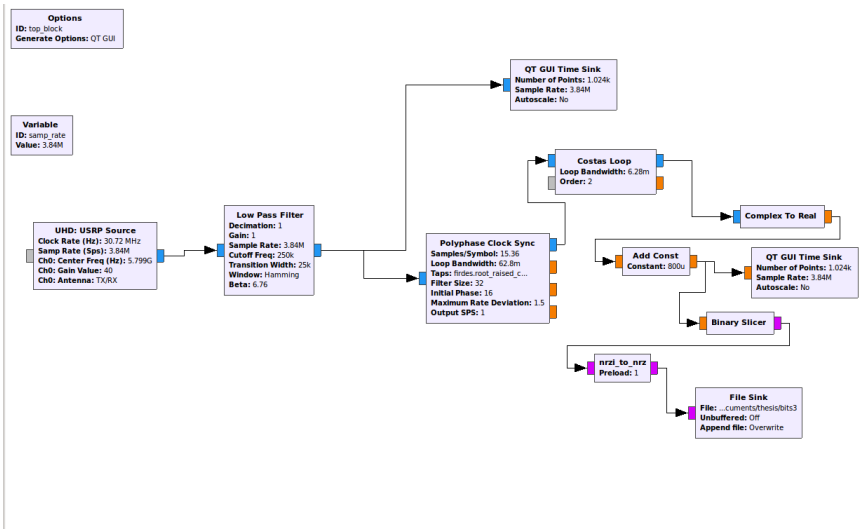

Figure 5.4: The GNU Radio flowgraph for the uplink receiver.

Note that, when connecting the USRP to the computer running GNU Radio, the UHD driver detects automatically the device. Using the "uhd_find_devices" command, the detected USRPs can be listed, and a correct connection can be verified.

Low Pass Filter

This block implements the filtering functionality defined in the previous section. Here, the cutoff frequency is set to 250 KHz, and the sample rate must be set with the same value as in the previous block. A parameter called Transition Width must also be configured. As the filter is not ideal, it will allow to pass some signal power above the cutoff frequency. This value represents the distance between the edge of the passband (where all the signal power passes) and the edge of the stopband (where no signal power passes). It has been set to 25 KHz. A shorter transition width would require more filter taps, therefore more calculations and computation resources.

Polyphase Clock Sync

The Polyphase Clock Sync block accounts for the timing recovery. It can also perform downsampling. In this flowgraph, one sample per symbol is output for further processing. Therefore, both the values of the input and output samples per symbol must be introduced. As the input sample rate is 3.84 MS/s, and the symbol rate is 250 symbols/sec, this ratio is 15.36. For the output, as mentioned before, one sample per symbol is configured.

Some other values, such as the loop bandwidth and the filter taps must be set. Here, the values included in the example flowgraph from the GNU Radio PSK tutorial have been taken.

Costas Loop

This block is used to correct the phase and frequency offset, and perform carrier recovery. It's a software implementation of the phase locked loop (PLL) circuit designed by the engineer John Costas back in 1956 [Cos56]. Here, a loop bandwidth has to be introduced again (like in the Polyphase Clock Sync, this is set to 2pi/100 as indicated in the GNU Radio tutorial), and also the loop order (2 for BPSK).

Add Const

Now, the signal should be ready to be decoded. Here, it's necessary to deal with the bursty nature of the signal. The time slots where no information is sent must be somehow skipped. The approach followed here has been adding a constant value to the signal, so that the white noise present in the no-information time slots never falls below 0, but the symbol transitions during information transmission do. This has to do with the next block, whose functionality will be described right below.

After performing some trial and error, recording the signal several times and obtaining the associated bit rate, a value of $800*10^{-6}$ has been configured. This value has been taken when putting the antenna as close to the OBU device as possible. Note that it might have to be readjusted depending on the antenna distance and other circumstances.

Binary Slicer

As BPSK is the most simple PSK modulation, the bit information contained in the symbols can be extracted just telling whether the sample taken at the highest energy point of the symbol is above or below 0. This is achieved with the Binary Slicer. This block outputs a 1 when the sample value is positive, and a 0 when the sample value is negative. Note that each sample has a size of one octet, so for each bit value one octet is used, where the only relevant bit is the least significant one.

In the slots with no information, the block will always output a 1. However, when decoding the signal from the NRZI format to the original information format, these time slots will be represented with many zeros, because the symbol value doesn't change (more information about the NRZI encoding is given in the Chapter 3).

NRZI decoder

Before saving the bit stream to a file, a decoder block is needed to undo the NRZI coding and get the actual data. Here, the block created by Thorsrud and rewritten by Hansen has been used. Some modifications had to be added to solve some compilation errors found when installing the block. The Python code for this block is included in the attachments, inside a zip file with the full GNU Radio module.

File Sink

Finally, this block is included to record the bit stream to a file. As mentioned before, each information bit is contained in one octet. In this block, the path of the file must be indicated. Also, the override option is active, which means that the previous information will be deleted upon new recording, unless the information is saved to another file (changing the path).

GUI Time Sink

Although this block was not necessary for the signal decoding itself, it allows to visualize the signal on the time domain, which has been very useful during the development process, and also allowed to gain a much better understanding of the signal. Here, the sample rate of the signal is requested in order to adjust the time scale.

5.4.3 The downlink receiver

There are some differences between the uplink and the downlink signal reception. These are related mainly to the difference with the physical parameters of the signal, like carrier frequency, modulation and coding.

In the USRP source block, the central frequency must be set to 5.7975 GHz. The gain is set to 25, because the downlink signal has more power than the uplink one. Therefore, the gain at reception can be reduced.

For the filtering, a band pass filter is used this time. This is due to some spurious low frequency component found in the baseband signal upon reception. It is unknown the source of this component, but it was necessary to remove it to obtain a baseband signal close to the expected one. This effect can be seen in Figure 5.5. The low cutoff frequency is set to 50 KHz, enough to remove the undesired frequency component. The high cutoff frequency is set to 1 MHz, and the transition bandwidth is set to 25 KHz.

After this, the baseband signal can be visualized with a Time Sink block. However, a complete demodulator was not achieved. There are two challenges here: the

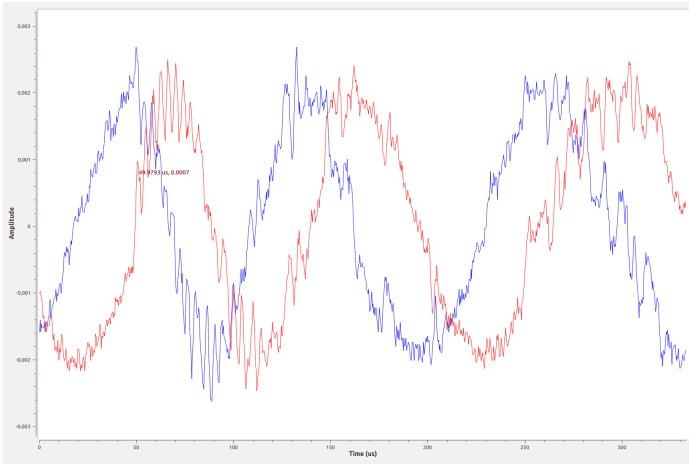

Figure 5.5: The downlink baseband signal (I/Q components), before bandpass filtering.

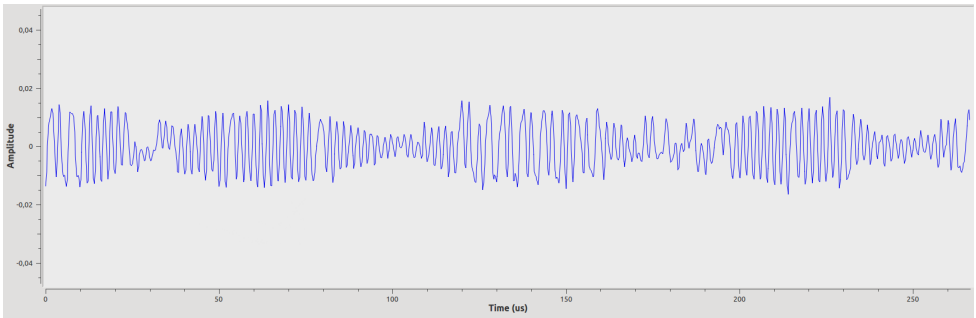

Figure 5.6: The downlink baseband signal (only real component) after filtering.

modulation (amplitude modulation instead of PSK) and the FM0 coding, which includes a signal transition in the middle of the symbol interval, and makes the decoding harder than in the uplink. The baseband signal is quite similar in both cases, with the difference that, due to the extra transition, two samples per symbol must be taken for decision making. Therefore, the synchronization blocks which were used for the BPSK signal should be valid for the downlink as well.

Finally, a decoder block similar to the one used in the uplink receiver has to be built. This block could also be written in Python.

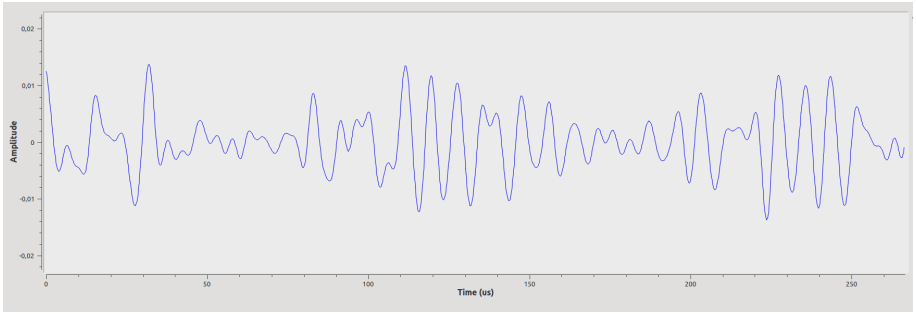

Figure 5.7: The uplink signal after filtering, before synchronization.

5.5 Results and discussion

In this section, the main results obtained with the developed receivers will be presented and discussed, as well as some other relevant outcomes about the developing process.

5.5.1 Reception and decoding

Throughout the receiver development process, many measures of the signal have been taken, with help of GNU Radio's time and frequency GUI sinks. Also, several samples of the binary flow have been saved and examined, with help of a HEX editor. In this project, Bless editor has been used.

Despite trying several different configurations for the receiver, the bit information seems to have many bit errors, comparing to what the content of the VST message should be according to the standards. The structure of both the BST and the VST messages is given in Appendix. Looking at the signal in the time, it is easy to realize that there are some time slots where the power of the wave is that low, that the decision maker block (Binary Slicer) can make wrong bit decisions. This can be seen in Figure 5.7.

It was expected that the synchronization blocks available in GNU Radio could solve this problem, but they don't work as expected with burst-based signals, as they are designed mainly for continuous signal streams. Many other GNU Radio users have reported these issues in the mailing lists, as can be seen in [gnu]. The signal after synchronization is shown in Figure 5.8.

In Figure 5.9, a capture of the Bless editor is given. Here, the bit information captured and saved to a file can be seen. The binary file corresponding to these capture is included as an attachment to the document. As mentioned before, several messages are captured, with some time intervals with no information in between

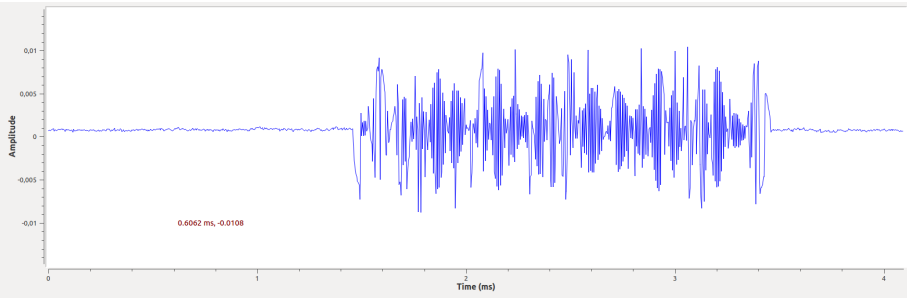

Figure 5.8: The uplink signal after synchronization, with one bit per symbol.

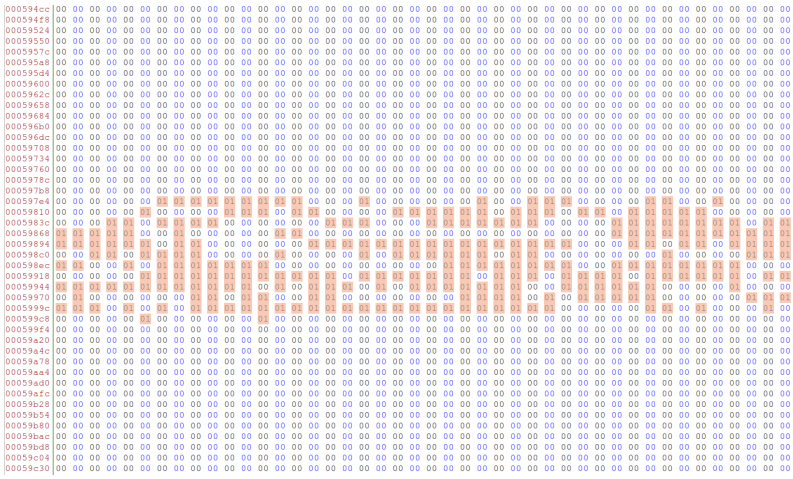

Figure 5.9: A capture of the uplink binary flow.

(represented with a "0" bit). Each of the hexadecimal digits corresponds with 1 bit of information.

In the VST seen in the capture, the "1" bits are marked in orange. It is possible to appreciate the preamble, consisting in several "1" bits after the time slot where no information is sent. However, although the frame length seems correct, it's hard to find coherence with this information and the contents indicated in the standards for a VST. Furthermore, as it can be seen in the binary file, in several of the subsequent captured messages even the preamble is difficult to distinguish. This is a good indicator that the receiver has many errors. The reason for this is surely the fact that it's difficult to make correct bit decisions when the received signal level is not high enough. The cause for this is unsure, but my guess is that a good synchronization is necessary to recover the signal to a level where the decoding can be performed correctly.

The downlink baseband signal suffers the same problem, as can be appreciated in Figure 5.6. Although the decoder for the downlink hasn't been developed, it is expected that the decoded bitflow won't be correct until this problem is solved.

Other approaches for burst signal reception

Some other available projects related to burst signal reception with GNU Radio have been studied. Some of these projects included new blocks for synchronization, which could potentially help for the signal recovery.

One of the studied projects in this area was carried out by a telecommunications student, Thomas Verelst, in 2017 [Ver17]. He designed new out-of-tree modules to implement packet encoding and decoding. He built a correlation estimator block, enhancing the implementation existing in GNU Radio, which is deprecated. He uses tags (a signaling procedure implemented in later versions of GNU Radio) to send synchronization information to both the timing and carrier recovery blocks (Polyphase Clock Sync and Costas Loop). Another interesting project about PSK burst demodulation was studied [bur]. Here, a very similar approach is followed, and the shown results seem to be promising working with file source signals.

Both of the projects include newly developed out-of-tree modules and blocks, which were downloaded, installed, and tested for the DSRC receiver. Although both works are very interesting and add value in order to develop a burst decoder, the blocks don't seem to solve the previously mentioned problem in this project. It's important to remark that these projects didn't have a real physical signal as an input for the decoder, but a virtual one generated in GNU Radio and blocks which simulate the channel. It is logical to think that more problems arouse when working with signals from real systems, specially when it comes to synchronization and baseband signal recovery.

Although none of these blocks were added to the design included as an attachment, it would be an interesting task to perform a further study on these projects, including the possibility of modifying some of the blocks to adapt them for a DSRC receiver. Al the GNU Radio project available on the internet are free software, which makes possible to access and modify blocks generated by many contributors. This is probably the greatest advantage about working with this tool.

Chapter 6
Signal replaying and interference in EFC

Additionally to developing a DSRC receiver with the goal of eavesdropping the communication involved in an EFC transaction, some other tests were performed with the available SDR tools. The focus was put in analyzing the possibility of capturing one of the signals involved in the communication, and later replay it to see the effect in the system. Also, the effect of interference in the communication was studied.

6.1 Recording and replaying signals

It is interesting to analyze the possibility of performing a replay attack in EFC. Since the communication is not encrypted, and the authentication and message integrity mechanisms are weak or sometimes even non existent, this could represent a big threat.

For this, a rather simple setup in GNU Radio was prepared. This consisted in using the same uplink and downlink receivers previously used, but removing all the blocks after filtering and saving the results to a file. After this, another flowgraph

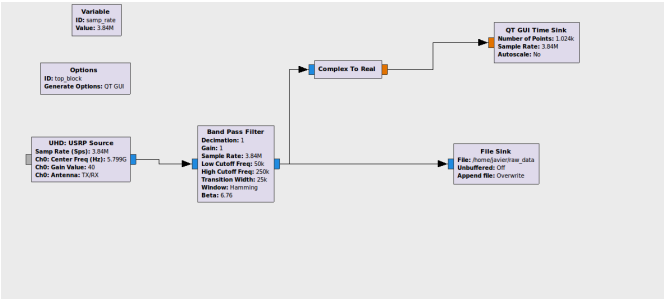

Figure 6.1: The flowgraph for signal recording.

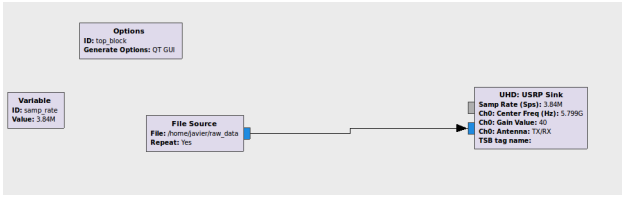

Figure 6.2: The flowgraph for re-transmission of the recorded signal.

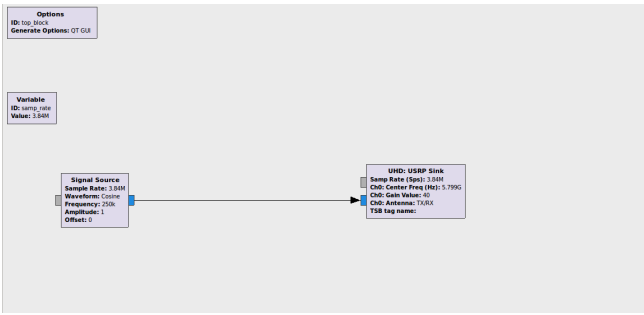

Figure 6.3: The interference generator flowgraph.

was created using the same file as a source, and transmitting the signal directly.

Unfortunately, after trying to record both the uplink and downlink signals and replay them, no effect is appreciated in the RSE (console message indicating a VST reception). In the led-OBU, the led indicating a signal received in the carrier frequency is intermittently on, but the led indicating a correct BST reception remains off. It's probable that, if the signal receiver had many problems to decode the signal, the replay setup doesn't work due to the same reception issues. It's likely that the recorded signal is not clean enough to be transmitted again, and be acknowledged by any of the legitimate devices.

6.2 Interference issues and communication interruption

Another important issue to be aware of, is the potential interference between EFC and other wireless technologies. The draft report [Com] published by the Electronic Communications Committee (ECC) gives an insight into this issue. The Radio Local Area Networks (RLAN) applications used in cars co-exist in the same frequency band as EFC, which could cause problems in the tolling service availability.

To test the effect of interference in DSRC signals, a transmitter setup is prepared with GNU Radio and an USRP. Here, a cosine function generator is used, which

modulates a carrier at 5.7975 GHz to cause interference in the downlink, or at 5.799 GHz in the uplink. The GNU Radio flowgraph is shown in Figure 6.3.

After testing the transmitter, both in the downlink and uplink frequencies, the communication between the RSE and the OBU was interrupted. When emitting power in the uplink carrier frequency, the RSE didn't receive any VST from the OBU. The same happened when causing an interference in the downlink, in this case because the OBU didn't receive correctly the BSTs.

This tests show that the interference in the 5.8 GHz band can affect the EFC communications in the toll roads. Therefore, it would be interesting to be aware of this problem and make a further research on it, and in case of being necessary, work on possible solutions.

Chapter 7
Conclusions

This Master's Thesis has focused on contributing to the analysis of the EFC system security, focusing on building a receiver for eavesdropping of the wireless communication between a RSE and an OBU, and analyzing the feasibility of this task using rather cheap and available tools. While a bit-error free receiver has not been successfully developed, the signal processing program can get much information about the signal, down to bit-level, and could be used for further work on the same area.

Developing a customized device to receive and transmit DSRC signals is not an easy task. It requires much DSP knowledge, together with C++ programming skills and a study on the standards. However, an attacker with the appropriate skill set, but at the same time using publicly available tools, can get a lot of information about the communication, and eventually develop a device which manages to eavesdrop the data exchange in the toll roads. The vulnerabilities of the access control and authentication mechanisms can also make key brute-forcing feasible, as shown by previous works. This can lead to undesirable consequences for the toll road service providers.

GNU Radio is a good and free tool for DSP, but its blocks have been mainly written by volunteer contributors. In addition, most of the existing projects related to burst signal detection were developed with test virtual signals, instead of a real signal of a working system. Therefore, it can be hard to integrate existing blocks for decoding a complex signal such as the one in DSRC. The most appropriate solution seems to be developing own blocks for specific applications, but as stated before, this is a task that requires C++ programming expertise and a broad background in DSP.

Additionally, the possibility of performing a signal replay attack have been explored. Despite not being able to successfully replay downlink or uplink bursts, this threat is real and should be taken into consideration. Some tests done with the USRP and with GNU Radio show that a device emitting power at the same

frequency spectrum as DSRC can interrupt the communication. This shows that, apart from a potential delivered interference, some of the other wireless applications using the same non-licensed band can disturb the communication.

7.1 Further work

There are several areas where future projects working with EFC security could focus on. Firstly, it would be interesting to try to enhance the receiver, so that the signal can be decoded without bit errors. Writing new synchronization blocks, or modifying the ones already created by third parties, could be a solution for the problem. If a DSRC transceiver is developed, it could open the door to perform many attacks against the system.

Another potential task would be to test the feasibility of brute-forcing the AcK and the AuK. Sandvold [San17] proposed in his Master's Thesis a time-memory trade off to obtain the AuK, performing a chosen-plaintext attack using a customized RSE. This wouldn't be possible with Security Level 1 in EFC, as the RSE must also authenticate using a valid AuK. Therefore, it could be interesting to research about attacking the access control, using a customized OBU to communicate with an RSE, and obtaining plaintext-ciphertext pairs.

It's also remarkable that the VST message is not integrity-protected, so its content could be modified and still be accepted by the RSE, as long as its format is valid. This can also be a good potential research. Finally, more time can be spent on analyzing the effects of signal replaying in the system. For all this, the legacy equipment used throughout this project can be useful.

References

[Adm14] Norwegian Public Roads Administration. EN 15509 AutoPASS On-board Equipment. Functional and Technical requirements. 2014.

[AG14a] Kapsch TrafficCom AG. Installation Manual On-Board Unit Programming Station OPS-1955 GT_IF & L7. 2014.

[AG14b] Kapsch TrafficCom AG. Programmer's Manual API Layer7 and BAC Interface Specification. Kapsch TrafficCom AG. 2014.

[bur] Burst PSK Modem with LDPC Coding in GNU Radio using Message Passing and Eventstream. https://oshearesearch.com/index.php/2015/04/02/burstpskmodem.

[CEN03a] CEN. EN 12795:2003. Road transport and traffic telematics – Dedicated short-range communication (DSRC) – DSRC data link layer: medium access and logical link control. 2003.

[CEN03b] CEN. EN 12834:2004. Road transport and traffic telematics (RTTT) – Dedicated short-range communication – DSRC application layer. 2003.

[CEN04a] CEN. EN 12253:2004. Road transport and traffic telematics – Dedicated short-range communication – Physical layer using microwave at 5,8 GHz. 2004.

[CEN04b] CEN. EN 13372:2004. Road transport and traffic telematics (RTTT) – dedicated short-range communication – profiles for rttt applications. 2004.

[CEN07] CEN. EN 15509:2007. Road transport and traffic telematics – Electronic Fee Collection – Introperability application profile for DSRC. 2007.

[CEN11] CEN. EN 14906:2011. Electronic Fee Collection – Application interface definition for dedicated short-range communication. 2011.

[CEN15] CEN. Electronic Fee Collection — Assessment of security measures for applications using dedicated short-range communication. 2015.

[Com] Electronic Communications Committee. ECC Report 277. Use of SRD applications in cars in the band 5725-5875 MHz.

[Com15] European Commission. Study on "State of the Art of Electronic Road Tolling". pages 3–6, 2015.

[Cos56] John P. Costas. Synchronous Communications. 1956.

[des] Data Encryption Standard. Wikipedia. https://en.wikipedia.org/wiki/data_encryption_standard.

[ETS04] ETSI. Electromagnetic compatibility and Radio spectrum Matters (ERM); Road Transport and Traffic Telematics (RTTT); Dedicated Short Range Communication (DSRC) transmission equipment (500 kbit/s / 250 kbit/s) operating in the 5,8 GHz Industrial, Scientific and Medical (ISM) band; Part 1: General characteristics and test methods for Road Side Units (RSU) and On-Board Units (OBU). 2004.

[gnu] GNU Radio mailing list. https://lists.gnu.org/archive/html/discuss-gnuradio/2015-12/msg00071.html.

[Han16] Jonathan Hansen. Wireless USRP Test-bed for DSRC applications. *Master's Thesis, NTNU*, 2016.

[MR] Vesa Tuukkanen Markku Renfors, Lauri Anttila. Maximum Likelihood Estimation Techniques for Symbol Timing and Carrier Phase Recovery. http://www.cs.tut.fi/kurssit/tlt-5806/synch.pdf.

[Rad] GNU Radio. GNU Radio Tutorial. https://wiki.gnuradio.org/index.php/main_page.

[Resa] Ettus Research. USRP B200 Mini Datasheet. https://www.ettus.com/content/files/usrp_b200mini_data_sheet.pdf.

[Resb] Ettus Research. USRP Hardware Driver and USRP Manual. General Application Notes. https://files.ettus.com/manual/page_general.html.

[San17] Sverre Turter Sandvold. Attacking Message Authentication Codes in EFC Using Rainbow Tables. *Master's Thesis, NTNU*, 2017.

[Tho09] Einar Thorsrud. Programvaredefinert radio - Mulige hyllevareløsninger for DSRC-anvendelser. *Master's Thesis, NTNU*, 2009.

[Ver17] Thomas Verelst. Implementation of a packet encoder/decoder pair in the GNU Radio framework. *Semester Project, École Polytechnique Fédérale de Laussane*, 2017.

Appendix

Structure of a BST and a VST

In this appendix, the structure of a BST and a VST will be given, with all its parameters and size. All the information is taken from EN 18234 [CEN03b] and EN 14906 [CEN11].

Note that both the BST and the VST are contained inside a LPDU in the link-layer frame, as explained in Chapter 3.

Table A.1: Structure of a BST

Parameter	**Downlink**
Rsu Beacon ID	6 octets
Time	4 octets
Profile	1 octet
Mandatory Application List (sequence)	1 octet
Non-mandatory Application List (sequence, optional)	1 octet
Profile List (sequence)	1 octet

Table A.2: Structure of a VST

Parameter	**Downlink**
Fill	4 bits
Profile	1 octet
Application List (sequence)	
–DSRC Application Entity ID	1 octet
–DSRC-EID	1 octet
–Application Context Mark	16 octets
Obe Configuration	
–Equipment Class	2 octets
–Manufacturer ID	2 octets
–OBE Status	2 octets