ELEC5881M Final Report

Signal Processing, Transmission and Reception by using Software Defined Radio and Open Source Tools

Miguel Angel Torres Lázaro

Student ID: 200912040

Submitted in accordance with the requirements for the degree of
Master of Science
in Embedded Systems Engineering

Supervisor: Dr. Des McLernon

Assessor: Mr. Ahmed Lawei

The University of Leeds

School of Electronic and Electrical Engineering

August 2016

Declaration of Academic Integrity

The candidate confirms that the work submitted is his/her own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated in the report. The candidate confirms that appropriate credit has been given within the report where reference has been made to the work of others.

This copy has been supplied on the understanding that no quotation from the report may be published without proper acknowledgement. The candidate, however, confirms his/her consent to the University of Leeds copying and distributing all or part of this work in any forms and using third parties, who might be outside the University, to monitor breaches of regulations, to verify whether this work contains plagiarised material, and for quality assurance purposes.

The candidate confirms that the details of any mitigating circumstances have been submitted to the Student Support Office at the School of Electronic and Electrical Engineering, at the University of Leeds.

Miguel Angel Torres Lázaro

Date: 02/08/2016

Table of Contents

Abstract

Traditional radio devices have a limited functionality which can only be modified by making changes in the hardware. In this sense, any improvement in the flexibility of a traditional radio device implies a considerable increase in production costs. On the other hand, software defined radio (SDR) systems provide a relatively inexpensive and effective solution, allowing multiple functionality that can be enhanced just by upgrading the software. The present report describes the implementation of four SDR systems by using a laptop with the software GNURadio, an RF transmitter board and a low-cost RF receiver device. This work is focused on the design of transmitters and receivers that are able to perform modulation techniques for analogue signal (FM) and digital signal (QPSK).

List of Abbreviations

SDR	Software defined radio
GPP	General purpose processor
FPGA	Field programmable gate array
GRC	GNU Radio Companion
VCO	Voltage controlled oscillator
WBFM	Wide band frequency modulation
RF	Radio frequency
LO	Local oscillator
DSP	Digital signal processor
SBC	Single board computer
DAC	Digital to analogue converter
ADC	Analog to digital converter
FSK	Frequency shift keying
QPSK	Quadrature phase shift keying
PSK	Phase shift keying
ppm	Parts per million
ppb	Parts per billion
RC	Raised-cosine
RRC	Root-raised-cosine
OOK	On-Off keying
RF	Radio frequency
BB	Base band
USB	Universal serial bus
KLE	Kilo logic elements

IC	Integrated circuit
IF	Intermediate frequency
DVB-T	Digital video broadcasting terrestrial
SPS	Samples per second
KSPS	Kilo samples per second
MSPS	Million samples per second
GSM	Global system for mobile communication
ITU	International Telecommunication Union
SoC	System on chip
PLL	Phase locked loop
VCTCXO	Voltage controlled temperature compensated oscillator

Chapter 1
Introduction

Traditional radio devices have a limited functionality which can only be modified by making changes in the hardware. In consequence, any improvement in the flexibility of a traditional radio device implies a considerable increase in production costs. On the other hand, software defined radio (SDR) systems provide a relatively inexpensive and effective solution, allowing multiple functionality and the possibility of enhancing it just by upgrading the software. Nowadays, the concept and technology of SDR are used in the development of techniques that aim to the optimisation of the use of radio frequency (RF) spectrum, such as adaptive radio, cognitive radio and intelligent radio.

The concept of SDR involves a group of software and hardware technologies where most operating functions of the system are implemented on modifiable software. This software can run on different platforms such as a general purpose processor (GPP) or an embedded processor.

The present work is focused on the design, implementation and testing of eight SDR systems that use specialised hardware for the RF front-end and back-end stages and a laptop to run the software .These systems can be used for transmitting and receiving analog and digital data. It includes a review of the technical specifications of the hardware used in the project, a detailed description of the software design, and analysis of results. This report presents a practical approach since it covers issues that are found in actual communication systems (such as carrier synchronisation and symbol synchronisation) and describes the use of 'GNURadio', which is a standard software used for SDR systems design.

The body of the report is divided into four main chapters, which provide the foundations of SDR, a description of the methodology used for the implementation of the project, and analysis of the results obtained during the testing.

Chapter 2 is meant to provide the background required for the full understanding of the project. This chapter covers basic concepts of communications theory, advanced techniques of digital signal processing, and parameters related to hardware performance.

Chapter 3 provides a detailed description of the tools used for the design and implementation of the project. This chapter includes a thorough technical description of the

devices used for transmission and reception, an explanation of the method used to calibrate these devices, and the basics of the software GNU Radio Companion.

Chapter 4 describes the process of design, implementation and testing of SDR systems with the software GNURadio. This chapter is focused on receivers and transmitters that work with two specific modulation techniques: frequency modulation (FM) and quadrature phase shift keying (QPSK).

Finally, Chapter 5 discusses the results and conclusion obtained from the tests performed in the previous chapter.

Chapter 2
Foundations

The present work is focused on the implementation and testing of four software defined radio communication systems based on different modulation techniques. Since these systems will be tested by performing an actual communication with proper transmitting and receiving devices, it will be necessary to consider some practical issues that are usually ignored when running simulations.

In order to clearly understand the content of this report it is necessary to know some basic concepts, which are described in the rest of the chapter.

2.1 Communication system

In order to explain what a communication system is, first it is necessary to define the scope of communication. Communication, also known as telecommunication, is defined as any transmission and reception of signals of any nature, typically electromagnetic, that contain symbols, sounds, images or any other type of data that is required to be sent to a remote place. In telecommunications, a communication system is formed by three basic elements: a transmitting station, which sends the data; a receiving station, which receives the data and uses it; and a channel, which is the medium that connect the transmitting station to the receiver. These three basic elements are suitable for one-way operation. However, communication systems that perform two-way operations require both a transmitting and a receiving station.

Any modern digital communication system counts with various stages to perform the transmission, reception and processing of the signal (see Fig 2.1). On one hand, the most important stages in a transmitter are source encoder, channel encoder, cipher, modulator, intermediate frequency (IF) stage, radio frequency (RF) stage and transmitter antenna. On the other hand, the most important stages in a radio frequency receiver are the receiver antenna, RF stage, IF stage, demodulator, decipher, channel decoder, and source decoder. Additionally, when working with digital signal, an equalizer is included at the receiver in order to reduce the effects of multipath. Every communication system counts with a processing central (or data processor) that generates the data to transmit or analyses the received data. Every stage is reviewed in further detail in the following lines.

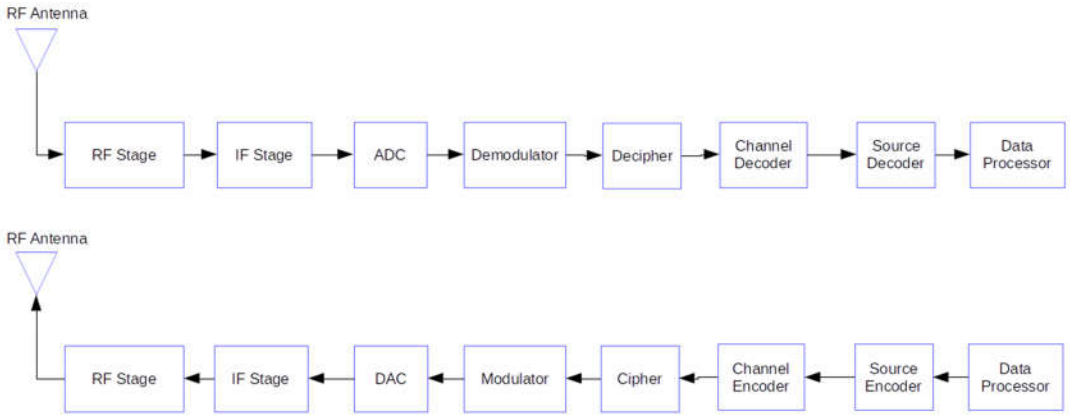

Figure 2.1 Block diagram of a typical trasnmitter and receiver communication system.

2.1.1 RF Stage

The RF stage in a communication system involves the hardware in charge of the conditioning and transmission/reception of the signal. The generic terms 'RF front-end' and 'RF back-end' are commonly used to indicate the circuitry formed by the antenna and the RF stage in a receiver and a transmitter, respectively.

In a receiver, the RF front-end is formed by an antenna, a low noise amplifier, an image rejection filter and a frequency mixer (see Figure 2.2). The antenna receives the RF signal and a tiny voltage is generated in the terminals of the antenna. This signal has to pass through a low noise amplifier in order to enhance only the signal of interest. Then, the signal passes through a filter that supresses RF images before the tuning is performed. Finally, a frequency mixer performs the down-conversion of the signal. The down-conversion is performed by multiplying the received signal by a sine carrier, which is generated by using a local oscillator.

In a transmitter, the RF back-end is formed by the antenna, a frequency mixer, a power amplifier, and an antenna (see Figure 2.2). The frequency mixer performs the up-conversion of the signal and the amplifier increases the power of the signal before it is transmitted through the antenna.

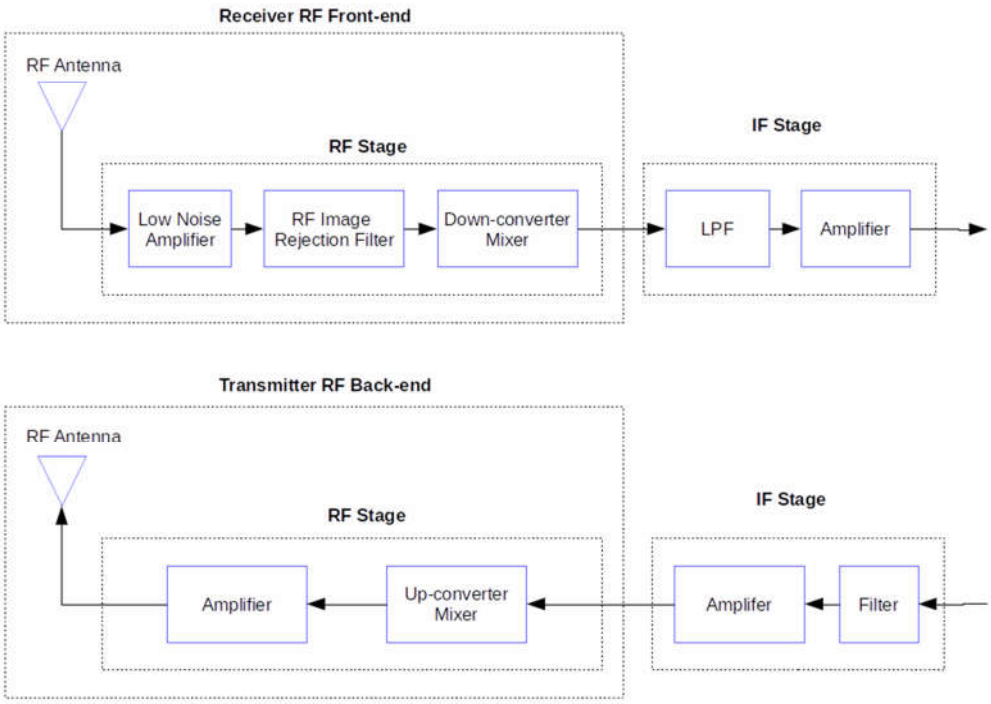

Figure 2.2 Detail of the RF Stage and IF Stage in a communication system.

2.1.2 IF Stage

As shown in Figure 2.2, in a receiver the IF stage is formed by a lop-pass filter that eliminates the undesired components present after the mixer and an amplifier.

In a transmitter, the IF stage is formed by a filter and an amplifier that prepare the signal before it enters to the RF mixer.

2.1.3 Analog to digital converter / Digital to analog converter

In a receiver, after being filtered and amplified, the IF signal passes through an anti-aliasing filter to eliminate the frequency components that exceed the limitations of the sampling rate of the ADC. Then, the ADC digitises the signal in order to perform digital signal processing techniques in the following stages.

In a transmitter, the modulated signal passes through a digital to analogue converter (DAC) to turn the digital values of the output into analogue values. Then, the resultant signal passes through a smoothing filter to eliminate the undesired frequency components. The precision of the data will depend on the resolution of the DAC.

2.1.4 Source encoder/decoder

Since the data in a digital system is represented by bits, the bits containing information should be transmitted through the channel. According to this, if a system needed to transmit a greater amount of information in the same time, then it would have to increase its baud rate in order to send more bits per time unit. However, this is not always possible since every country has its own laws that regulate the use of the spectrum. An alternative solution is to use the same baud rate, but sending as few bits as possible to represent the data. The process for optimizing the number of bits used might generate some redundancy. The process used to efficiently convert digital value into a sequence bits is called source encoding. Therefore, in simple terms, the source encoder compresses the data in order to reduce the number of bits per second required. The resultant bit sequence generated by the source encoder is called information sequence.

2.1.5 Channel encoder/decoder

During the transmission, the signal get distorted due to the effects produced by the noise and interference that are present in the transmission channel . In order to reduce the probability of error in the data, some redundancy is introduced in the information sequence. Later, the receiver can check the redundancy information in order to verify if that the data has no errors. Therefore, channel encoding is the process of adding extra bits to the information sequence at the transmitter in order to allow the system to perform error correction at the receiver.

2.1.6 Cipher/decipher

In several applications the transmitted signal contains confidential information and users do not want other people to be able to pick up the signal and get the information. In these cases it is necessary to perform some encryption algorithm in order to protect the information from undesired receivers. In this sense, a cipher implements an algorithm that encrypts the information in order to make the communication safer.

2.1.7 Modulator/demodulator

The power of the transmitted signal depends, basically, on the signal amplitude, signal frequency and the size of the transmitter antenna. However, in most applications if the signal was transmitted with its original frequency (baseband) then massive antennas would be necessary to perform the communication. An alternative solution is to increase the frequency of the signal in order to increase its power without increasing the antenna size. Modulation is the process of increasing the frequency of the signal containing data, also known as modulating signal, by mixing it with other signal that has a higher frequency, also known as

carry signal, in order to transfer the data signal over a bandpass channel. Currently there are various modulation techniques and they can be classified in analogue modulation techniques and digital modulation techniques. On one hand, analogue modulation techniques include the two classic methods amplitude modulation (AM) and frequency modulation (FM), both used for radio broadcasting. On the other hand, digital modulation techniques include frequency shift keying (FSK) and phase shift keying (PSK), among others.

2.2 Software Defined Radio

Software defined radio (SDR) is a wide term that refers to any radio communication system in which all or most of the functional blocks associated with the physical layer (such as the modulator or the channel encoder) are implemented in software by using algorithms of digital signal processing. The components that form the radio frequency front-end and the intermediate frequency stage count with their own hardware. The fact of having most of the blocks implemented in software gives the communication system the advantage of being more flexible compared to a system that uses an specific hardware. For example, if a traditional communication system needs to receive information from three different transmitters, and these transmitters use different techniques to modulate their signals, then the communication system will require specialized hardware for the demodulation of every signal. However, if the demodulator is implemented in software, the demodulation algorithm can be modified to work with different demodulation techniques when it is required.

The scope of SDR covers a collection of software and hardware technologies. Most operating functions of the system are implemented on modifiable software operating on programmable devices such as a general purpose processors (GPP), digital signal processors (DSP), programmable system on chip (SoC), field programmable gate array (FPGA). [www01]

The adoption of SDR has generated benefits for various sectors such as radio equipment manufacturers and radio service providers.

SDR allows radio equipment manufacturers to derive various products by using a common hardware architecture, reuse parts of the software for different products and fix errors just by re-programming the device. In consequence, the production costs, development time and maintenance costs are considerably reduced.

Radio service providers take advantage of the flexibility and lower cost of the SDR in several ways. On one hand, they can add new features to existing infrastructure without requiring major new investments. On the other hand, they can use a common hardware

platform for multiple markets. In consequence, radio service providers can reduce the operating and logistical support costs.

Additionally, SDR has served as base for the development of new technologies such as adaptive radio, cognitive radio and intelligent radio. The development of these technologies has great relevance nowadays since they allow a communication system to optimise the use of the frequency spectrum that is available at a specific time.

2.3 Frequency deviation

Oscillators are widely used in electronics, particularly for generating clock signals in digital circuits. An ideal oscillator is supposed to generate a periodic waveform with a nominal frequency which is defined by the manufacturer. However, in practice the real frequency of the generated waveform drifts from the nominal value. This difference between the actual value of the frequency and its prescribed value is known as frequency deviation and it is commonly expressed in parts per million (ppm). The maximum difference between the actual frequency and its nominal value (∆f) is defined by equation 1.1.

$$\Delta f(Hz) = \pm \frac{Nominal\_Freq(Hz) \times Freq\_Deviation(ppm)}{10^6} \quad (1.1)$$

For example, if a crystal oscillator with a nominal frequency of 16 MHz has a frequency deviation of 8 ppm, then the maximum difference between the nominal frequency and the actual frequency will be ±128 Hz.

In order to increase the frequency accuracy, some oscillators are temperature compensated. In these cases the frequency deviation is lower than 1ppm and, for this reason, it is usually expressed in parts per billion (ppb).

Frequency deviation is an issue that affects communication systems, particularly at the tuning stage. This stage generates a carrier by using a local oscillator (LO) in order to pick up signals from a specific frequency band. Since the LO is affected by frequency deviation, there will be an error margin between the desired centre frequency (nominal frequency) and the actual centre frequency. For this reason, frequency deviation is considered an important index during the system setup. [1]

2.4 Multi-rate sampling

Many practical applications in communication systems require to change the original sampling rate of a signal. On one hand, the computational cost of the implementation of a finite impulse response (FIR) filter is closely related to its sampling rate, so a decrease in the sampling rate is desired. On the other hand, some modulation techniques (FM, for instance) increase the bandwidth of the original signal and, consequentially, it is necessary to increase the sampling rate in order to avoid aliasing.

The sampling rate conversion process can be considered as a filtering operation (see Fig 1.2). The filter that perform this operation is known as resampler. Considering a resampler with an input signal x(n) and an output signal y(m), where the signals are sampled at F_X and F_Y samples per second, respectively, the ratio F_Y / F_X can be expressed as in equation 1.2.

$$\frac{F_Y}{F_X} = \frac{I}{D} \tag{1.2}$$

Before continuing, it is important to take a couple of considerations in the equation: First, the ratio F_Y / F_X is constrained to be rational; second, 'I' and 'D' are relatively prime integers. After these considerations two special cases can be found. The first case is when 'I' is equal to 1 and, therefore, the sampling frequency is decreased (down conversion) by a factor 'D'. The second case is when 'D' is equal to 1 and, therefore, the sampling frequency is increased (up conversion) by a factor 'I'. Technically speaking, the first case is called decimation, and the second case is called interpolation. [2]

2.4.1 Decimation

Decimation is, basically, the process of decreasing the sampling rate (down conversion) by a constant factor. Even though in practice the decimation factor can take rational values, in the present work only integer values will be considered in order to keep calculations simple during the analysis.

It is important to clarify that downsampling by a factor 'D' implies that only 1 out of every D original samples will be taken, while the other D-1 samples will be discarded.

Considering the resampler shown in Figure 1.2 , the bandwidth of the signal x(n) should be lower than $2F_Y$ to avoid aliasing. In order to ensure this condition, the signal usually

passes through a low-pass filter (LPF) before the sampling rate conversion is performed. Usually the term decimation involves both the filtering and the down conversion process.

2.4.2 Interpolation

Interpolation is, basically, the process of increasing the sampling rate (up conversion) by a constant factor. Even though in practice the interpolation factor can take rational values, in the present work only integer values will be considered in order to keep calculations simple during the analysis.

It is important to clarify that upsampling by a factor 'I' implies adding I-1 new samples between two original samples. This is usually achieved by expanding the original signal by inserting samples with zero value between two original samples.

A negative effect of raising the sampling rate by inserting zeroes between two original samples is that, in the frequency domain, spectral images of the original signal appear at integer multiples of the original sampling rate. For example, considering the resampler shown in Figure 1.2 working with an interpolation factor 'I' , the spectrum of the output signal y(m) will be equal to the spectrum of the original signal x(n) repeated in the frequencies $F_X/2$, $3F_X/2$, $5F_X/2$, and so on. However, since only the frequency components in the range $0 < F_Y < F_X/2$ are relevant, all the images located above $F_X/2$ can be supressed by passing the signal through an LPF after the sampling rate conversion is performed. Usually the term interpolation involves both the up conversion process and the filtering.

2.5 FM Emphasis

Is a technique used to improve the signal to noise ratio in a FM modulated signal. It consist in the reduction of the high frequency components in the transmitter (pre-emphasis) and the recovery of the original waveform in the receiver (de-emphasis).

Chapter 3
Methodology

This chapter provides a detailed description of the tools used for the design and implementation of the SDR that will be described in Chapter 4. This description involves both hardware aspects and software aspects. The hardware description will cover technical details of the transmitting and receiving devices in order to identify their limitations and consider them when designing an SDR system. The software description will cover the tools used for the calibration of the transmitting and receiving devices and the software used to implement the functional blocks in the SDR systems.

3.1 Hardware description

The transmitter is formed by two elements: a transmitting device and a laptop. The transmitting device contains the hardware corresponding to the RF back-end circuitry and the laptop contains the CPU that will generate the data and perform the digital signal processing (e.g. modulation, filtering and multi-rate sampling). The laptop communicates to the transmitting device through a universal serial bus (USB) port.

On the other hand, the receiver is also formed by two elements: a receiving device and a laptop. The receiving device contains the hardware corresponding to the RF front-end circuitry and the laptop contains the CPU that will perform the digital signal processing (e.g. demodulation, filtering and multi-rate sampling) and data storage. As in the case of the transmitter, the laptop communicates to the receiver device through a USB port.

In order to reduce interferences during the communication the distance between the transmitter and the receiver antenna is 50 cm.

In order to understand the advantages and limitations of the transmitting and receiving devices used in the present work, a detailed description of these devices is provided in the following sections.

3.1.1 Transmitting device features

The transmitting device used for this project is the bladeRF x40 transceiver board, produced by the Nuand group. The bladeRF, shown in Figure 3.2, is an electronic board specially designed for SDR applications. The most relevant technical specifications for the present project are the following[www02]:

- Fully powered by USB bus.
- Supports SuperSpeed USB 3.0
- Frequency range: 300 MHz - 3.8 GHz .
- Independent RX/TX 12-bit 40 million samples per second (MSPS), quadrature sampling.
- Factory calibrated voltage controlled temperature compensated oscillator (VCTCXO) tuned at +/-26 ppb.
- On-board 200MHz ARM Cortex-A9 processor with 512KB embedded SRAM.
- On-board 40 Kilo logic elements (KLE) Altera Cyclone 4E Field programmable logic array (FPGA).
- Software support for Windows, Mac, Linux and GNURadio.

Figure 3.1 Nuand bladeRF transceiver board used in the transmitter.[www02]

In order to provide a more detailed description, a functional block diagram of the bladeRF board is shown in Figure 3.3. The signal data and configuration parameters (i.e. sampling rate, centre frequency and gain) are transmitted from the laptop and received by the FX driver integrated circuit (IC). The FX3 is a USB peripheral controller that supports SuperSpeed USB 3.0. This driver sends the information to the Altera Cyclone IV system on

chip (F484 FPGA), which uses the configuration parameters to set the voltage that controls the frequency of the VCTCXO (through a DAC) and sends the signal data to the LMS6002D transceiver. Finally, a solid state switch connects the output of the transceiver to the transmitter antenna.

When the board is used as a transmitter, then the signal data flows in opposite way (from the receiver antenna) but the configuration parameters flow in the same way. It is important to indicate that the clock signals for the whole system are provided by a clock buffer (Si5330) and a VCTXO.

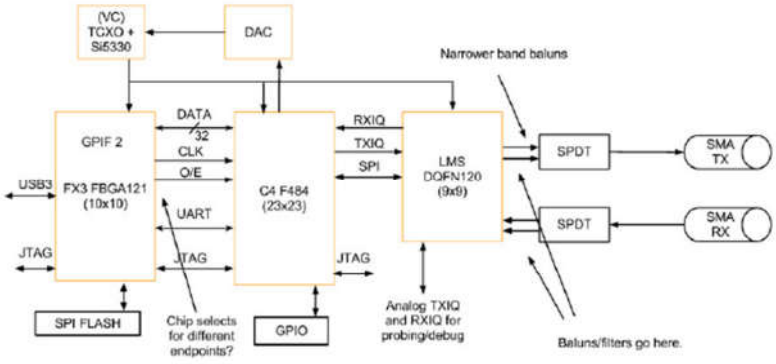

Figure 3.2 Block diagram of the Nuand BladeRF hardware architecture.[www02]

As shown in the diagram in Figure 3.4, the LMS6002D contains the DAC, filters and RF back-end required for transmitting. The signal data values come modulated in quadrature and each component (I and Q) passes through its corresponding DAC and filter. Then, the I/Q signal is demodulated by using a quadrature mixer and the resulting signal is transmitted.The carrier of the mixer is generated by a PLL that receives its clock signal from the VCTCXO mentioned above.

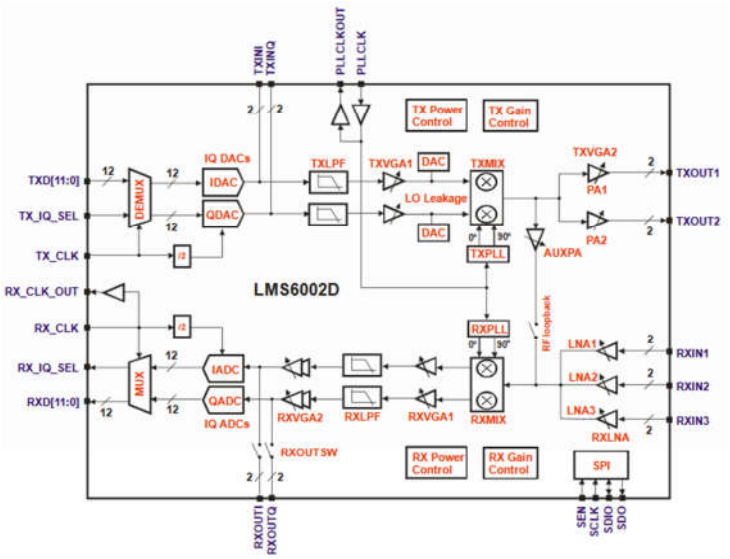

Figure 3.3 Functional block diagram of the LMS6002D transceiver.[www02]

3.1.2 Receiving device features

The receiving device used for this project is the R820T2 USB dongle. Since this device is based on the tuner RTL2832U (manufactured by Realtek) and it is widely used in SDR application, it is also known as RTL-SDR.

The R820T2, shown in Figure 3.5, is produced by NooElec, and it was originally meant to work as a digital video broadcasting terrestrial (DVB-T) receiver. However it is more popular for being used as a low cost receiver in SDR applications. The most relevant technical specifications for the present project are the following [1]:

- Fully powered by USB bus.
- Frequency range: 25 MHz - 1.75 GHz .
- 8-bit, 2.8 MSPS, quadrature sampling.
- Software support for Windows, Linux, GNURadio, Matlab and Simulink.

Figure 3.4 R820T2 USB dongle.[1]

In order to provide a more detailed description, a functional block diagram of the RTL-SDR is provided in Figure 3.6. As shown in the diagram, the RTL-SDR contains the RF front-end and the IF stage required for receiving signal. The configuration parameters (i.e. sampling rate, centre frequency and gain) are transmitted from the laptop and received by the RTL2832U coded orthogonal division multiplexed (COFDM) demodulator. The RF signal is received by the antenna that is connected to the Rafael Micro R820T silicon tuner. This IC contains the RF and IF stages (image rejection filter, mixer, LPF and amplifiers) and its output is the IF signal. Then, the IF signal enters to the RTL2832U demodulator. In this device the signal passes through an anti-aliasing filter and it is digitised by an ADC, then the signal is down-converted from IF to base band (BB). Finally, the BB signal is decimated and sent to the laptop in I/Q format (complex numbers) through the USB port.

It is important to clarify that the carrier used in the RF mixer of the R820T is generated by a voltage controlled oscillator (VCO) that is controlled with a fractional PLL. On the other hand, the carriers used in the quadrature sampling in the RTL2832U are synchronised by using a numeric controlled oscillator (NCO) controlled with a PLL.

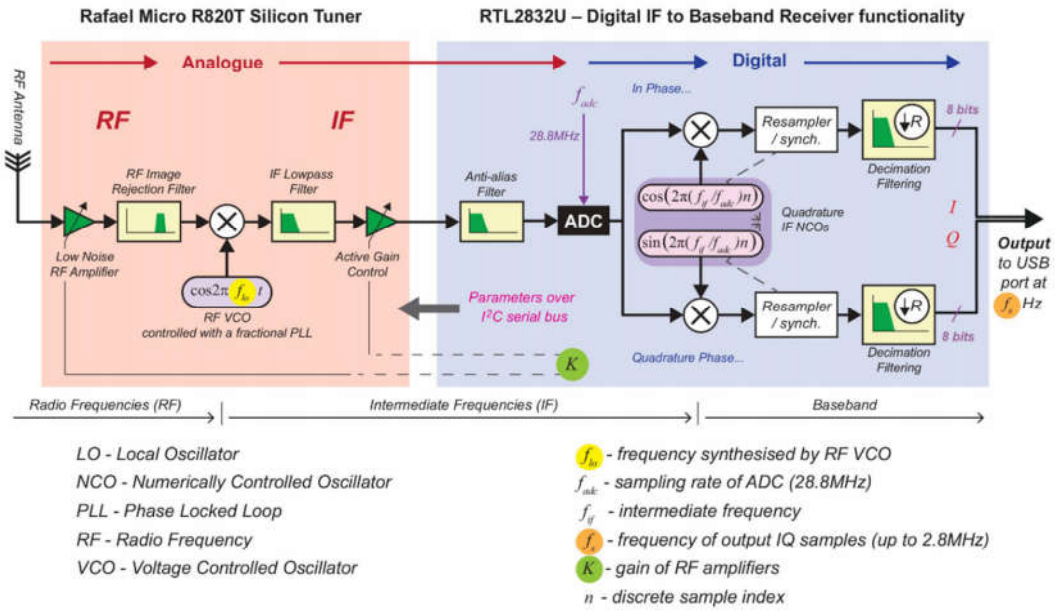

Figure 3.5 Block diagram of the R820T RTL-SDR hardware architecture.[1]

3.1.3 Receiving device calibration

As mentioned in section 2.5, the frequency deviation is an important index in a communication system since it affects the accuracy of the tuner. Moreover, standard software such as Matlab and GNURadio use it as a correction parameter. This is particularly critical in digital modulation (it is explained in detail in section 2.8).

Unlike the VCTCXO in the bladeRF, which has a small and stable frequency deviation (in the order of bpm), the VCO used in RTL-SDR has a considerable frequency deviation that is temperature dependant. This implies that the RTL-SDR has to receive random signals for a period of time before real reception starts, so that the device “warms up” until the frequency deviation of the VCO becomes stable.

In order to determine the frequency deviation in the RTL-SDR, a reference signal is necessary. The reference signal should have its maximum power component at a known frequency, which will be considered the nominal frequency (or reference frequency). Then, the tuner in the RTL-SDR is set to receive signal with a centre frequency equal to the reference frequency. Finally, the received signal is analysed in frequency domain and the value of the frequency with the highest power component is compared to the reference frequency. The difference between these two frequencies provides the frequency deviation. Since the result may vary slightly depending on the environment conditions, this process is repeated several times in order to calculate the average value of the frequency deviation.

There are several applications that can be used to determine the frequency deviation of the RTL-SDR. For example, in [1] the author provides two Simulink programs that can be used in conjunction with the RTL-SDR and a transmitting device to determine the frequency deviation in the former. In this program the transmitting device sends a sine wave centred at a known frequency, and the RTL-SDR receives and analyses the signal in frequency domain in order to detect the peak corresponding to the sine wave and calculate the actual centre frequency.

The main disadvantage of the method mentioned above is the fact that a reliable (in terms of accuracy) transmitter device is required. An alternative approach is the use of a signal generated by an external agent. This procedure can be performed by using the software Kalibrate, which is supported by all Linux distributions.

The calibration of the RTL-SDR used in the present project was performed by using both methods (the Simulink program and the Kalibrate software) in order to compare the results obtained with each method. The result in the Simulink program was a frequency deviation equal to 30 ppm , while the value of the frequency deviation obtained in Kalibrate was equal to 36 ppm.

3.1.4 Transmitting device calibration

Even though the bladeRF counts with a very accurate VCTCXO which is factory calibrated at +/-26 ppb, it is necessary to calibrate the device periodically since the frequency deviation of an oscillator increases through time.

The calibration of the transmitting device was performed by using the software Kalibrate, and the result obtained was a frequency deviation equal to +/-27 ppb.

3.2 Software description

This section describes the software used for the calibration of the RTL-SDR and the blade RF (Kalibrate) and the software used for the implementation of the functional blocks in the SDR systems designed in the project.

3.2.1 Kalibrate

Kalibrate is an open source software that is used to calculate the frequency deviation in the tuner of an SDR receiver device. This software scans for a global system for mobile (GSM) base station in a given frequency band and extract the reference signal from it to calculate the frequency deviation in the receiver [www03].

Since the International Telecommunication Union (ITU) allocates the radio spectrum around the world, the software provides different options to scan for a GSM base station. The options provided by Kalibrate are described in Table 3.1. The GSM-900 and E-GSM bands are available in the United Kingdom and for that reason they were used for the calibration of the receiving and transmitting devices in the present work.

Table 3.1 GSM Bands available for scanning in Kalibrate.

GSM Band	Frequency (MHz)	Uplink (MHz)	Downlink (MHz)	Channel number
GSM-850	850	824.2 – 849.2	869.2 – 893.8	128-251
GSM-R	900	876.0 – 915.0	921.0 – 960.0	955 - 1023 0 - 124

GSM-900	900	890.0 – 915.0	935.0 – 960.0	1 - 124
E-GSM	900	880.0 – 915.0	925.0 – 960.0	975 – 1023 0 - 124
DCS	1800	1710.2 – 1784.8	1805.2 – 1879.8	512 – 885
PCS	1900	1850.2 – 1909.8	1930.2 – 1989.8	512 - 810

Kalibrate is widely used due to its flexibility. On one hand, it allows the user to calibrate a receiver without acquiring a transmitter. On the other hand, it counts with various versions that are compatible with most commercial devices (e.g. Kalibrate-RTL and Kalibrate-bladeRF) and it is supported by all Linux distributions.

3.2.2 GNURadio

GNU Radio is an open software development toolkit that provides the processing functions and runtime required to build and run SDR systems. The applications created with GNURadio are called ‘flowgraphs’, and they are formed by a series of functional blocks connected togheter. The way these blocks are connected defines the dataflow of the application.

Two different programming languages are used to implement an application in GNURado, depending on the complexity of the operations to be performed. On one hand, the dataflow and interaction among the blocks are implemented by using the Python programming language. On the other hand, the functional blocks that are expected to perform time-critical signal processing are usually implemented by using the C++ programming language. Therefore, it is possible to develope real-time SDR applications by using standar programming languages [www04, www05].

Additionally, GNURadio can operate on various Linux-based embedded operating systems , such as Raspbian or Angstrom. This represents a major advantage in the development of embedded systems, since the software can operate on a single board computer (SBC) such as the Beaglebone or the Raspberry Pi. For this reason many developers, students and hobbysts use a transceiver device and an SBC with GNURadio to perform rapid prototyping (see Figure 3.6).

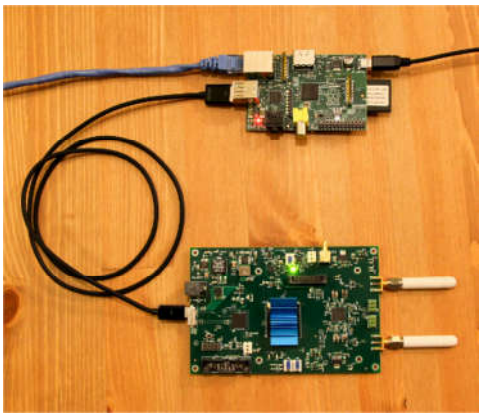

Figure 3.6 bladeRF transceiver connected to a Raspberry Pi running GNURadio.[www02]

3.2.3 GNURadio Companion

GNURadio Companion, usually abreviated as GRC, is a graphical user interface (GUI) used to develop applications with GNURadio. In this GUI the user places functional blocks, configures them and connects them to create a system (similar to Mathworks Simulink). Then, the resulting block diagram is built to generate a 'flowchart' implemented in Python. In this sense GRC is, basically, a Python code generator. [www04,www05]

GRC provides a wide variety of blocks that are classified in different groups. Every block counts has inputs and outputs that can present different format depending on the block (float, byte, short, integer, complex). The type of format can be identified by the color of the input/output port in the block. For example, orange color represents float values and light blue color represents complex values.

The most important blocks are mathematical operators, packet operators source blocks, sink blocks, hardware blocks, signal processing blocks and GUI blocks.

On one hand, mathematical operators are applied to a signal sample by sample. These operators include absolute value, add, multiply, divide and integrate. On the other hand, packet operators are applied to groups of samples (streams). These blocks include convert packed to unpacked, convert unpacked to packed, packed header generator, among others. Packet operators are important for the generation of packets that contain access code and checksum.

A source block is used to generate a signal from a random pattern, a standard pattern (e.g. sine, square, triangle), a specific vector or a file. Since the parameters for the generation of the signal are specified inside the block, no inputs are required for this block. For this reason a source block does not have input ports (see Figure 3.7).

Figure 3.7 Source blocks used in GNURadio Companion.

A sink block is used to store, display or transmit a signal to a specific device. Since the data that enters to this block flows to a file, a chart or a specific device, no outputs are required for this block. For this reason a sink block does not have output ports (see Figure 3.8).

Figure 3.8 Sink blocks used in GNURadio Companion.

Hardware blocks are particular cases of sources and sinks. These blocks interact with the peripherals incorporated in the system where GNURadio is running on, or with external components connected to the system through the ethernet or the USB port. For example, the microphone and the speaker of a laptop are represented in the block diagram as an audio source and an audio sink, respectively. Similarly, the SDR receiving device is represented in the block diagram as an RTL-SDR source.

A particular characteristic of the blocks that represent hardware for SDR applications (transmitting and receiving devices) is the fact that the input and output values of these blocks are represented by complex numbers (see Figure 3.9). In GNURadio this format is used to represent signals in I/Q format. The reason for this is that all the commercial devices supported by GNURadio perform a quadrature modulation/demodulation on the signal that comes to/from the computer that runs the application. This can be appreciated in the descriptions provided in sections 3.1.1 and 3.1.2.

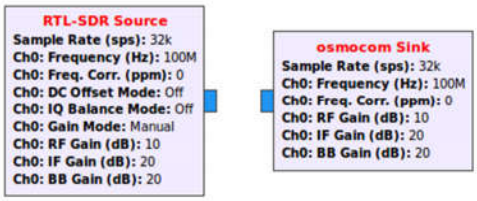

Figure 3.9 Blocks used to represent the RTL-SDR and the bladeRF in GRC.

Signal processing blocks perform digital signal processing techniques on the input signal. These blocks include include filters, encoders, decoders, modulators, demodulators, resamplers, among others. Modulators and demodulators are processing blocks characterised by having either the input or the output represented by complex numbers. The reason is that, as mentioned in section 2.1, the demodulator block in a receiver goes immediately after the ADC. In GNURadio this implies that the SDR receiver block is expected to be connected to the demodulator (see Figure) and, therefore, the output of the former and the input of the latter should have the same format (I/Q). Similarly, a modulator block is expected to be connected to a SDR transmitter block and, therefore, the output of the demodulator is represented in I/Q format.

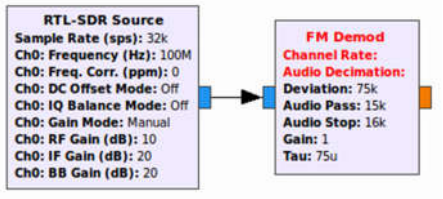

Figure 3.10 RTL-SDR connected to a FM demodulator block.

GUI blocks are a special case of sink blocks used to plot graphs and charts of the signals in the system. There are various types of GUI blocks that allow the user to plot the signal in time domain, plot the signal in frequency domain, and even plot a constellation graph. In this sense, GUI blocks are equivalent to the scope blocks in Mathworks Simulink.

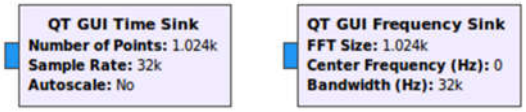

Figure 3.11 QT GUI blocks used in GNURadio Companion.

Even though the blocks provided by default are enough to implement most basic applications, GRC allows the user to create new blocks. These blocks, also known as out of the tree (OOT) blocks, can be implemented by using Python or C++.

3.3 Transmitter block parameters setting

As mentioned in the previous section, the SDR hardware blocks are special cases of source and sink blocks and, as in any other block, it is necessary to configure their parameters in order to get a correct performance.

The technical information provided in section 3.1 will be particularly useful to set the parameters of the blocks corresponding to the SDR hardware used in the project.

The transmitting device (bladeRF) is represented in GNURadio with the 'osmocom Sink' block. The parameters to be configured in this block are shown in Figure 3.12.

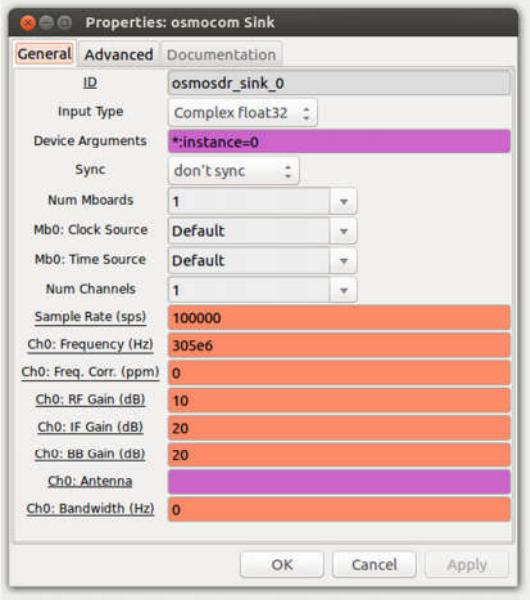

Figure 3.12 Parameters of the osmocom Sink block.

The ID represents the name used to identify the block in the program. Its value is automatically created and it is not necessary to modify it.

The input type indicates the format of the input values. As mentioned in the previous section, the input values are represented by complex values because the bladeRF expects to receive data in I/Q format.

The parameters Sync, Num Mboards, Clock Source, Time Source and Num Channels are useless in this device model and for that reason it is not necessary to change their values.

The sample rate is expressed in samples per second (SPS) and it is restricted to some values. On one hand, the hardware limits the sampling rate of the device to 40 MSPS maximum. On the other hand, the software limits the sampling rate to 100 Kilo samples per second (KSPS). The sampling rate of the device should be selected according to the applicaton.

The frequency parameter refers to the center frequency of the signal after it is up-converted by the transmitter mixer. On one hand, the bladeRF can transmit signal in the range from 300 MHz to 3.8 GHz. On the other hand, the RTL-SDR can receive signal in the range from 25 MHz to 1.75 GHz. Therefore, in order to set a communication between the bladeRF and the RTL-SDR it is necessary to set a frequency in the range form 300 MHz to 7.75 GHz. For the applications implemented in the present work, the frequency will be set to 305 MHz.

The Frequency Correction parameter refers to the frequency deviation of oscillator used in the mixer, expressed in ppm. According to the calibration performed, this value is equal to 0.027 ppm.

RF Gain, IF Gain and BB Gain represent the gain in the different stages of the transmitting device (Radio frequency, intermediate frequency and baseband). However, the bladeRF does not count with a programmable amplifier in the baseband stage, so the BB Gain parameter is meaningless in this case. Both RF Gain and IF Gain will be set to 20 dB for the present work.

The Antenna parameter is left empty, since the bladeRF counts with only one antena which is selected by default.

Finally, the value of the bandwidth is left in 0 since the bladeRF does not count with any programmable filter that limit the bandwidth of the transmitted signal.

3.4 Receiver block parameters setting

The receiving device (RTL-SDR) is represented in GNURadio with the 'RTL-SDR Source' block. The parameters to be configured in this block are shown in Figure 3.13.

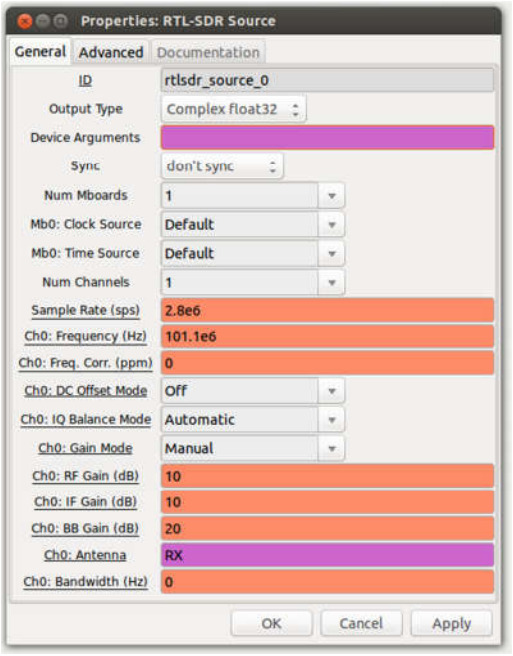

Figure 3.13 Parameters of the RTL-SDR Source block.

The ID represents the name used to identify the block in the program. Its value is automatically created and it is not necessary to modify it.

The output type indicates the format of the output values. As mentioned in previous sections, the input values are represented by complex values because the RTL-SDR provides data in I/Q format.

The parameters Sync, Num Mboards, Clock Source, Time Source and Num Channels are useless in this device model and for that reason it is not necessary to change their values.

The sample rate is expressed in samples per second (SPS) and it is restricted to some values. On one hand, the hardware limits the sampling rate of the device to 2.8 MSPS maximum. On the other hand, the software limits the sampling rate to 100 Kilo samples per second (KSPS). The sampling rate of the device should be selected according to the applicaton.

The frequency parameter refers to the center frequency of the signal before it passes through the RF mixer. As in the case of the bladeRF, for the applications implemented in the present work, the frequency will be set to 305 MHz.

The Frequency Correction parameter refers to the frequency deviation of oscillator used in the mixer, expressed in ppm. According to the calibration performed, this value is equal to 37 ppm.

RF Gain, IF Gain and BB Gain represent the gain in the different stages of the transmitting device (Radio frequency, intermediate frequency and baseband). However, the RTL-SDR does not count with a programmable amplifier in the baseband stage, so the BB Gain parameter is meaningless in this case. Both RF Gain and IF Gain will be set to 20 dB for the present work.

The Antenna parameter is left empty, since the RTL-SDR counts with only one antena which is selected by default.

Finally, the value of the bandwidth is left in 0 since the RTL-SDR does not count with any programmable filter that limit the bandwidth of the transmitted signal.

Chapter 4
SDR Systems Design

This section describes the design and implementation of the block diagrams of four SDR systems that are able to transmit and receive FM and QPSK modulated signals. In all the designs the parameters of the bladeRF and the RTL-SDR are set as indicated in sections 3.3 and 3.4.

Since the the systems will be tested by performing an actual transmission, some practical issues described in Chapter 2 (such as emphasis and synchronisation) will be considered in the designs.

4.1 Frequency modulation (FM)

Two types of FM modulation can be performed: NBFM and WBFM. In order to test the implemented system with radio broadcast signals the WBFM modulation will be used.

4.1.1 WBFM Transmitter

The components required for the design of the WBFM transmitter depend, basically, on the characteristics of the source and the limitations of the hardware used for the transmission.

A first basic design for testing the WBFM modulator block will be implemented. The first design consists on a system that takes a 100Hz sine wave, modulates it in WBFM and transmits the resulting signal.

In order to perform a reliable communication four elements are required in the first design: A sine wave, a WBFM modulator, a pre-emphasiser, and an SDR transmitting device. The signal flow is as show in Figure 4.1.

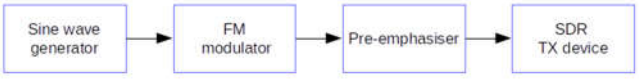

Figure 4.1 Basic components required for the WBFM transmitter.

Since the WBFM Transmit block in GRC includes both a WBFM modulator and an emphasiser, only three blocks are required for this design. However, two additional blocks will be used in order to visualize the original signal and part of the modulated signal (see Figure 4.2) .

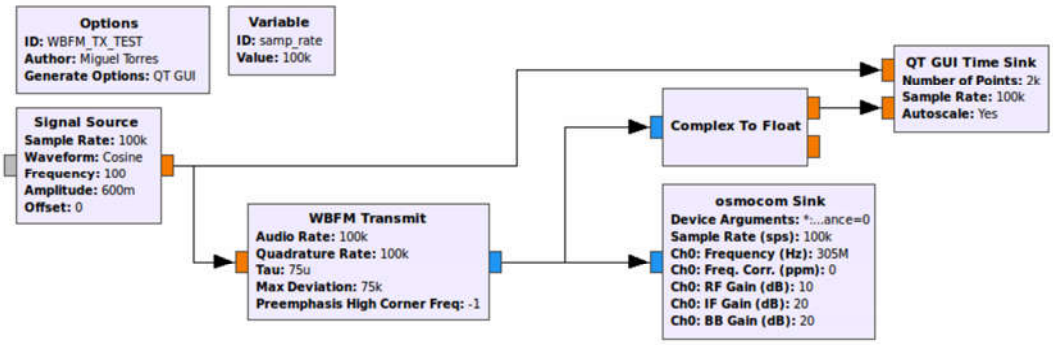

Figure 4.2 WBFM transmitter designed in GNURadio Companion.

There is no need to use multi-rate sampling in this design, in consequence the sampling rate is the same in the whole system, and its value is stored in the variable “samp_rate”.

The sine wave is generated by using a signal source block and setting the parameters as shown in Figure 4.3.

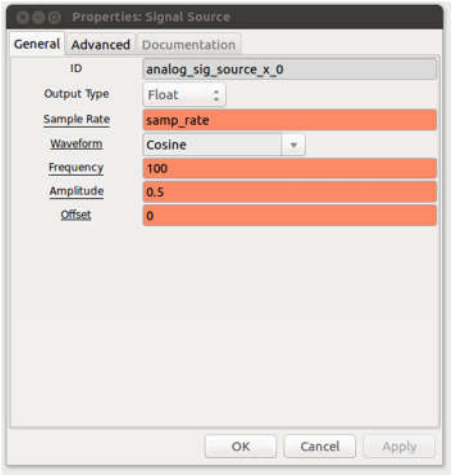

Figure 4.3 Parameters of the Signal Source block.

The sine wave comes from the signal source to a WBFM transmit block (see Figure 4.4). This block includes three sub-blocks inside: FM modulator, pre-emphasiser, and quadrature modulator. The parameter Audio Rate indicates the rate used to sample the input signal (the sine wave); Quadrature Rate indicates the rate used to sample the I/Q signal at the output of

the WBFM modulator; Tau indicates the time constant of the emphasiser (considering it behaves as an RC filter); Max Deviation indicates the frequency factor of FM modulation, which is 75 KHz for standard WBFM .

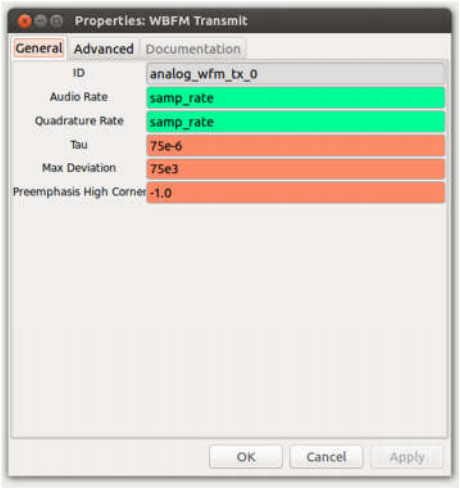

Figure 4.4 Parameters of the WBFM transmit block.

Then, the modulated signal comes from the modulator output to the block that represents he transmitting device (osmocom Sink). As in the other blocks, the sampling rate is set to the value of the variable “samp_rate”.

Additionally, both the original signal and the modulated signal are connected to a time scope (QT GUI Time Sink), which will plot these signals. However, since the WBFM modulator generates complex values (I/Q format) it is necessary to adapt its output values (from complex to real) in order to make them compatible with the format of the inputs in the time scope. For this reason, only the Q component of the signal is sent to the QT GUI Time Sink .

Finally, the main criteria for the selection of the sampling rate in this design is meeting the Nyquist theorem when sampling both the original signal and the modulated signal. Even though a 1 KHz sampling rate would be enough for the original signal, a higher sampling rate is required to detect the details of the modulated signal. As a thumb rule, the sampling rate should be at least 50 times the highest frequency component of the original signal. In order to have a detailed view of the modulated signal, the value of the sampling rate is set to 100 KHz. The original signal and modulated signal obtained are shown in Figure 4.5.

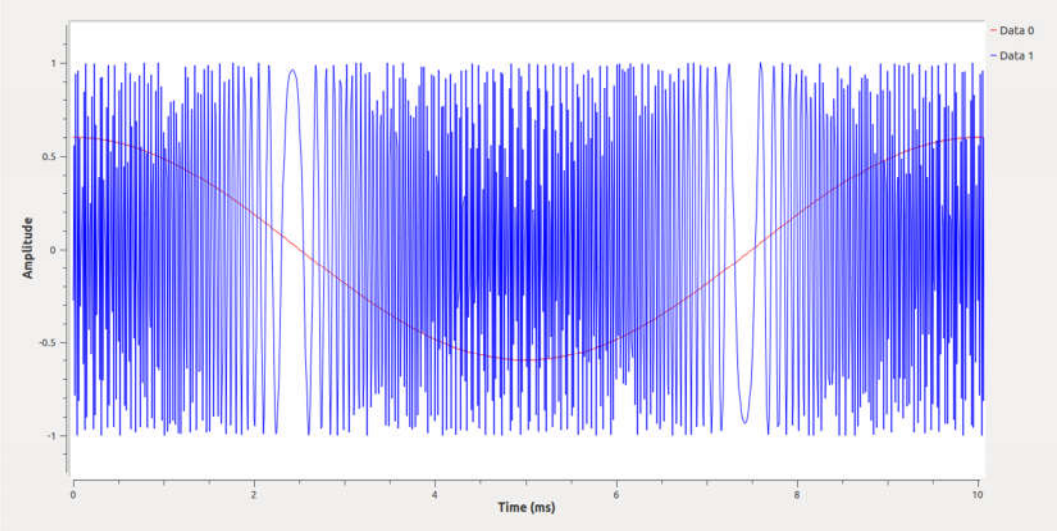

Figure 4.5 Original signal and modulated signal plotted on a QT GUI TIme Sink block.

A more realistic design consists on a system that takes an audio signal sampled at 48 KHz, modulates it in WBFM and transmits the resulting signal. In this case seven blocks are required and, additionaly a time scope (QT GUI Time Sink) will be included to visualize the modulated signal (see Figure 4.6) .

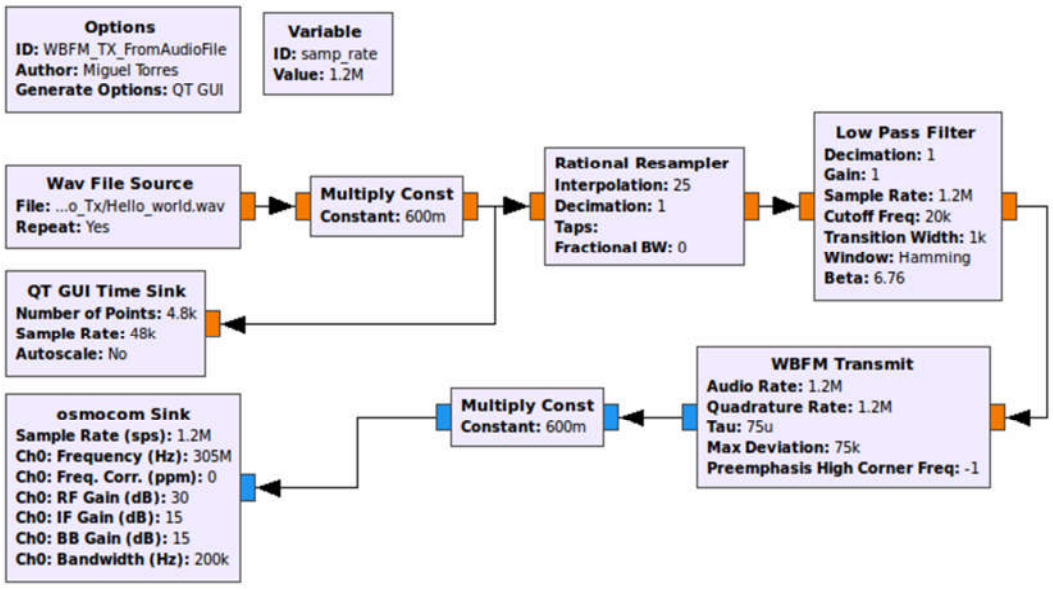

Figure 4.6 WBFM audio transmitter designed in GNURadio Companion.

Unlike the previous design, in this case it is necessary the use of multi-rate sampling because the audio signal is sampled at 48 KHz and the transmitting device does not support sampling rates lower than 100 KHz.

The audio signal is taken from an audio file (specifically a WAV file) and it is “amplified” by a gain lower than 1. This amplification is included to reduce the frequency components of the modulated signal. The resulting waveform is displayed on a time scope (see Figure 4.7).

Then, the audio signal is interpolated in order to get a sampling frequency suitable for the modulated signal. Considering the thumb rule mentioned before (in the previous design) and the fact that the audible spectrum covers from 20 Hz to 20 KHz, the decimation factor is set to 25. Therefore, the new sampling frequency is equal to 1.2 MHz (25 times 48 KHz) which is suitable for sampling the modulated signal.

As mentioned in section 2.6.1, a LPF should be applied to the signal after it is resampled. The parameters of the LPF are set so that only the frequency components that do not belong to the audible spectrum are supressed.

After passing through the filter the signal enters to the WBFM modulator. Then, a gain is applied to the modulated signal in order to control its amplitude to avoid the saturation of the transmitter. Finally, the resulting signal is sent to the transmitting device.

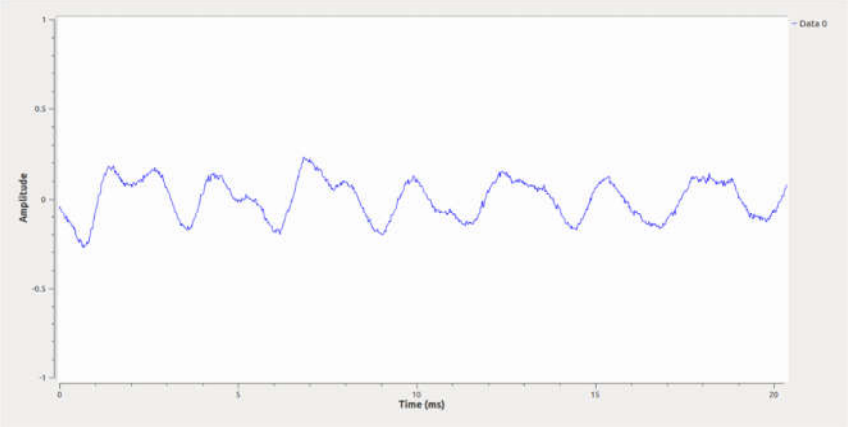

Figure 4.7 Waveform of the audio signal transmitted.

4.1.2 WBFM Receiver

In this section a receiver for FM radio broadcasting will be implemented. This receiver should be able to reproduce audio signals transmitted by both commercial radio stations and the transmitter implemented in the previous section. The basic design of this system requires six blocks as shown in Figure 4.8.

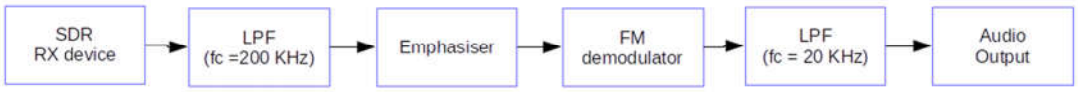

Figure 4.8 Basic components required for the WBFM audio receiver.

The design in GRC requires a minimum of six blocks to work. However, two additional blocks will be added in order to plot the spectrum of the received signal and the waveform of the audio signal. The resulting design is shown in Figure 4.9.

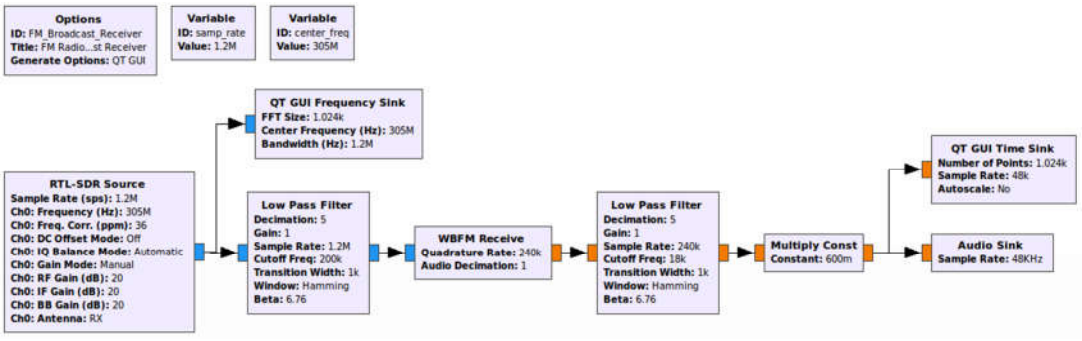

Figure 4.9 WBFM audio receiver designed in GNURadio Companion.

It is important to consider the fact that the Audio Sink block represents hardware that is limited to some standard samplig rates (e.g. 32 KSPS, 48 KSPS) and, for this reason, it is necessary to perform decimations in the blocks. In order to keep the calculations simple, the sampling rate of the RTL-SDR will be a multiple of the sampling rate in the Audio Sink block.

The signal is sampled at 1.2 MSPS by the RTL-SDR and it is connected to a QT GUI Sink to visualise its power in frequency domain (see Figure 4.10). Then the signal passes through a LPF in order to limit ists bandwidth. The parameters of the LPF are set to provide a cut-off frequency of 200 KHz (standard bandwidth for radio broadcasting stations) and a transition

width of 1KHz, in order to reduce the number of coefficients required in its implementation. Additionally, the decimator incorporated in the filter is set to produce a decimation by 5. This means that the sampling rate for the next block should be set to 240 KSPS.

After the signal is filtered, it passes through the WBFM Receive block, which demodulates the I/Q components to get a real FM modulated, passes this signal through a de-emphasiser, and then applies FM demodulation to this signal. Then, a gain lower than 1 is applied to the signal in order to avoid the saturation of the audio output. The resulting waveform is expected to be identical to the original signal at the transmitter.

An additional LPF is added to eliminate the frequency components outside the audible spectrum (cut-off frequency equal to 20 KHz) and to decimate the sampling rate by 5. Finally, the audio signal is sampled at 48 KSPS by the Audio Sink block and it is plotted in time domain with a QT GUI Sink (see Figure 4.10).

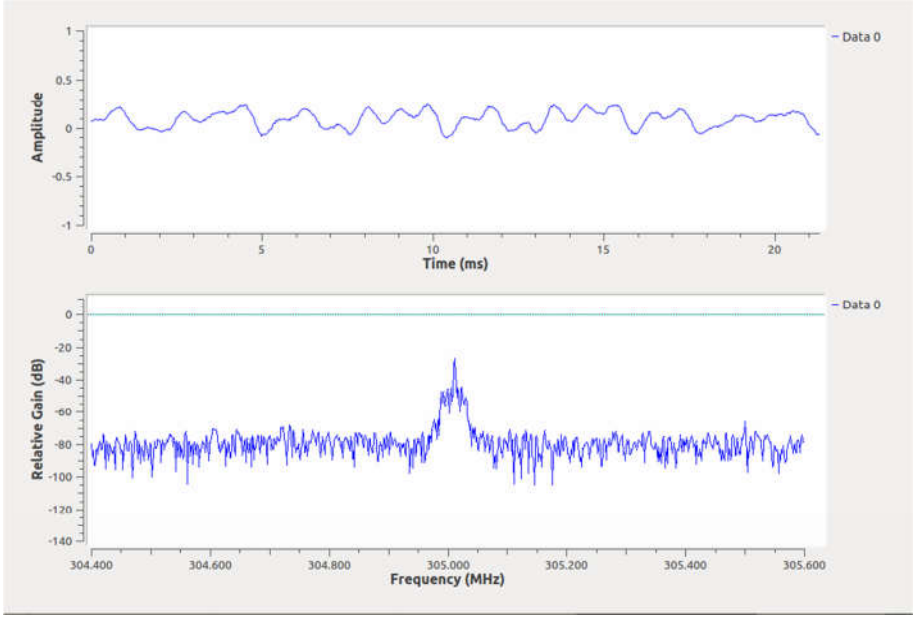

Figure 4.10 Spectrum of the received signal (bottom) and waveform of the audio signal (top).

4.2 Quadrature phase shift keying (QPSK)

This section describes the implementation of a QPSK transmitter and a QPSK receiver in the software GNURadio Companion. Unlike the previous designs, these systems work with digital signal and the concept of symbols. Also, synchronisation becomes a critical issue in these systems.

4.2.1 QPSK Transmitter

The task in this section is the design of a system that takes the bit sequence '0000111100110101', modulates it in QPSK and transmits it.

The basic elements required to implement this systems are: A signal source (that provides digital signal), a QPSK modulator, a pulse-shaping filter (such as an RRC filter)[3] and an SDR transmitter device. The data flows through these elements as shown in Figure 4.11.

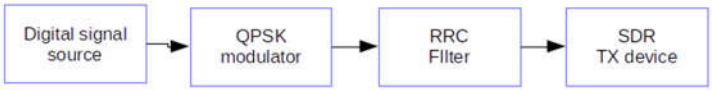

Figure 4.11 Basic elements of a QPSK transmitter.

The design in GRC is shown in Figure 4.12. The block corresponding to the RRC filter does not appear explicitly because it is included in the Constellation Modulator block. The vector source generates the values 15 (00001111) and 53 (00110101) indefinitely and send them to the modulator. The modulator takes two bits and generate a symbol according to the information provided by the Constellation Rect. Object. Finally, the modulated signal is transmitted through the SDR hardware and plotted in tme and frequency domain. As in the case of the WBFM modulator, the output of the Constellation Modulator is in I/Q format (complex numbers).

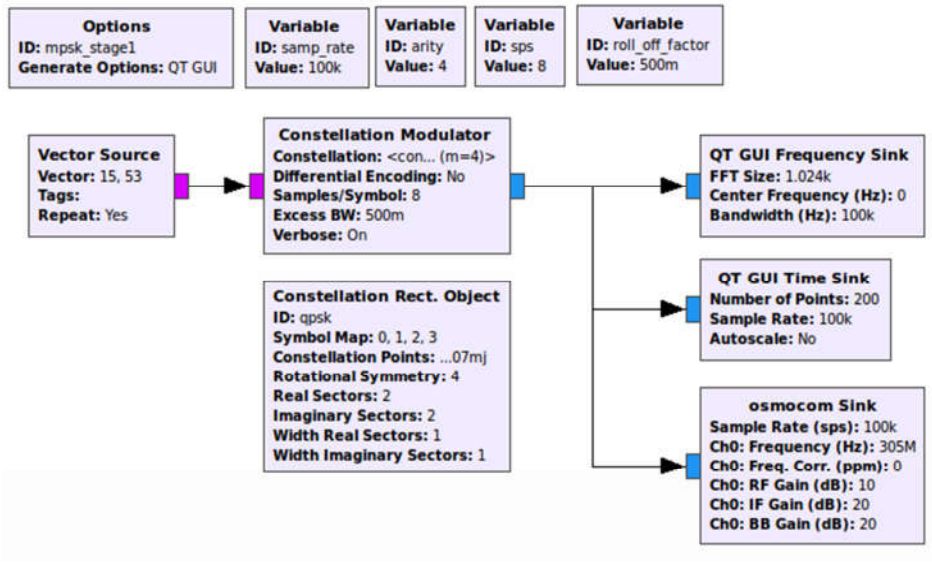

Figure 4.12 QPSK transmitter designed in GNURadio Companion.

The Constellation Rect. Object contains the main information of the modulator. The symbol map, in this case, presents the possible 2-bit values that can be received by the modulator (00, 01, 10, 11) . The constellation points define the complex values corresponding to the 2-bit values indicated in the symbol map (see Figure 4.13).

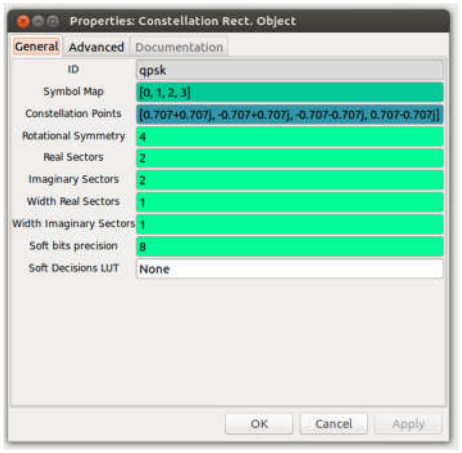

Figure 4.13 Parameters of the Constellation Rect. Object block.

As shown in Figure 4.14, the constellation modulator block generates the symbols defined by the Constellation Rect. Object block. The parameter constellation identifies the constellation object that will be used. The number of samples per symbol is defined by the variable sps (equal to 8). The excess bandwidth define the roll-off factor of the RRC filter included in the modulator.

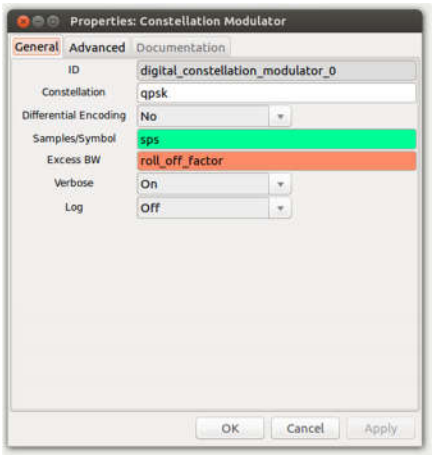

Figure 4.14 Parameters of the Constellation Modulator block.

At the end of the processing an I/Q signal limited in bandwidth (due to the RRC filter) is expected. Figure 4.15 shows this signal plotted in time and frequency domain. The blue curve in the time domain graph represents the I component (real) of the signal and the red curve represents the Q component (imaginary).

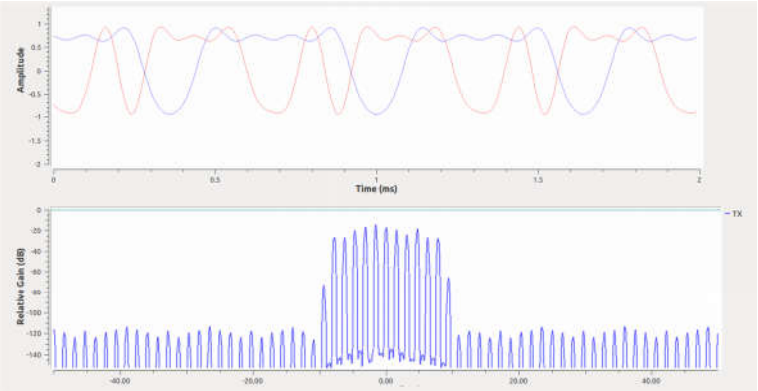

Figure 4.15 Complex QPSK modulated signal plotted in time and frequency domain.

4.2.2 QPSK Synchronisation

The previous design will be modified to simulate the effects of timing synchronisation at a QPSK receiver. On one hand, the SDR transmitter and the time and frequency GUIs will be removed. On the hand, one Clock Sync block and two GUI Constellation Sink blocks will be added to display the constellations and perform the time synchronisation of the signal, respectively. The resulting design is shown in Figure 4.16.

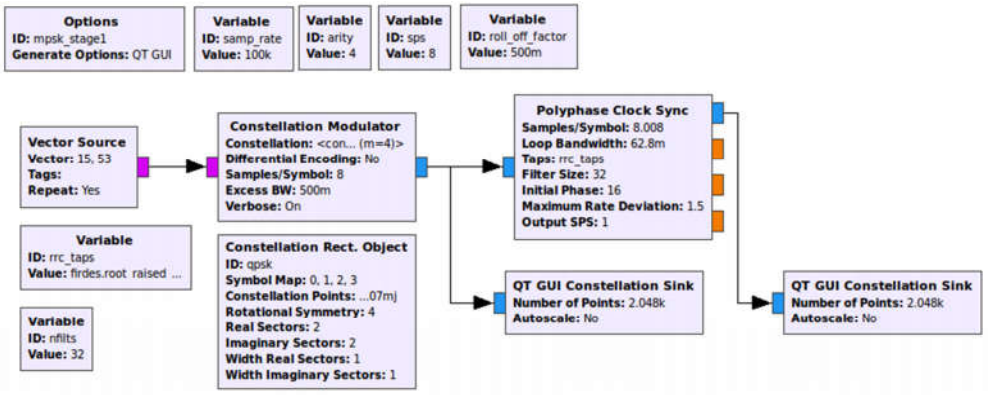

Figure 4.16 System to simulate the time synchronisation effects in QPSK communication.

The Polyphase Clock Sync block is, basically, a bank of filters shifted in time working together [www05]. In this case, a bank of 32 root raised cosine filters has been implemented (see Figure 4.17).

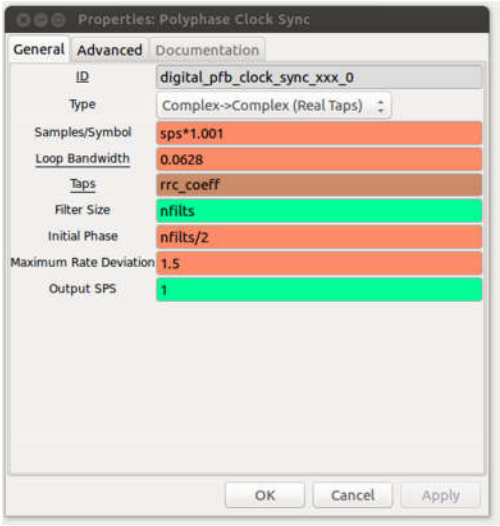

Figure 4.17 Parameter of the Polyphase Clock Sync block.

The symbols detected before and after the Clock Sync block are plotted on the QT GUI Constellation sinks and the resulting graphics are shown in Figure 4.18. The data before the synchronisation is represented by the graph on the left (Rx Data) and the data after the synchronisation is represented by the graph on the right (Synch Data).

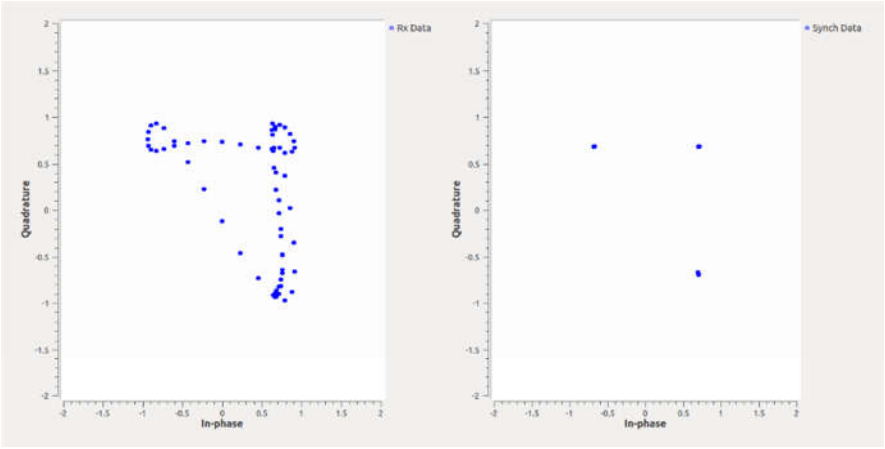

Figure 4.18 Results of the simulation: Received Data vs Synchronised data.

The graph corresponding to the signal after the synchronisation presents the expected values. The vector source block transmits the sequences 00001111 and 00110101 indefinetly. Then the constellation modulator takes the bits in groups of two, so the input values considered by the modulator are: 00, 00, 11, 11, 00, 11, 01, 01. This means that all

symbols except the one corresponding to the value 10 (-0.707 - j0.707) are generated by the modulator.

Chapter 5
Results and Conclusion

- The accuracy of the oscillators used for generating the carriers in the transmitter and the receiver affect directly the carrier synchronisation in a communication system.
- It is considered a good practice to emphasize FM signals before transmitting them in order to reduce the noise.
- The sampling frequency in audio applications is restricted by the harware.
- Frequency decimation is necessary to reduce computational costs.
- The design and implementation of a QPSK receiver present more complexity than in the case of a transmitter. This is due to the synchronisation issues that must be solved at the receiver.
- The inclusion of matching filters at the transmitter and the receiver is required to reduce the inter symbol interference in the signal.
- The implementation of a QPSK receiver was not possible in the present work, due to problems in the symbol synchronisation.
- The design of SDR applications with the GNURadio Companion software is very intuitive and can be used to implement embedded apllications.

References

[1] B. Stewart, K.Barlee, D. Atkinson and L. Crockett *Software Defined Radio using Matlab & Simulink and the RTL-SDR*. Glasgow: University of Strathclyde Engineering, 2012.

[2] J. Proakis, D.Manolakis *Digital Signal Processing: Principles, Algorithms and Applications* . Pearson Education Limited, 2014.

[3] B. Sklar *Digital Communications: Fundamentals and Applications*. Second edition, Prentice Hall, 2001.

[www01] SDR Forum web site. Accessed 23/08/2016
http://www.sdrforum.org/

[www02] Nuand bladeRF-x40 web page. Accessed 23/08/2016
https://www.nuand.com/blog/product/bladerf-x40/

[www03] GitHub – Kalibrate RTL web page. Accessed 23/08/2016
https://github.com/steve-m/kalibrate-rtl

[www04] GNURadio web site. Accessed 23/08/2016
http://gnuradio.org/

[www05] GNURadio Tutorials web site. Accessed 23/08/2016
http://gnuradio.org/redmine/projects/gnuradio/wiki/Tutorials