Università di Pisa

Facoltà di Ingegneria

CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA INFORMATICA

Tesi di Laurea

Design and implementation of a software-defined VHF-DSC multichannel monitoring system

Candidato

Luca Melette

Relatori

Prof. Marco Luise

Prof. Luciano Lenzini

Anno Accademico 2007-2008

Sommario

Nell'ambito di questa tesi vengono proposti il progetto e l'implementazione di un sistema per il monitoraggio delle comunicazioni marittime (vocali e digitali) in banda VHF, utilizzando la tecnologia *Software Defined Radio*.
Partendo dal framework GNU Radio e la periferica di acquisizione (*Universal Software Radio Peripheral*) collegata ad esso, è possibile ricevere simultaneamente tutti i canali VHF con una sola interfaccia radio. A partire dalla porzione di spettro acquisito, tutte le successive elaborazioni del segnale sono effettuate via software da un applicativo server, che effettua l'analisi e demodulazione della porzione di banda acquisita, e da un applicativo client che permette all'utente la selezione e l'ascolto di uno dei canali voce.
La comunicazione tra client e server avviene tramite IP (UDP/Multicast), consentendo la distribuzione simultanea dei canali voce a più client.
È anche presente un applicativo dedicato alla decodifica dei messaggi DSC (*Digital Selective Calling*), usati dal sistema GMDSS (*Global Maritime Distress and Safety System*) per trasportare informazioni di emergenza o di servizio.

Abstract

The focus of this thesis is on monitoring the VHF maritime communications, including both voice and data, using the *Software Defined Radio* technology.
Starting from the GNU Radio framework and the *Universal Software Radio Peripheral*, it is possible to simultaneously receive all VHF channels with just one radio interface.
The acquired radio spectrum is then manipulated via software by a server application, that analyzes and demodulates channels, and processed by a client application, that allows an operator to choose and listen one radio channel.
The client/server communication occurs by means of IP (specifically Multicast/UDP), allowing the simultaneous distribution of all channels.
Moreover, a dedicated application decodes the messages on the DSC (*Digital Selective Calling*) channel: these ones are used by the GMDSS (*Global Maritime Distress and Safety System*) to carry emergency or service information.

Contents

Introduction

In the radio communications domain, there is a new approach that is being employed in more and more applications for its value: the Software Defined Radio (SDR).
Starting from the military field, this technology is spreading to the industry market and will probably reach the consumer electronics market soon.

The main idea of this approach is quite simple: it consists of creating a radio device that is controlled by software from the lowest level, allowing every possible elaboration on the radio signal. This means that the radio frequency must be converted to a digital form as close as possible to the antenna, avoiding to manipulate the signal with other dedicated hardware.
All the work is performed by a programmable microprocessor and an analog to digital signal converter (ADC). Since the acquired bandwidth is limited, it is only possible to cover a part of the radio spectrum, therefore the ideal software radio architecture can be used only from low to high frequency communications (below GHz).
Fast and expensive ADCs are now reaching a sampling rate of 1 GS/s, but they are not easily usable cause the amount of data they produce is really high, especially for a general purpose processor.

To solve the problem of reaching higher frequencies, an RF frontend is usually employed between the antenna and the digital converter: after a band pass filter, a local oscillator shifts down to the intermediate frequency the original signal, then the analog converter can sample the interesting part of the spectrum with a lower sample rate. Sometimes, multiple intermediate frequency stages are used.

Obviously, ADCs cover the receiption process only, thus digital to analog converters (DACs) are needed for transmission. Problems to solve are comparable for both directions.

One of the advantages of a software defined device is the flexibility: it can be adapted to a large number of applications without changing the hardware, simply changing or selecting a particular function of the software. For instance, in a cellular communications system the over the air standards can evolve in the years to increase the user density or allow the deployment of new services. Using SDR base stations can save a lot of money, therefore it is a good opportunity for service providers. The same advantages are available for TV broadcasting providers.

The flexibility can be also viewed in a multi-standard environment: an SDR is capable of speaking different standards at different frequencies adapting itself according to the ones that are available (cognitive radio). A typical example is the SpeakEasy project born from the U.S. military research in the 90s that is capable of operating from 2 MHz to 2 GHz and can communicate, and be used as a relay, between ground, air and sea forces, including also satellite communications.

The majority of analog voice communication systems, from LF to UHF, use a channelized medium access policy (FDM). Channels are usually contiguous in frequency and have a standard spacing and bandwidth occupation.
Looking to the maritime communications, we can find a large variety of systems: at the beginning, after the radio invention, the radiotelegraph was used on the ships for medium and long distance communications; nowadays, the Morse code has been completely substituted by voice and digital communication systems operating both ship to ship and ship to ground.
A set of frequencies are globally allocated by IMO (International Maritime Organization) for these purposes: for instance, in the VHF range, 68 channels (around 160 MHz) have been defined with 25 kHz spacing.
A common VHF transceiver can operate to a single channel and can monitor and automatically switch to the emergency channel when some activity is found. Here comes the need of multiple receivers for important circumstances, for example on big ships.

The SDR capability of capturing a large portion of radio frequencies can be exploited to operate over multiple channels at the same time with just one radio interface.
In particular, this thesis shows in concrete terms what is possible to do with a Universal Software Radio Peripheral (USRP) and GNU Radio: the USRP is the cheapest, but very flexible, software radio device on the market (open hardware), and GNU Radio is an open framework, written in both C++ and Python, that controls the USRP.

First chapter will introduce you to the software radio technology and GNU Radio. It starts with radio basic and explain what reasons led to develop SDR.
Next, a description of maritime communications is needed to understand what are the characteristics of the transmissions to monitor. A common VHF transceiver, used for monitoring system test, is presented in this section.
These are the basics to understand the next chapter: the design and implementation of the monitoring system. This chapter, the main work of the thesis, describes in detail the system components and will make you understand every problem and solution encountered during the implementation.
Afterwards, the conclusions will summarize all the work done and will suggest future enhancements for the project.

Chapter 1

Software Defined Radio

1.1 Radio basics

Radio frequency (RF) refers to a particular portion of the electromagnetic spectrum, within the range of about 3 Hz to 300 GHz, in which electromagnetic waves can be generated by an alternating current fed to an antenna. Any device which works within the RF region is called a radio.
Since the radio spectrum is very wide, it is subdivided into frequency ranges, as shown in Table 1.1, which have specific properties and are used for different applications.

Radio frequency is one of the most popular mediums for telecommunications. Information in radio waves is transmitted by modulating a carrier signal of higher frequency (usually in the range of MHz or GHz) with a particular modulation.
Receivers extract information from these waves using the reverse operation of the modulation, the demodulation. Audio, video or data, in both analog and digital forms, can be used as a modulating signal. In the traditional radio approach, the modulation process is done by a chain of hardware components (oscillators, mixers, etc).

This approach has some limitations: the most important is the low flexibility to adapt to new services and standards.

Name	Symbol	Frequency (Hz)	Wavelength (m)	Applications
Extremely low frequency	ELF	$3 \cdot 10^0 \div 3 \cdot 10^1$	$10^7 \div 10^8$	Audible sound, submarine communications
Super low frequency	SLF	$3 \cdot 10^1 \div 3 \cdot 10^2$	$10^6 \div 10^7$	Audible sound, AC power lines
Ultra low frequency	ULF	$3 \cdot 10^2 \div 3 \cdot 10^3$	$10^5 \div 10^6$	Audible sound, communication with mines
Very low frequency	VLF	$3 \cdot 10^3 \div 3 \cdot 10^4$	$10^4 \div 10^5$	Audible sound, ultrasounds
Low frequency	LF	$3 \cdot 10^4 \div 3 \cdot 10^5$	$10^3 \div 10^4$	AM broadcasting, navigational beacons
Medium frequency	MF	$3 \cdot 10^5 \div 3 \cdot 10^6$	$10^2 \div 10^3$	AM broadcasting, navigational beacons, maritime communications, aviation
High frequency	HF	$3 \cdot 10^6 \div 3 \cdot 10^7$	$10^1 \div 10^2$	Shortwave, amateur radio, citizens' band radio, skywave propagation
Very high frequency	VHF	$3 \cdot 10^7 \div 3 \cdot 10^8$	$10^0 \div 10^1$	FM broadcasting, amateur radio, TV broadcasting, maritime communications, aviation
Ultra high frequency	UHF	$3 \cdot 10^8 \div 3 \cdot 10^9$	$10^{-1} \div 10^0$	TV broadcasting, amateur radio, mobile/cordless telephones, wireless networking, microwave ovens
Super high frequency	SHF	$3 \cdot 10^9 \div 3 \cdot 10^{10}$	$10^{-2} \div 10^{-1}$	Wireless networking satellite links, microwave links, satellite TV
Extremely high frequency	EHF	$3 \cdot 10^{10} \div 3 \cdot 10^{11}$	$10^{-3} \div 10^{-2}$	Microwave data links, radio astronomy, remote sensing, weapons systems, security scanning

Table 1.1: Radio frequency ranges

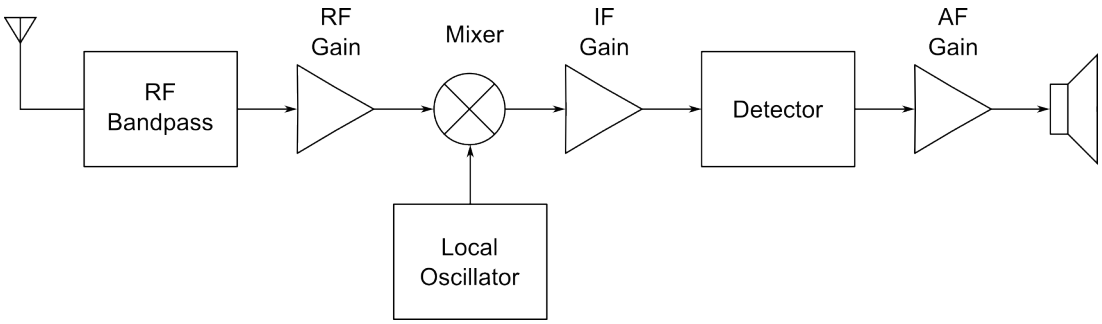


Figure 1.1: Block diagram of a traditional radio receiver

Each hardware element in radio chain (see Figure 1.1) performs a specific function. These components are designed to operate in a particular frequency band (RF) and standard. When frequency or any of the parameters of a standard changes, traditional radios cannot correctly decode the information. Before being able to operate under the new conditions, the system must be redesigned and hardware modules have to be replaced. Since redesigning, manufacturing and replacing hardware components imply higher costs and longer time to market, traditional radios suffer longer time to market and higher costs for the development and manufacturing of new products.

1.2 The software approach

Contrary to traditional radio technology, SDR follows an approach that could remove drawbacks of current radios and also add new values, for example the cognitive radio concept (not part of this thesis).

Software pieces, and not hardware components, treat the signals to extract the information. The idea behind software-defined radio is to do the modulation and demodulation functions with software instead of using dedicated circuitry.

The most obvious benefit is that instead of having to build extra circuitry to handle different types of radio signals, you can just load an appropriate software (or firmware).

Following this principle, a common computer can be turned to an AM radio receiver or a wireless data transceiver or a digital HDTV receiver or furthermore to everything men-

tioned before at the same moment. Because of this, software defined radios are not limited in the number of services they can provide.

This flexibility of software could be exploited to do things that are difficult, if not impossible, with traditional radio setups. Some of them will be illustrated in this thesis.

The method of extracting information in software requires the received signal to be transformed into the digital domain for processing (i.e. A/D Conversion), in contrast the transmit path does the opposite (i.e. D/A Conversion) while sending the signal to the antenna.

All the transceiving functions are performed by digital circuitry under software control. Software radios use digital signal processing (DSP) for filtering, modulation and demodulation of signals within the radio. While DSP can do everything an analog radio can do, it can also do functions that are difficult or impossible for analog radios. But more importantly, software radios are related to the computer software revolution that is affecting our society.

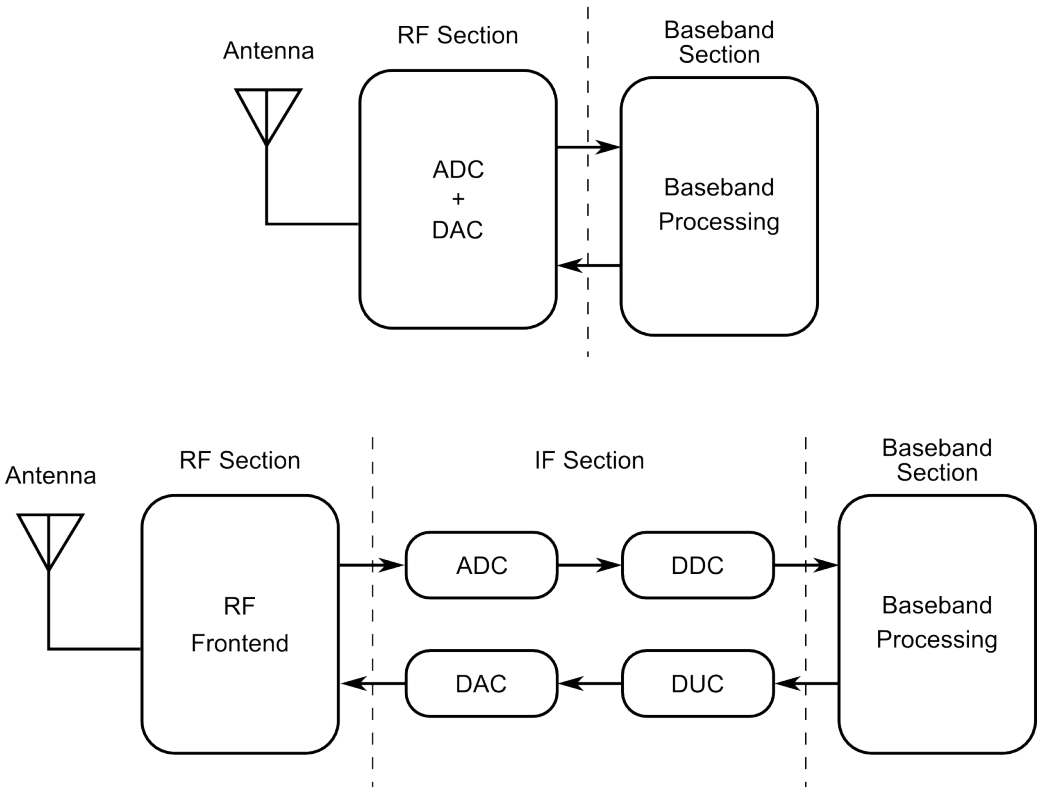


Figure 1.2: Ideal(top) against actual(bottom) SDR architecture comparison

1.3 SDR architectures

To understand how SDR works, it is necessary to recall some principles (from [9]).
Sampling or digitization is the process of converting an analog signal (a function of continuous time) into a numeric sequence (a function of discrete time).
This process is done in accordance with the Nyquist Criterion, which states:

> *Exact reconstruction of a continuous-time baseband signal from its samples is possible if the signal is bandlimited and the sampling frequency is greater than twice the signal bandwidth.*

Then the condition for exact reconstructability from samples at a uniform sampling rate in samples per unit time is: $Fs > 2B$
In some cases this condition is intentionally not respected to produce the aliasing effect, that normally is harmful.
Now we can analyze diagrams in Figure 1.2.

Traditional radios use no digitization or baseband digitization. In ideal SDR architecture, the digitization happens just after the antenna.
This is the best approach, since it allows processing the signal fully in software, but this kind of digitization is currently impossible to implement due to the state-of-the-art of analog-digital converters (ADC) and the limitations on computational capacity of contemporary processors.
Once digitized, the signal is manipulated with signal processing techniques to extract the information. In this manner, A/D converters, general purpose processors and signal processing software replace the whole hardware based radio chain.
This approach is highly flexible and ideal because the same piece of equipment may be used for any new frequency, standard and application with simple software upgrades but is limited by the present state of the art of A/D converters and the limitations on computational capacity of present processors.
As stated in introduction, present A/D converters are limited in speed and resolution at high frequencies such as GHz. Moreover, when A/D converters are placed right after the

antenna, sampling is done over signals with very different strengths: the dynamic range of the signals may vary from μV to V. Current ADC resolutions are not able to cover such dynamic ranges. Nevertheless, significant research efforts are taking place to surmount this A/D converters exhibit higher power consumption than slower ones. If power consumption is very high, the A/D converter could dissipate too much heat and overheat the device. This issue is particularly critical in mobile devices, where cooling systems cannot be installed and the battery life is an extremely limiting factor. In fact, for mobile devices, the ADC power consumption should be within the range of 50 to 150 mW.
Currently, there are two trends in the A/D research. Some researchers direct their efforts to achieve high speeds, others focus on reducing the power consumption.
Another problem concerns computational capacity. When placing an A/D right after the antenna, the converter will be required to digitize the whole band, for instance from baseband to one (or more) GHz.
Such wideband receivers are still not a viable option for cost effective solutions. Also, the software must filter the samples to select the targeted signals. Such filtering has enormous computational cost that, today, can be only achieved using multiple processors.

To surmount the present problems of RF digitization we can chose a different point where to take the signal to digitize. Like in the traditional radio, we can exploit the conversion to intermediate frequency (IF).
A limited bandwidth of initial spectrum can be acquired, but we can freely chose a convenient IF to match ADC limits. This is the base of actual SDR implementations.
This design requires an RF front-end, which consists of a RF filter, a RF/IF converter and an IF filter. Subsequently the A/D converter digitizes the signal. The sampling frequency can be greater than required rate and can be adjusted later with a digital downconverter (DDC). The same applies to the transmission line with a digital upconverter (DUC). Then, the digital baseband signal can be processed with whatever processor and software.
There are two main advantages of this configuration. Firstly, current A/D converters can achieve enough speed and resolution at IF frequencies. Secondly, this design requires less computational resources because the tunable RF filter of the front-end limits the number of received channels and thus the actual sample rate.

1.4 GNU Radio project

From project site ([4]):

> *GNU Radio is a free software development toolkit that provides the signal processing runtime and processing blocks to implement software radios using readily-available, low-cost external RF hardware and commodity processors.*
> *It is widely used in hobbyist, academic and commercial environments to support wireless communications research as well as to implement real-world radio systems.*

Founded by Eric Blossom, GNU Radio wants to allow an easy building of modulation, demodulation, or more complex signal processing systems through the combination of signal processing blocks.
The core of GNU Radio is composed of a set of signal processing primitives written in C++. Furthermore, by using SWIG, an interface compiler which allows an easy integration of C/C++ into scripting languages, GNU Radio provides a simple interface to the signal processing blocks from Python. Thus, the power of scripting languages is used to simply connect the signal processing blocks which can run at native speed without any interpretation. A detailed explanation about available blocks is present in Section 1.4.3.

1.4.1 Applications

GNU Radio was initially adopted by amateur radio enthusiasts, but the popularity of the open-source hardware/software has stretched the interest to universities as well.
Today, several applications are already written with GNU Radio. For example (from [12]), there are applications which decode HDTV pictures, which can receive and send AM/FM broadcast radio, and there is support for some simple modulation schemas like AM/FM/PSK.
Additionally there is an implementation of packet radio system using GMSK modulation and demodulation to transmit packets from one host to an other. The problem with this

system is, that GNU Radio doesn't have a good support for packet based processing since it is stream oriented.
A DARPA project called "Adaptive Distributed Radio Open-source Intelligent Network (ADROIT)" attempts to change this by implementing a new primitive into GNU Radio, called the "M-block". This will allow a simpler implementation of block based processing and it will also allow to annotate data with meta data.

Here is a list of published applications. Most of these works derive from the use of the USRP board (described at page 17).
A complete GPS receiver has been implemented with the DBSRX daughterboard. It receives GPS signals, and contains interfaces to the Google Earth data. Eric Blossom presented a passive radar by using FM frequencies and exploiting the MIMO capabilities of the USRP.
A DVB module ([10]) with a full software implementation is capable of transmitting DVB-T compliant signals in realtime. The corresponding receiver part is in development.
Part of a DAB receiver has also been implemented, including a working physical layer, part of Fast Information Channel (FIC) to see the names of the radio stations, but without the Main Service Channel (required to actually hear something).
Wiser S.r.l. is working on a GSM-R Signal Integrity Detection System that should be available soon.
Implementations for IEEE 802.11, Bluetooth, IEEE 802.15.4, and GSM are also available with certain limitations because of USB 2.0 bandwidth, but should be resolved with the new USRP 2 interface.

1.4.2 Flowgraphs

With GNU Radio is simple to build an FM receiver just creating a graph where the nodes are signal processing blocks and the edges represent the data flow between them.
The graph, called "flowgraph", is a group of signal processing blocks that are connected together to reach a target that is usually a communications system.
Behind flowgraphs, there is a stack (see Figure 1.3) of different hardware and software

layers that makes the user create its own project transparently to the implementation. Everything concerns a flowgraph is written in Python programming language. A primary Python thread has the goal of initializing blocks and handling runtime parameter modifications to the flow graph (e.g. multiplier, squelch level, etc.). Any information (i.e. samples or bits) passing through a flow graph block is managed in a separate thread, the signal processing thread.

In order to start the flow graph, each of the blocks must be connected together using the input and output ports of the flow graph block. The first block in the flow graph is called the source block and only contains output connections. The source block can be a sine wave, a set of samples stored in a file, a vector, or the USRP (for receiving), etc. The last block in the flow graph is called the sink block and only contains input connections. The sink block can be an output to a file, or a vector, or the USRP (for transmitting), etc. All other blocks contain both input and output connections. All the details can be read in the following section.

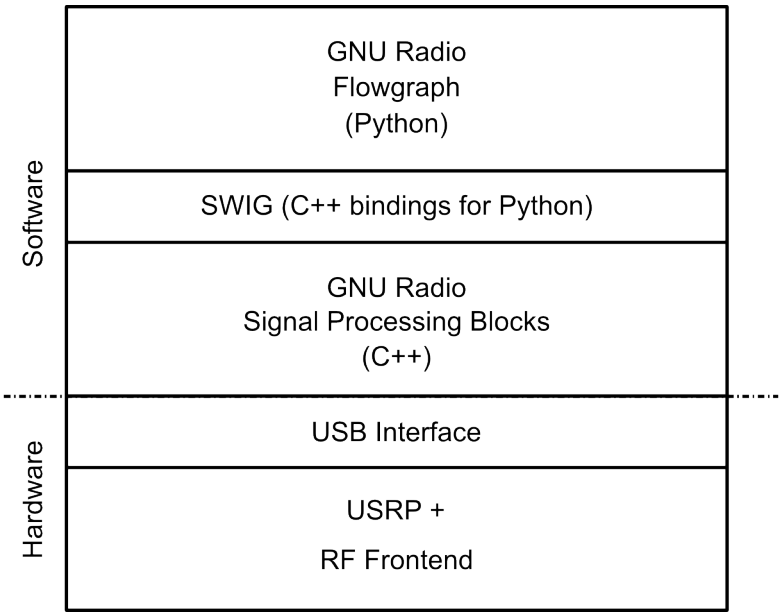


Figure 1.3: GNU Radio hardware and software stack

1.4.3 Blocks

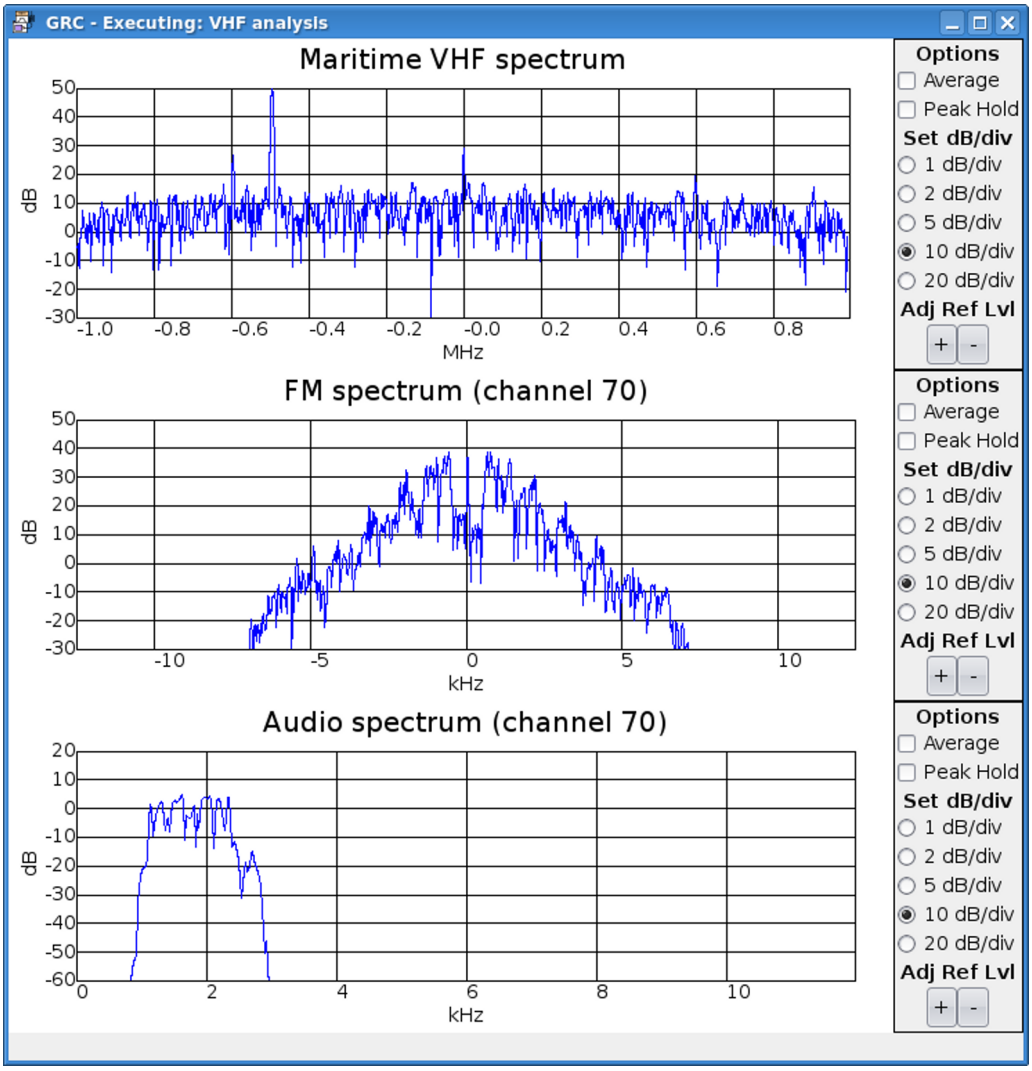
Signal processing blocks are implemented in C++ exploiting some CPU optimized routines, written in Assembler language, and other optimized libraries, like FFTW for Fourier transforms.

Conceptually, a signal processing block processes an infinite stream of data flowing from its input ports to its output ports except special blocks, like sources and sinks, that have only one port. Block attributes include the number of input and output ports it has as well as the type of data that flows through each.

More than 100 blocks are currently implemented in GNU Radio.

A summary is reported below:

- Sources (signal, noise, vector, random, null, file, udp, audio, wav, pad)
- Sinks (vector, null, file, udp, audio, wav, pad)
- Graphical sinks (number, scope, fft, constellation, waterfall)
- Operators (arithmetic ops, logic ops, fft, max, rms, integrate, conjugate)
- Type conversion (complex, float, [i]short, [u]char, mag, square mag, real, imag)
- Stream conversion ([de]interleave, stream to streams, vector to stream[s])
- Synchronizers (clock recovery, pll, correlator, psk sync, packet [en,de]coder)
- Level controls (peak detector, agc, mute, squelch, threshold, sample and hold)
- Filters (lp, hp, bp, br, fft, iir, hilbert, goertzel, resampler, decim, interp)
- Modulators (vco, [wb,nb]fm, pm, fsk, quad demod, constellation, gmsk, qam)
- USRP (source, sink, dual source, dual sink)
- Variables (variable, slider, chooser, text box, sink, parameter)
- Misc (import, throttle, delay, repeat, selector, valve, nop, head, copy)



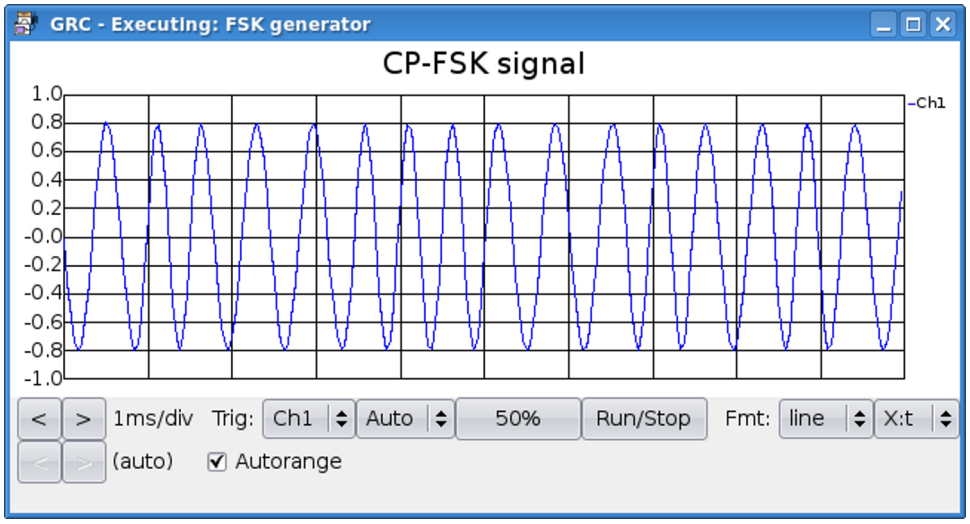


Figure 1.4: GNU Radio graphical components: Scope and FFT analyzers

1.4.4 GRC

Programming applications with GNU Radio toolkit has some drawbacks. "Toolkit" means a set of libraries and programs that offers a quite simple interface to complex functions. Before GRC, that is now part of the toolkit, the program writing was difficult due to the steep learning curve of GNU Radio. With no graphical helpers, to generate a flowgraph, a user had to learn how to write looking to other programmers' work. Since everything is command line based, and the library is very large, the learning period prior to have a result was quite long.

To aid beginners, Josh Blum has developed a graphical interface for GNU Radio. This GUI, named GNU Radio Companion (GRC), allows users to interact with GNU Radio signal blocks in a manner similar to Labview or Simulink (see Figure 1.5).

The entire interface is completely designed with GNU Radio in mind, and encompasses over 150 blocks from the GNU Radio Project. New blocks can be integrated into GRC via descriptive Python definitions. The definitions are very flexible, and allow multiple GNU Radio blocks to be grouped into a single GRC super-block.

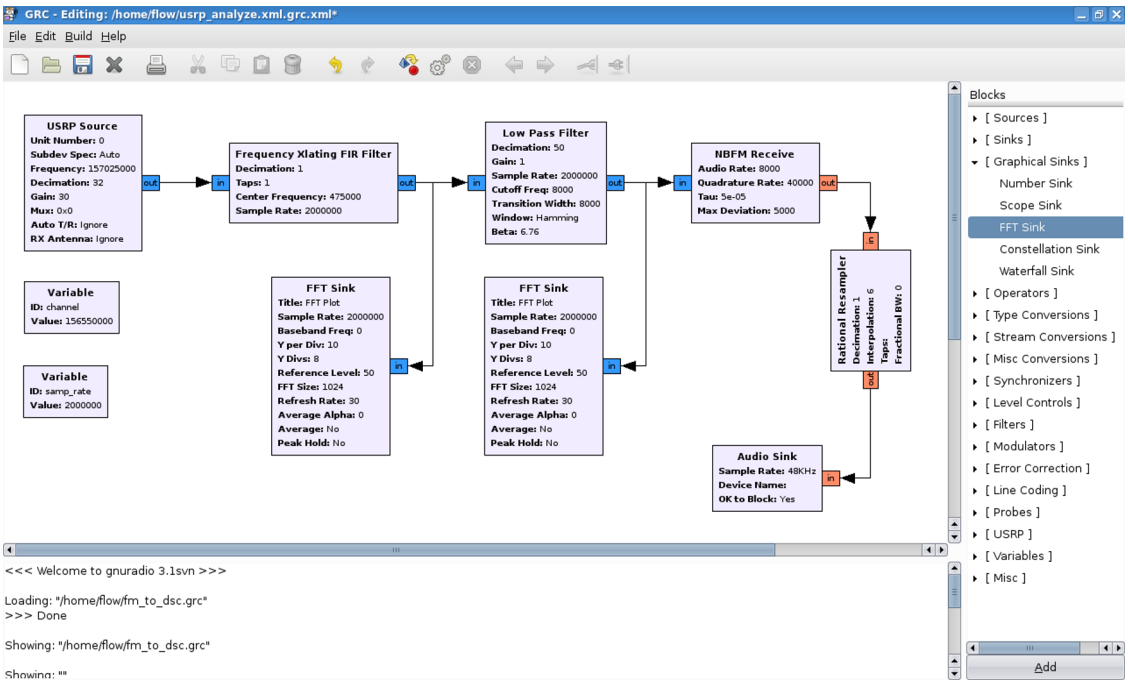


Figure 1.5: Designing a flowgraph with GRC

GRC quickly realizes what you have in mind. Just selecting some block and connecting them together with some clicks and the flowgraph is ready. After adding some processing blocks, represented as rectangles, the working space can be a little crowded and connecting lines can be overlap themselves.
The working space can be sized up to 2048 x 2048 pixels.
To restore a better environment, blocks can be manually reorganized, moved and rotated.
With the mouse is also possible to change the properties for each block with a double-click.
When handling the signal routing between blocks, the interface aids the user denying or highlighting wrong choices (i.e. the user connects blocks with different that accepts unmatching types of data).
Only when the graph is consistent, from source to sink, the GUI allows to create a Python file with the flowgraph.
The graph validation occurs by following these steps:

- Connections are made between input and output sockets of the same data type. Different data types have different colors. Therefore, input and output colors must match. Invalid connections are highlighted red.
- All sockets must be connected (except for the optional sockets). Disconnected sockets cause their signal block to have a red label.
- All signal block parameters must be valid. For instance, numerical expressions can be parsed. Invalid parameters have red labels and cause their signal block to have a red label.

GRC is also able to directly run (and stop) created programs. Both command line programs, Wx windowed programs and hier blocks can be created.

1.5 Universal Software Radio Peripheral

The Universal Software Radio Peripheral (USRP) is a device, linked with GNU Radio, which allows the creation of a software defined radio. A USB 2.0 port allows its easy integration to a normal PC. Various plug-on daughterboards allow the USRP to be used on different radio frequency bands. Presently, daughterboards operating from DC to almost 5.9 GHz are available ([2]). The entire schematics design of the USRP is open source. A typical setup of the USRP board consists of one mother board and up to four daughter boards.

1.5.1 A valuable device

One USRP can simultaneously receive and transmit on two antennas in real time. All sampling clocks and local oscillators are fully coherent, thus allowing the creation of MIMO (multiple input, multiple output) systems. In the USRP, high sampling rate processing takes place in the FPGA, while lower sampling rate processing occurs at the host computer. The two onboard digital downconverters (DDCs) mix, filter, and decimate (from 64MS/s) incoming signals in the FPGA. Two digital upconverters (DUCs) interpolate baseband signals to 128 MS/s before translating them to the selected output frequency. The DDCs and DUCs combined with the high sampling rates also greatly simplify analog filtering requirements. Daughterboards mounted on the USRP provide flexible, fully integrated RF front-ends. The USRP accommodates up to two RF transceiver daughterboards (or two transmit and two receive) for RF I/O.

At the moment, there is nothing comparable to the USRP in the market. Its price is very competitive among high-end devices, and its performance is superior to other common implementations.

Of course, this board has some limitations, that will be described further, thus a new model, USRP 2, with major improvements is near to be commercialized.

1.5.2 Hardware components

The USRP consists of a motherboard containing four ADCs, four DACs, a million gate Field Programmable Gate Array (FPGA) and a programmable USB 2.0 controller.
Each component on the board is fully documented, allowing the owners to modify or add frontends to address every need.
Subsequently, a detailed description explains hardware and software properties.

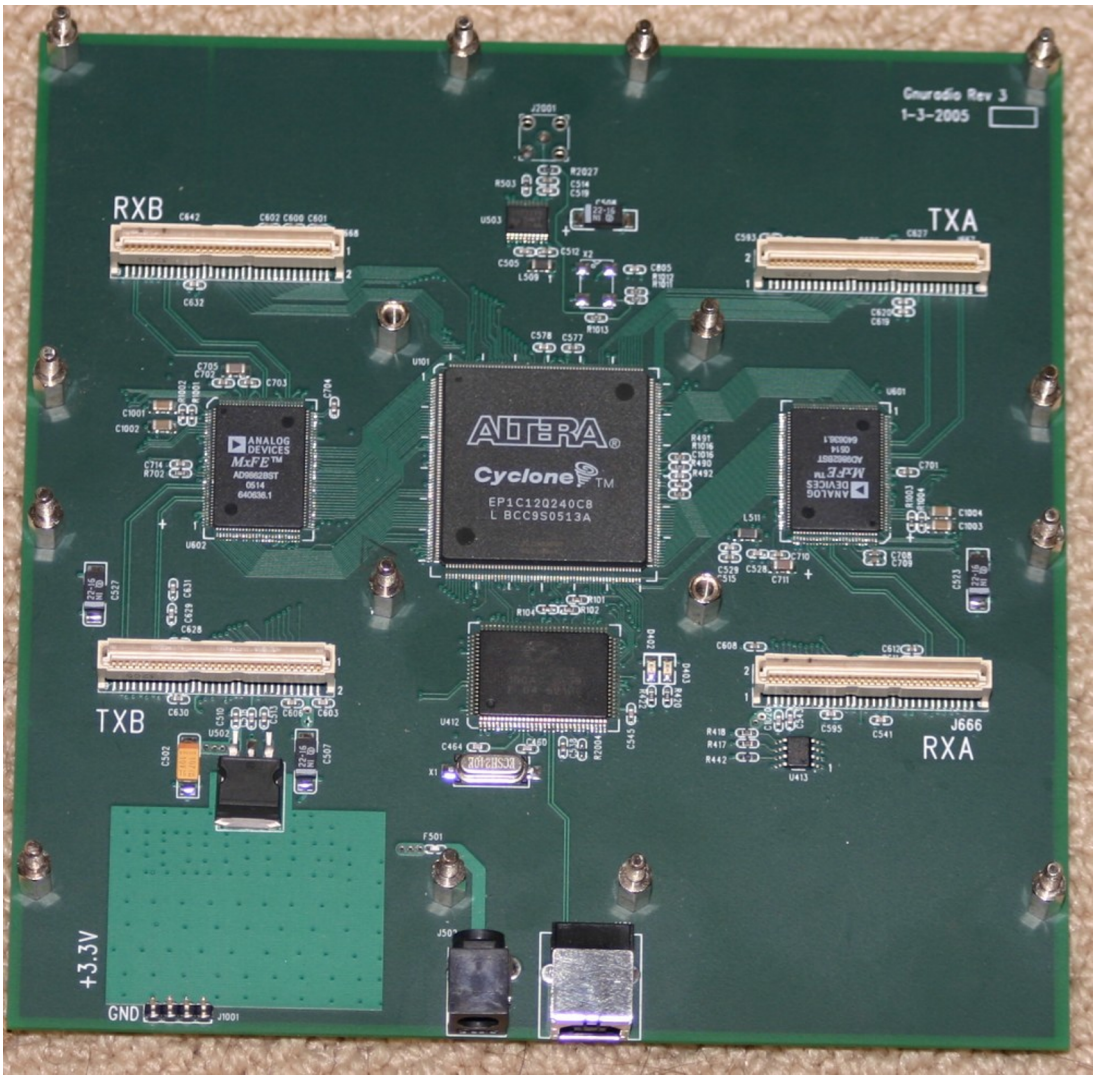


Figure 1.6: A plain USRP board (official photo)

Cypress USB controller

From CY7C68013's datasheet:

> *Cypress's EZ-USB® FX2^{TM} is the world's first USB 2.0 integrated microcontroller. By integrating the USB 2.0 transceiver, SIE, enhanced 8051 microcontroller, and a programmable peripheral interface in a single chip, Cypress has created a very cost effective solution that provides superior time-to-market advantages.*
>
> *The ingenious architecture of FX2 results in data transfer rates of 56 MByte/s, the maximum allowable USB 2.0 bandwidth, while still using a low-cost 8051 microcontroller in a package as small as a 56 SSOP. Because it incorporates the USB 2.0 transceiver, the FX2 is more economical, providing a smaller footprint solution than USB 2.0 SIE or external transceiver implementations.*
>
> *With EZ-USB FX2, the Cypress Smart SIE handles most of the USB 1.1 and 2.0 protocol in hardware, freeing the embedded microcontroller for application-specific functions and decreasing development time to ensure USB compatibility. The General Programmable Interface (GPIF) and Master/Slave Endpoint FIFO (8- or 16-bit data bus) provides an easy and glueless interface to popular interfaces such as ATA, UTOPIA, EPP, PCMCIA, and most DSP/processors.*

In other words, it is a high-speed USB peripheral controller that allows to initialize the FPGA (loading firmware) and further data transfer from/to the ADC/DACs. It is handled with a Python user space driver that communicates via 3 USB endpoints, 1 for commands and 2 (in bulk mode) for sample in/out.

Cyclone FPGA

The core of the USRP is an Altera FPGA, more exacly a Cyclone EP1C12Q240C8. Understanding the FPGA operation is one of the important steps for GNU Radio users. As shown on Figure 1.8, all the ADCs and DACs are connected to the FPGA. The FPGA plays a key role in the GNU Radio system. Basically what the FPGA does is to perform high bandwidth math, and to reduce the data rates to something that can be easily transferred over a USB 2.0 interface. The FPGA connects to an USB 2.0 interface chip, the Cypress FX2. Everything (FPGA circuitry and USB 2.0 Microcontroller) is programmable over the USB 2.0 bus.

The standard FPGA configuration (firmware) includes digital down converters (DDC) implemented with cascaded integrator-comb (CIC) filters. CIC filters are very high-performance filters using only adds and delays. The FPGA implements 4 digital down converters (DDC), which allows 1, 2 or 4 separate RX channels. At the RX path, there are 4 ADCs and 4 DDCs. Each DDC has two inputs I and Q. Each of the 4 ADCs can be routed to either the I or the Q input of any of the 4 DDCs, which allows for having multiple channels selected out of the same ADC sample stream.

The digital up converters (DUCs) on the transmit side are actually contained in the AD9862 CODEC chips, not in the FPGA. The only transmit signal processing blocks in the FPGA are the interpolators. The interpolate outputs can be routed to any of the 4 CODEC inputs.

The multiple RX channels must all be the same data rate. The same applies to the TX channels, where each channel must be at the same data rate, which may be different from the RX rate.

The digital down converter (DDC), first, down converts the signal from the IF band to the base band. Second, it decimates the signal so that the data rate can be adapted by the USB 2.0 and is reasonable for the computers' computing capability. The complex input signal (IF) is multiplied by the constant frequency (usually also IF) exponential signal.

The resulting signal is also complex and centered at 0. Then the signal is decimated with a factor of N.

The decimator can be treated as a low pass filter followed by a downsampler. Consider the decimation factor to be D. The digital spectrum, the low pass filter selects the band [-p/D, p/D], and the downsampler spreads the spectrum in [-p/D, p/D] to [-p, p]. Thus, the bandwidth of the digital signal of interest is narrowed down by a factor of D.
The USB 2.0 interface can sustain about 32 MByte/s across it. All samples sent over the USB 2.0 interface are in 16-bit signed integers in IQ format, i.e. 16-bit I and 16-bit Q data (complex), resulting in 8 MS/s (complex) across the USB 2.0.
This provides a maximum effective total spectral bandwidth of about 8 MHz by Nyquist criteria. Of course a much narrower range can be selected by changing the decimation rate.
For example, the bandwidth of a FM station is generally 200 kHz. Therefore the decimation factor can be set to 250, then the data rate across the USB 2.0 is 64 MHz / 250 = 256 kHz, which is well suited for the 200 kHz bandwidth without losing any spectral information.
The IF frequency of the DDC can be set using usrp.set_rx_freq() method and the decimation factor can be set using the usrp.set_decim_rate() method in Python. The decimation rate must be in the range [1,256].
Note that when there are multiple channels (up to 4), the channels are interleaved. For example, with 4 channels, the sequence sent over the USB 2.0 interface would be I0 Q0 I1 Q1 I2 Q2 I3 Q3 I0 Q0 I1 Q1, etc.

The MUX is like a router or a circuit switch. It determines which ADC (or a zero constant) is connected to each DDC input. As shown on Figure 1.7, there are 4 DDCs, each having two inputs. The MUX is controlled with Python.

Each input specified with (I0, Q0, I1 ... I3, Q3), i.e denoting which ADC is connected to it by using 4 bits (0, 1, 2, 3 or 0xf). Therefore a 32-bit integer would be enough for all 8 inputs to know which ADC is connected.
Of course an integer in hexadecimal system will be more convenient if we want to use

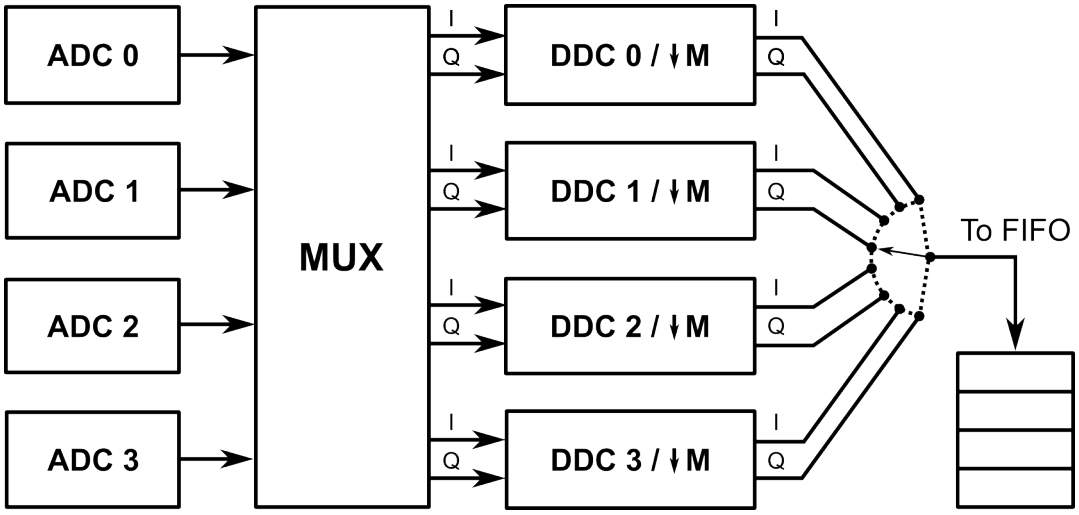


Figure 1.7: Flow multiplexing in the USRP

the set_mux() method. For most real sampling applications, the Q input of each DDC is constant zero. So quite often the modification of standard configuration of the FPGA is not required.

The USRP can operate in full duplex mode. When in duplex mode, the transmit and the receive sides are completely independent of one another. The only consideration is that the combined data rate over the USB 2.0 bus must be 32 MByte/s or less.
Finally when the I/Q complex signal enters the computer via the USB 2.0 interface, that's where the software takes over the processing.
At the TX path, the story is pretty much the same, except that it happens in inverse. A baseband I/Q complex signal needs to be sent to the USRP board from the computer. The digital up converter (DUC) interpolates the signal, up converts it to the IF band, and finally sends it through the DAC.

ADCs

The 4 high-speed 12-bit AD converters can sample at a rate of 64 MS/s. In theory, it could digitize a band as wide as 32 MHz.
The AD converters can bandpass-sample signals of up to about 150 MHz. If an IF signal larger than 32 MHz is sampled, it introduces aliasing. The higher the frequency of the

sampled signal, the more the SNR degrades by jitter. The recommended upper limit is 100 MHz.

The full range on the ADCs is 2V peak-to-peak, and the input is 50 Ohms differential, i.e 40 mW, or 16 dBm.

A programmable gain amplifier (PGA) is used before the ADCs to amplify the input signal, and to utilize the entire input range of the ADCs, in case the signal is weak. The PGA range is up to 20dB.

Other sampling rates can be used if needed. The available rates are all submultiples of 128, such as 64, 32 etc. In principle, the 4 input and 4 output channels use real samplings. However, there will be more flexibility (and bandwidth) if a complex (I/Q) sampling is used. For instance, the input and output channels can be paired up, resulting in 2 complex inputs and 2 complex outputs.

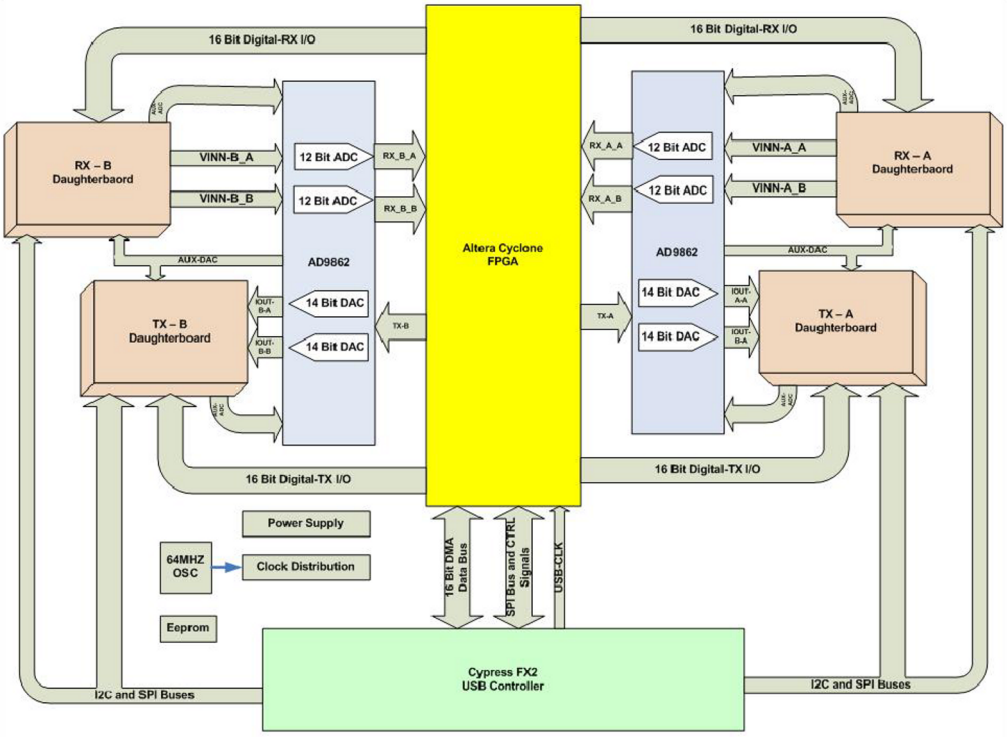


Figure 1.8: USRP block diagram

DACs

At the transmitting path, there are 4 high-speed 14-bit DA converters. The DAC clock frequency is 128 MHz, thus Nyquist frequency is 64 MHz. Staying below 50 MHz or so is suggested to make filtering easier, therefore the useful output frequency range is DC to about 50 MHz.
The DACs can supply a peak output of 1 V, or 10 mW (10 dBm), to a 50 Ohm differential load.
Similarly to the RX path, in the TX path there is programmable-gain amplifier that provides up to 20 dB gain to the DAC output.

Analog and digital I/O

There are 8 Auxiliary analog input channels connected to low speed 10 bit ADC inputs which can be read from software. These ADCs can convert up to 1.25 MS/s and have a bandwidth of around 200 kHz. These analog channels are useful for sensing the RSSI signal levels, temperatures, bias levels, etc.

Additionally, there are 8 analog output channels connected low-speed 8bit DAC outputs. These DACs can be used for supplying various control voltages such as external variable gain amplifiers control. Moreover, there are two additional DACs which are constructed using 12 bit sigma-delta modulator with external simple low pass filter.
The USRP motherboard connectors share one set of the 4 analog output channels and each have 2 independent analog input channels. The RXB and TXB share their other own independent set. There is also AUX_ADC_REF which can provides a reference level for gain setting if it is necessary.

The USRP motherboard has also high speed 64 bit digital I/O ports. These are divided in two groups (32 bit for IO_RX and 32 bit for IO_TX). These digital I/O pins are connected to the daughterboards interface connectors. Each of these connectors has 16 bit digital I/O bits.
These signals can be controlled from software by reading/writing to special FPGA regis-

ters and each can be independently configured either as digital input or digital output. Some of these pins are used to control specific operations on the installed daughterboards such as controlling the selection of receiving RF input port, in automatic transmit/receive mode, controlling power supply feeding for different TX and RX parts, synthesizer lock detection, etc. It can be also used to implement AGC processing and can be very helpful in debugging FPGA implementations when connected to a logic analyzer.

Figure 1.9: USRP enclosure: daughterboards connectors replicated on front panel.

Daughterboards

The daughterboards are used to hold an RF receiver interface or an RF transmitter or both. The USRP has 4 slots that can be used to plug in up to 2 RX daughterboards and 2 TX daughterboards.

The TX slots are labeled TXA and TXB, and the 2 corresponding RX slots RXA and RXB. Each daughterboard slot has access to 2 of the 4 high-speed AD/DA converters, this allows each daughterboard, which uses real (not I/Q) sampling, to have 2 independent RF sections (RX or TX) and 2 antennas (4 total for the system). If complex I/Q sampling is used, each board can support a single RF section, for a total of 2 for the whole system.

Usually, on each daughterboard there are one or two SMA connectors that can be used to connect the input or output signals to the board. An exception is the TV-RX board, used for this thesis, that mounts an integrated TV tuner with F connector.

Here is a list of currently available daughterboards:

- BasicTX: Transmitter for use with external (user supplied) RF hardware
- BasicRX: Receiver for use with external (user supplied) RF hardware
- LFTX: DC-30 MHz Transmitter
- LFRX: DC-30 MHz Receiver
- TVRX: 50 MHz to 870 MHz Receiver
- DBSRX: 800 MHz to 2.4 GHz Receiver
- RFX400: 400 MHz - 500 MHz Transceiver, 100+mW output
- RFX900: 800-1000MHz Transceiver, 200+mW output
- RFX1200: 1150 MHz - 1450 MHz Transceiver, 200+mW output
- RFX1800: 1.5-2.1 GHz Transceiver, 100+mW output
- RFX2400: 2.3-2.9 GHz Transceiver, 20+mW output
- XCVR2450: 2.4-2.5 GHz and 4.9 to 5.85 GHz Dual-band Transceiver, 100+mW output on 2.4 GHz, 50+mW output on 5GHz

Chapter 2

Maritime Communications

An Italian engineer, Guglielmo Marconi, invented radio in 1895. The first use of radio, for communication over sea, succeeded on 3 March of 1899 when the East Goodwin Lightship which was anchored ten miles offshore from the south east coast of England required assistance. A distress call was transmitted by wireless to a shore station at South Foreland and help was dispatched. Since the invention of radio, ships at sea have relied on Morse code for maritime communications. Although telecommunications technology is improving quickly, people at sea do not have access to the same telecommunications infrastructure people ashore have.

Morse encoded distress calling saved thousands of lives since its inception almost a century ago, but its use requires skilled radio operators spending many hours listening to the radio distress frequency. Its range on the medium frequency (MF) distress band (500 kHz) is limited, and the amount of traffic Morse signals can carry is also limited.

Mariners, not only need to access international shore telephone and data public switched networks, but also, they need to be able to communicate with other ships of any size or nationality, to receive and send urgent maritime safety information, and to send or receive distress alerts in an emergency to or from rescue coordination centers ashore and nearby ships anywhere in the world.

Maritime telecommunications systems must be internationally interoperable, affordable, acceptable and available to most ships and maritime countries.

Over fifteen years ago the International Maritime Organization (IMO) ([3]), a United Nations agency specializing in safety of shipping and preventing ships from polluting the seas, began looking at ways of improving maritime distress and safety communications. In 1979, a group of experts drafted the International Convention on Maritime Search and Rescue, which called for development of a global search and rescue plan.

This group, in close co-operation with the International Telecommunication Union (ITU) and other international organizations, notably the World Meteorological Organization (WMO), the International Hydrographic Organization (IHO) and the COSPAS-SARSAT partners passed a resolution calling for development of a Global Maritime Distress and Safety System (GMDSS) to provide the communication support needed to implement the search and rescue plan. The main guideline of the IMO convention was solving the inadequacy of existing distress and safety systems.

The subsequent development process enabled the shipping industry to put the new distress and safety system into use on 1 February 1 1992. The SOLAS (Safety Of Life At Sea) Convention introduced some stages between 1993 and 1 February 1999 to allow the migration from old systems and the new GMDSS. On that last date, Morse wireless telegraphy, used since the beginning of the 20th century, discontinued worldwide.

The GMDSS creation is marked the biggest change to maritime communications since the invention of radio.

2.1 Global Maritime Distress and Safety System

The basic concept of GMDSS is that search and rescue authorities ashore, as well as shipping in the immediate vicinity of the ship in distress, will be rapidly alerted to a distress incident so that they can assist in a co-ordinated search and rescue operation with the minimum delay.

The system also provides for agency and safety communications and the promulgation of Maritime Safety Information (MSI): navigational and meteorological warnings and forecasts and other urgent safety information to ships.

In other words, every ship is able, irrespective of the area in which it operates, to perform

those communication functions which are essential for the safety of the ship itself and of other ships operating in the same area.
Major changes affects the users: before it, the communications were carried by radiotelegraphs with the need of a specialized operator; now the operator needs a certificate, GOC or ROC, released from the national organs that controls the sea traffic, that does not requires an advanced or specific knowledge to the operator, for instance the Morse code. Furthermore, automatic distress alerting and locating task become available also if the radio operator does not have time to send an SOS or MAYDAY call, and, for the first time, ships are required to receive broadcasts of maritime safety information which could prevent a distress from happening in the first place ([6]).
The system is able to reliably perform the following functions: alerting (including position determination of the unit in distress), search and rescue coordination, locating (homing), maritime safety information broadcasts, general communications, and bridge-to-bridge communications. The system also provides redundant means of distress alerting, and emergency sources of power.

Specific radio carriage requirements depend upon the ship's area of operation, rather than its tonnage. Under IMO legislation, all passenger vessels and all other vessels over 300 GRT had to be fitted with the necessary equipment. By 1 August 1993, all these ships are required to be equipped with satellite emergency position-indicating radiobeacons (EPIRBs) and NAVTEX receivers, to automatically receive shipping safety information. The required onboard equipment depends also on the area of operation as described in the following section.

2.2 Sea areas

GMDSS sea areas serve two purposes: to describe areas where GMDSS services are available, and to define what GMDSS ships must carry. Prior to the GMDSS, the number and type of radio safety equipment ships had to carry depended upon its tonnage. With GMDSS, the number and type of radio safety equipment ships have to carry depend upon the areas in which they travel. GMDSS sea areas are defined by governments.

Specific equipment requirements for ships vary according to the sea area in which the ship operates. The GMDSS combines various subsystems - which all have different limitations with respect to coverage - into one overall system, and the oceans are divided into four sea areas:

- **Area A1**: Within range of VHF coast stations with continuous DSC (2187.5 kHz) alerting available (about 20-30 miles).
- **Area A2**: Beyond area Al, but within range of MF coastal stations with continuous DSC alerting available (about 100 miles). GMDSS-regulated ships travelling this area must carry a DSC-equipped MF radiotelephone in addition to equipment required for Area A1.
- **Area A3**: Beyond the first two areas, but within coverage of geostationary maritime communication satellites (Inmarsat). This covers the area between roughly 70 deg N and 70 deg S. Ships travelling this area must carry either an Inmarsat F77, B or C ship earth station, or a DSC-equipped HF radiotelephone/telex, in addition to equipment required for an A1 and A2 Area.
- **Area A4**: The remaining sea areas. The most important of these is the sea around the North Pole (the area around the South Pole is mostly land). Geostationary satellites, which are positioned above the equator, cannot reach this far. Ships travelling these polar regions must carry a DSC-equipped HF radiotelephone/telex, in addition to equipment required for areas A1 and A2.

In addition to previously listed equipments, all GMDSS-regulated ships must carry a satellite EPIRB, a NAVTEX receiver (if they travel in any areas served by NAVTEX), an Inmarsat-C SafetyNET receiver (if they travel in any areas not served by NAVTEX), a DSC-equipped VHF radiotelephone, two or more VHF handhelds, and a search and rescue radar transponder (SART).

2.3 NAVTEX

NAVTEX is an international, automated system for instantly distributing maritime navigational warnings, weather forecasts and warnings, search and rescue notices and similar information to ships.
The NAVTEX system is designed to be used in GMDSS Sea Area A2, and is utilized mainly by those countries with relatively small areas of coastline and/or sea areas to cover. System range is generally 300 or so nautical miles from the transmitter.
Major areas of NAVTEX coverage include the Mediterranean Sea, the North Sea, coastal areas around Japan and areas around the North American continent.
The system mainly operates in the Medium Frequency radio band just above and below the old 500 kHz Morse Distress frequency. Three broadcast frequencies has been allocated for NAVTEX:

- 518 kHz - the main NAVTEX channel
- 490 kHz - used for broadcasts in local languages (i.e. non-English)
- 4209.5 kHz - allocated for NAVTEX broadcasts in tropical areas - not widely used at the moment.

All broadcasts from stations within the same NAVAREA must be coordinated on a time sharing basis to eliminate interference. In addition, power outputs from each station are adjusted to control the range of each broadcast. This is particularly important during night-time hours, as medium frequencies always have a large propagation.

All messages are received by a printing radio receiver (Figure 2.1), installed in the pilot house of a ship or boat, that checks each incoming message to see if it has been received during an earlier transmission. Messages can be filtered on a category base depending on the ship's master interest.

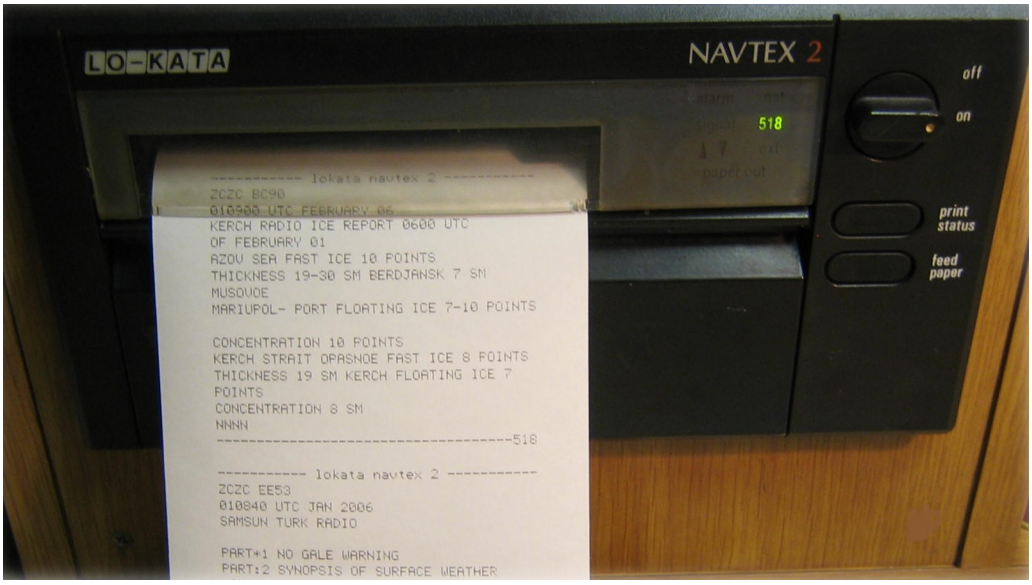


Figure 2.1: NAVTEX direct printing

2.4 COSPAS-SARSAT and EPIRBs

COSPAS-SARSAT is an international satellite-based search and rescue system, established by Canada, France, the U.S.A., and Russia. These four countries jointly helped develop a 406 MHz satellite emergency position-indicating radiobeacon (EPIRB), an element of the GMDSS designed to operate with COSPAS-SARSAT system.

These automatic-activating EPIRBs, now required on SOLAS ships, commercial fishing vessels, and other ships, are designed to transmit to a rescue coordination center a vessel identification and an accurate location of the vessel from anywhere in the world.

There are three types of distress radio beacons compatible with the system:

- EPIRBs (emergency position-indicating radio beacons) signal maritime distress.
- ELTs (emergency locator transmitters) signal aircraft distress.
- PLBs (personal locator beacons) are for personal use and are intended to indicate a person in distress who is away from normal emergency services.

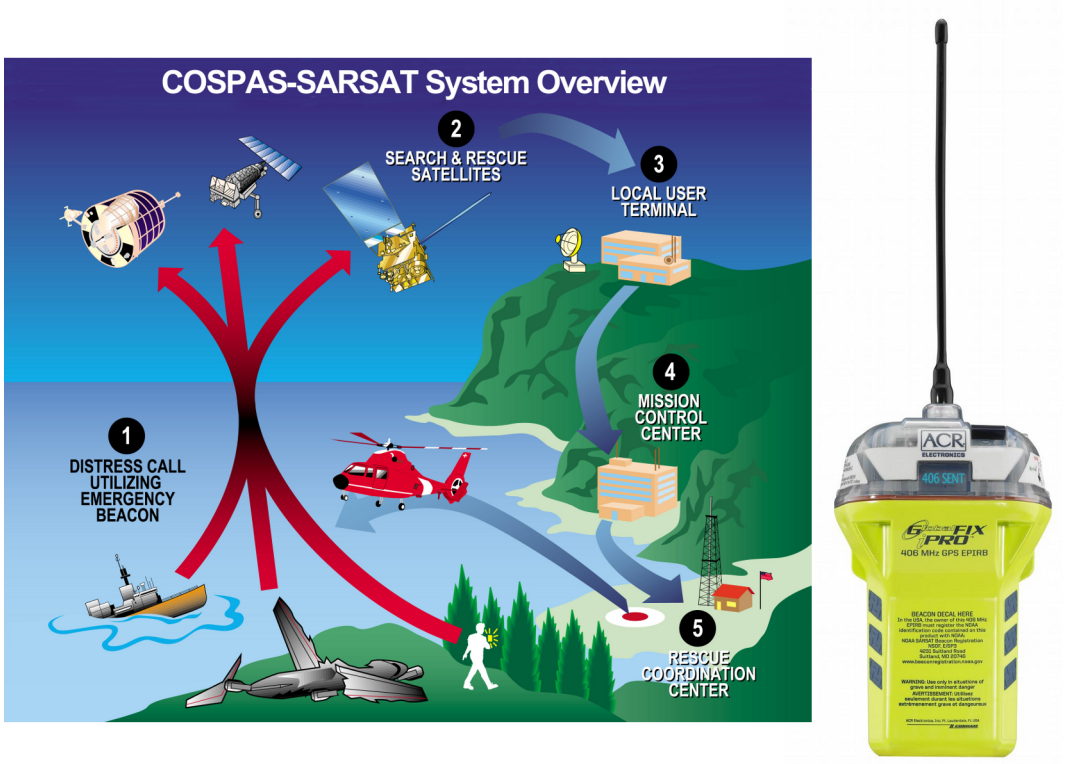


Figure 2.2: COSPAS-SARSAT operation schema and a GPS equipped EPIRB

Most beacons are brightly colored and waterproof. EPIRBs and ELTs are larger, and would fit in a cube about 30 cm on a side, and weigh 2 to 5 Kg (an example of EPIRB is shown on Figure 2.2). PLBs vary in size from cigarette-packet to paperback book and weigh 200 g to 1 Kg.

The EPIRB units can be purchased from marine suppliers or aircraft refitters and have a useful life of 10 years. Every unit is programmed with a unique identifier and registered to national coast guard service.
The frequencies of operation are:

- 121.5 MHz ± 6 kHz (frequency band protected to ± 50 kHz)
- 243.0 MHz ± 12 kHz (frequency band protected to ± 100 kHz)
- 406 MHz - carrier wave at 406.025 MHz ± 0.005 MHz

The 121.5 and 243 MHz are the most common and least expensive type of EPIRB, designed to be detected by overflying commercial or military aircraft. Satellites were designed to detect these EPIRBs, but are limited.

Instead, the 406 MHz EPIRB was designed to operate with satellites. The signaling frequency has been designated internationally for use only for distress. Other communications and interference, such as on 121.5 MHz, is not allowed on this frequency. Its signal allows a satellite local user terminal to accurately locate the EPIRB and identify the vessel anywhere in the world (there is no range limitation).

2.5 SARTs

Figure 2.3: An installed radar SART

SARTs are self contained, portable and buoyant radar transponders typically cylindrical, and brightly colored (see Figure 2.3).

These systems are designed to guide rescuers to your location. SARTs operate in the 9 GHz marine radar band, and when interrogated by a searching ship's radar, respond with a signal which is displayed on a radar screen: a line of 12 dots, where each dot corresponds to 0.6 nautical miles, extending outward from the SARTs position along its line of bearing. As the searching vessel approaches the SART, the radar display will change to wide arcs. These may eventually change to complete circles as the SART becomes continually triggered by the searching ship's radar. Some slight position error will also be caused by the SART switching from receive to transmit mode.

SARTs will also provide a visual and audible indication to users when interrogated by a

searching radar. The range achievable from a SART is directly proportional to its height above the water. A SART mounted at 1 m (i.e. in a liferaft) should be able to be detected at 5 nautical miles by a ship's radar mounted at 15 m. The same SART should be able to be detected at 30 nautical miles by an aircraft flying at 8000 feet.
GMDSS vessels from 300 to 500 GRT are required to carry 1 SART, and vessels over 500 GRT are required to carry 2.

In the future, from 2010, AIS-SARTs, instead of radar-SARTs, will be available. AIS-SARTs will be used, similarly to radar ones, to locate a survival craft or distressed vessel. The AIS-SART derives and transmit its position and get time synchronization from a built in GNSS receiver. Every minute the position is sent as a series of equal position reports, this is to maintain a high probability that at least one of the position reports is sent on the highest point of a wave. The specification for the AIS-SART is currently being developed by the AIS work group in IEC, TC80.

Figure 2.4: An AIS-SART

2.6 Inmarsat

The International Mobile Satellite Organization (Inmarsat), previously the International Maritime Satellite Organization, was established by IMO in 1976 to operate satellite maritime communication systems and has become a privately owned company, while retaining its public sector obligations to the maritime distress and safety system.

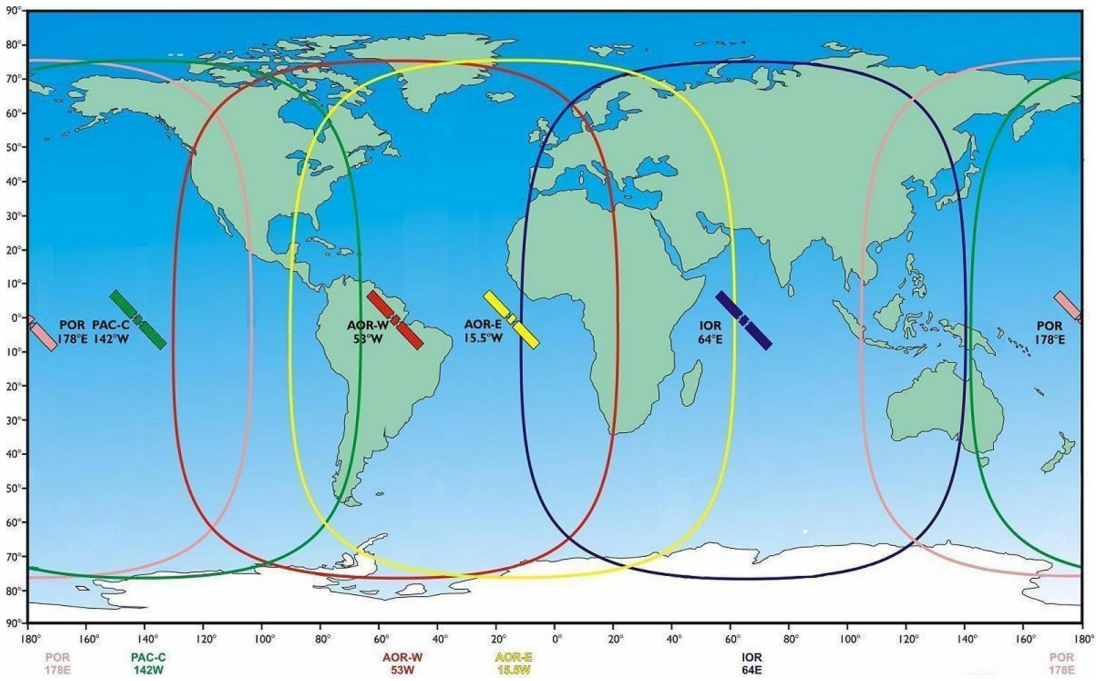


Figure 2.5: Inmarsat satellites coverage

Three types of Inmarsat terminals are recognized by the GMDSS: A, B and C.

The Inmarsat A and B provide ship/shore, ship/ship and shore/ship telephone, telex and high-speed data services, including a distress priority telephone and telex service to and from rescue coordination centers.

The Inmarsat C provides ship/shore, shore/ship and ship/ship store-and-forward data and telex messaging, the capability for sending preformatted distress messages to a rescue coordination center, and the SafetyNET service. Inmarsat C equipment is relatively small and lightweight, and costs much less than an Inmarsat B.

The SafetyNET service is a satellite-based worldwide maritime safety information broadcast service of high seas weather warnings, navigational warnings, radionavigation warnings, ice reports and warnings generated by the International Ice Patrol, and other similar information not provided by NAVTEX. SafetyNET works similarly to NAVTEX in areas outside NAVTEX coverage.

SOLAS now requires that Inmarsat C equipment have an integral satellite navigation receiver, or be externally connected to a satellite navigation receiver. That connection will ensure accurate location information to be sent to a rescue coordination center if a distress alert is ever transmitted.

2.7 Radiotelephones (MF/HF/VHF)

Radiotelephone systems allows ship/ship or ship/shore wireless communication. They are similar to telephones but usually simplex (half-duplex) instead of duplex. This is because a duplex system would require a radio system to simultaneously transmit and receive on two separate channels, which both wastes bandwidth and presents some technical challenges. Simplex frequency operation allows one person to talk and the other to listen alternately. Another common way to communicate is the dual-frequency simplex: the communication splits into two separate channels, but only one is used to transmit at a time.
Normally, the user presses a special switch on the transmitter when he wish to talk. For this reason, the button is called the "push-to-talk" switch or PTT. Usually it is fitted on the side of the microphone or other obvious position. Users may use a special code-word such as "over" to signal that they have finished transmitting, or it may follow from the conversation.

Radiotelephones operates on different frequencies with different modulations. Common maritime channels are present at 2, 4, 6, 8, 12, 16, 18, 22 and 25 MHz for MF/HF, and near 160 MHz for VHF.

VHF Channel 16 (156.8 MHz) is monitored 24 hours a day by coastguards around the world. In addition, all sea bound vessels are required to monitor channel 16 VHF when sailing, except when communicating on other marine channels for legitimate business or operational reasons.
Coastguards and others are permitted to broadcast short informative safety messages on channel 16, however, it is an offense in most countries to make false "Mayday" calls or broadcast on channel 16 unless in distress.
On MF, 2182 kHz channel is analogous to VHF channel 16, but unlike VHF which is limited to about 50 nautical miles (90 km) range, communications on 2182 kHz (and nearby frequencies) have a typical range of around 150 nautical miles (280 km) during the day and 500 (or more) nautical miles at night.

On MF/HF channels, single-sideband (SSB) modulation is used. This because the short wave bands are crowded with many users, and SSB permits a single voice channel to use a narrower range of radio frequencies (bandwidth) than AM or FM.
On VHF, instead, the frequency modulation is used. This allows transmitted signal to be more robust against noise and interference.

Here are some more accurate technical specifications of maritime VHF:

- Allowed frequencies are from 156.000 to 162.500 with 25 kHz channelization. Channels are numbered from 1 to 28 and from 60 to 88 (see Table 2.1 for details).
- Modulations used are: G3E or F3E for voice channels and G2B or F1B for DSC.
- Maximum frequency deviation (for FM) is 5 kHz. Modulation index is 5/3.
- Audio signal (modulating signal) must be limited to 3 kHz.
- Pre-emphasis and de-emphasis starting at 300 Hz with 6 dB/octave filters.
- Authorized nominal bandwidth is 16 kHz for voice channels and 20 kHz for data.
- Power: coast stations 100 W, ship stations 6 to 25 W (50 W if it can be remotely reduced by coast stations) and 5 W for handhelds.

An example of VHF-DSC ship radiotelephone, is shown on subsequent pictures. This apparatus, kindly supplied by Elman S.r.l., is fully compatible with VHF-DSC standard and has been used for testing the DSC functionalities of the monitoring system.

Figure 2.6: Elman RTV-1077-D VHF-DSC transceiver (front panel)

Figure 2.7: Elman RTV-1077-D VHF-DSC transceiver (rear panel)

Channel number	Ship TX (MHz)	Base TX (MHz)	Simplex Duplex	Channel number	Ship TX (MHz)	Base TX (MHz)	Simplex Duplex
1	156.050	160.650	D	60	156.025	160.625	D
1A	156.050	156.050	S	60A	156.025	156.025	S
2	156.100	160.700	D	61	156.075	160.675	D
2A	156.100	156.100	S	61A	156.075	156.075	S
3	156.150	160.750	D	62	156.125	160.725	D
3A	156.150	156.150	S	62A	156.125	156.125	S
4	156.200	160.800	D	63	156.175	160.775	D
4A	156.200	156.200	S	63A	156.175	156.175	S
5	156.250	160.850	D	64	156.225	160.825	D
5A	156.250	156.250	S	64A	156.225	156.225	S
6	156.300	160.900	D	65	156.275	160.875	D
6A	156.300	156.300	S	65A	156.275	156.275	S
7	156.350	160.950	D	66	156.325	160.925	D
7A	156.350	156.350	S	66A	156.325	156.325	S
8	156.400	156.400	S	67	156.375	156.375	S
9	156.450	156.450	S	68	156.425	156.425	S
10	156.500	156.500	S	69	156.475	156.475	S
11	156.550	156.550	S	70	156.525	156.525	S
12	156.600	156.600	S	71	156.575	156.575	S
13	156.650	156.650	S	72	156.625	156.625	S
14	156.700	156.700	S	73	156.675	156.675	S
15	156.750	156.750	S	74	156.725	156.725	S
16	156.800	156.800	S	75	156.775	156.775	S
17	156.850	156.850	S	76	156.825	156.825	S
18	156.900	161.500	D	77	156.875	161.475	D
18A	156.900	156.900	S	77A	156.875	156.875	S
19	156.950	161.550	D	78	156.925	161.525	D
19A	156.950	156.950	S	78A	156.925	156.925	S
20	157.000	161.600	D	79	156.975	161.575	D
20A	157.000	157.000	S	79A	156.975	156.975	S
21	157.050	161.650	D	80	157.025	161.625	D
21A	157.050	157.050	S	80A	157.025	157.025	S
22	157.100	161.700	D	81	157.075	161.675	D
22A	157.100	157.100	S	81A	157.075	157.075	S
23	157.150	161.750	D	82	157.125	161.725	D
23A	157.150	157.150	S	82A	157.125	157.125	S
24	157.250	161.850	D	83	157.175	161.775	D
25	157.300	161.900	D	83A	157.175	157.175	S
26	157.350	161.950	D	84	157.225	161.825	D
27	157.400	162.000	D	84A	157.225	157.225	S
28	157.450	162.050	D	85	157.275	161.875	D
				85A	157.275	157.275	S
				86	157.325	161.925	D
				86A	157.325	157.325	S
				87	157.375	161.975	D
				87A	157.375	157.375	S
				88	157.425	162.025	D
				88A	157.425	157.425	S

Table 2.1: Maritime VHF channels

2.8 Digital Selective Calling

Digital Selective Calling (DSC) is one of the most important parts of GMDSS. DSC is primarily intended to initiate ship/ship, ship/shore and shore/ship radiotelephone and MF/HF radiotelex calls. DSC calls can also be made to individual stations, groups of stations, or "all stations" at once.

Each DSC equipped (and so GMDSS compliant) ship, shore station and group is assigned unique 9-digit, the Maritime Mobile Service Identity. The MMSI is built into the DSC equipment and is not user-changeable. If a DSC unit is moved to another vessel, or the vessel is sold, a technician must re-program the MMSI. MMSI's can also be assigned to a group of vessels. This can prove invaluable, for example, to fishing fleets and shipping lines to keep in touch with other vessels within that group. A call made to a group rather than an individual MMSI will alert all vessels within that group.

The Maritime mobile Access and Retrieval System (MARS), maintained by the ITU is a searchable database which can be used to identify a vessel or coast stations MMSI number. The database is updated on a weekly basis. The first three non-zero digits of an MMSI number are used to distinguish the country of origin. Leading zero's are used to distinguish between vessels (no leading zero's), vessel groups (one leading zero), and coast stations (two leading zeros).

DSC distress alerts, which consist of a preformatted distress message, are used to initiate emergency communications with ships and rescue coordination centers. DSC was intended to eliminate the need for persons on a ship's bridge or on shore to continuously guard radio receivers on voice radio channels, including VHF channel 16 (156.8 MHz) and 2182 kHz now used for distress, safety and calling.

DSC system is available on both MF (2187.5 kHz), HF (4207.5, 6312, 8414.5, 12577, 16804.5 kHz) and VHF (156.525 MHz) maritime frequencies. Messages are transmitted using frequency shift keying (FSK) with different specifications depending on the radio frequency of transmission.

In all the cases, the FSK modulated signal has a carrier frequency of 1700 Hz and is treated as an audio spectrum for subsequent transmission. In the MF/HF bands, the modulation is obtained with a 170 Hz shift between tones and has a symbol rate of 100 baud. The transmission is accomplished with a single-sideband transmission (J2B) at the frequencies listed above.
In the VHF band, the modulation is more performant, has a symbol rate of 1200 baud and the shift between tones is 800 Hz (1300/2100 Hz). The frequency tolerance for tones is ±10 Hz. The modulated spectrum is processed with a pre-emphasis filter of 6 dB/octave and then transmitted over the air with the FM modulation.

Using digital coding, DSC automates all the radio functions with which existing marine operators are familiar. It also relieves the person at the other end from the tedious task of manual watch keeping. All the old familiar functions are still in place, but they now have English names and are accessible at the touch of a button.
In a distress situation, all necessary information can be sent automatically at the touch of a single button. The vessel's position can be determined from a GPS navigation receiver connected to the radio or entered manually. Its identity is permanently coded into the radio in the form of the MMSI number. The nature of distress can also be selected by the operator if there is time to do so.

Messages have a common structure, also named call sequence. All the information in a message is coded in 10 bit symbols, where 7 bits code a number in [0,127] range, and the remaining 3 are used as a parity check. In the [8] standard, bits are named B for 0 and Y for 1. In the FSK modulated signal, the higher frequency corresponds to B and the lower frequency corresponds to the Y. All the available bit configurations are represented in Table 2.2. The symbols from 0 to 99 are used to code two decimal figures, the others, from 100 to 127, are used to code service commands.
We are now ready to analyze a call sequence.

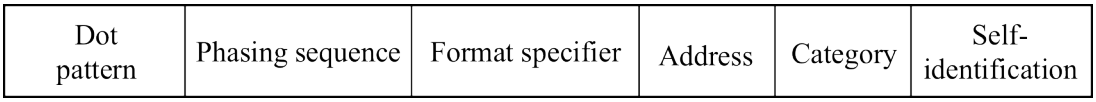


Figure 2.8: Generic DSC message structure

The dot pattern consists of 20 (for VHF or MF/HF acks) or 200 (for MF/HF) alternating bits. It is manly used to detect an incoming message and provides appropriate conditions for earlier bit synchronization.
Since these alternating bits do not allow to identify where is the start bit of a symbol, another synchronization procedure must be used.

The phasing sequence provides the receiver further information to permit correct bit phasing and unambiguous determination of the positions of the characters within a call sequence. Specific phasing characters are sent in the DX and RX position alternatively: 6 DX characters (symbol 125) and a set of decreasing value RX symbols, from 111 to 104 (details at Figure 2.9(b)).
Phasing is considered to be achieved when two DXs and one RX, or two RXs and one DX, or three RXs in the appropriate DX or RX positions, respectively, are successfully received. These three phasing characters may be detected in either consecutive or non-consecutive positions, but in both cases all bits of the phasing sequence should be examined for a correct 3-character pattern. A call should be rejected only if a correct pattern is not found anywhere within the phasing sequence.

The format specifier characters, which are transmitted twice in both the DX and RX positions, must be one of the following seven symbols (from [8]):

- 112 - Distress alert
- 116 - All ships call
- 120 - Selective call to an individual station
- 123 - Selective call to an individual station with automatic service
- 114 - Selective call to a group of ships having a common interest
- 102 - Selective call to a particular geographic area
- 121 - Reserved for non calling purposes

GMDSS telecommunications equipment should not be reserved for emergency use only. The International Maritime Organization encourages mariners to use that equipment for routine as well as safety telecommunications.

For a selective call directed to an individual ship, to a coast station or to a group of stations having a common interest, the address field consists of the characters corresponding to the station's maritime mobile service identity. Distress alerts and "all ships" calls do not have addresses since these calls are implicitly addressed to all stations (ship stations and coast stations).
For a selective call directed to a group of ships in a particular geographic area a numerical geographic coordinates address consisting of ten digits constructed as specified in [8].

The category information defines the degree of priority of the call sequence. For a distress alert the priority is defined by the format specifier and no category information is included in the call sequence. For safety related calls, the category information can specify urgency or safety. For other calls category is routine.

After category, the maritime mobile service identity (MMSI) assigned to the calling station is used for self-identification.

Then, depending on message type, some other fields can be present: nature of distress, distress coordinates, UTC time, type of communication, various telecommands and ship's position. IMO and ITU both require that the DSC-equipped MF/HF and VHF radios be externally connected to a satellite navigation receiver to ensure accurate time and location information are sent to a rescue coordination center if a distress alert is transmitted.

When all the other information is transmitted, the end of sequence (EOS) character closes the message. It is transmitted three times in the DX position and once in the RX position (see Figure 2.9). It must be one of the three unique characters that follows: 117 if the call requires acknowledgment, 122 if the sequence is an answer to a call that requires acknowledgment, and 127 for all other calls.

The error-check character (ECC) is the final character transmitted character and it serves to check the entire sequence for the presence of errors which are undetected by the ten-unit error-detecting code and the time diversity employed. The seven information bits of the ECC shall be equal to the least significant bit of the modulo-2 sums of the corresponding bits of all information characters. The format specifier and the first EOS character are considered to be information characters and must be used for ECC computation.

Automatic acknowledgment transmissions should not start unless the ECC is received and decoded correctly. A received ECC which does not match that calculated from the received information characters may be ignored if this was due to an error detected in the ten-unit error-detecting code of the information characters which was correctable by use of the time diversity code.

Symbol	Bit sequence		Symbol	Bit sequence		Symbol	Bit sequence
00	BBBBBBBYYY		43	YYBYBYBBYY		86	BYYBYBYBYY
01	YBBBBBBYYB		44	BBYYBYBYBB		87	YYYBYBYBYB
02	BYBBBBBYYB		45	YBYYBYBBYY		88	BBBYYBYYBB
03	YYBBBBBYBY		46	BYYYBYBBYY		89	YBBYYBYBYY
04	BBYBBBBYYB		47	YYYYBYBBYB		90	BYBYYBYBYY
05	YBYBBBBYBY		48	BBBBYYBYBY		91	YYBYYBYBYB
06	BYYBBBBYBY		49	YBBBYYBYBB		92	BBYYYBYBYY
07	YYYBBBBYBB		50	BYBBYYBYBB		93	YBYYYBYBYB
08	BBBYBBBYYB		51	YYBBYYBBYY		94	BYYYYBYBYB
09	YBBYBBBYBY		52	BBYBYYBYBB		95	YYYYYBYBBY
10	BYBYBBBYBY		53	YBYBYYBBYY		96	BBBBBYYYBY
11	YYBYBBBYBB		54	BYYBYYBBYY		97	YBBBBYYYBB
12	BBYYBBBYBY		55	YYYBYYBBYB		98	BYBBBYYYBB
13	YBYYBBBYBB		56	BBBYYYBYBB		99	YYBBBYYBYY
14	BYYYBBBYBB		57	YBBYYYBBYY		100	BBYBBYYYBB
15	YYYYBBBBYY		58	BYBYYYBBYY		101	YBYBBYYBYY
16	BBBBYBBYYB		59	YYBYYYBBYB		102	BYYBBYYBYY
17	YBBBYBBYBY		60	BBYYYYBBYY		103	YYYBBYYBYB
18	BYBBYBBYBY		61	YBYYYYBBYB		104	BBBYBYYYBB
19	YYBBYBBYBB		62	BYYYYYBBYB		105	YBBYBYYBYY
20	BBYBYBBYBY		63	YYYYYYBBBY		106	BYBYBYYBYY
21	YBYBYBBYBB		64	BBBBBBYYYB		107	YYBYBYYBYB
22	BYYBYBBYBB		65	YBBBBBYYBY		108	BBYYBYYBYY
23	YYYBYBBBYY		66	BYBBBBYYBY		109	YBYYBYYBYB
24	BBBYYBBYBY		67	YYBBBBYYBB		110	BYYYBYYBYB
25	YBBYYBBYBB		68	BBYBBBYYBY		111	YYYYBYYBBY
26	BYBYYBBYBB		69	YBYBBBYYBB		112	BBBBYYYYBB
27	YYBYYBBBYY		70	BYYBBBYYBB		113	YBBBYYYBYY
28	BBYYYBBYBB		71	YYYBBBYBYY		114	BYBBYYYBYY
29	YBYYYBBBYY		72	BBBYBBYYBY		115	YYBBYYYBYB
30	BYYYYBBBYY		73	YBBYBBYYBB		116	BBYBYYYBYY
31	YYYYYBBBYB		74	BYBYBBYYBB		117	YBYBYYYBYB
32	BBBBBYBYYB		75	YYBYBBYBYY		118	BYYBYYYBYB
33	YBBBBYBYBY		76	BBYYBBYYBB		119	YYYBYYYBBY
34	BYBBBYBYBY		77	YBYYBBYBYY		120	BBBYYYYBYY
35	YYBBBYBYBB		78	BYYYBBYBYY		121	YBBYYYYBYB
36	BBYBBYBYBY		79	YYYYBBYBYB		122	BYBYYYYBYB
37	YBYBBYBYBB		80	BBBBYBYYBY		123	YYBYYYYBBY
38	BYYBBYBYBB		81	YBBBYBYYBB		124	BBYYYYYBYB
39	YYYBBYBBYY		82	BYBBYBYYBB		125	YBYYYYYBBY
40	BBBYBYBYBY		83	YYBBYBYBYY		126	BYYYYYYBBY
41	YBBYBYBYBB		84	BBYBYBYYBB		127	YYYYYYYBBB
42	BYBYBYBYBB		85	YBYBYBYBYY			

Table 2.2: DSC symbol coding

Figure 2.9: DSC message fields (a), symbols in a message (b), consecutive frames (c)

Chapter 3

The VHF monitoring system

This system comes from the demand of controlling maritime communications for safety and security reasons to monitor for emergency transmissions or frequency abuses.
As a second purpose, it can also help the routine procedures at coast stations informing operators of what channels are being used.

Since some channels can be used both in simplex and duplex operation, depending on national regulations, the system should monitor all the allocated spectrum for maritime communications. As illustrated in Table 2.1 at page 40, the frequency shift in duplex channels is 4.6 MHz and the frequency occupation of contiguous channels is 1.45 MHz.
As an approximation, it was initially planned to acquire 8 MHz of bandwidth near the center frequency of the segments and extract the useful part in a second time.
This hypothesis was quickly abandoned after a few practical tests because the transfer rate of the USB 2.0 and the processing power of a common computer system cannot handle in realtime the 32 MByte/s stream of samples produced by the USRP.
Another problem, related to the one before, arises from the TV-RX daughterboard: the total amount of 6.1 MHz of bandwidth, that includes the two channel segments and about 2 MHz of useless spectrum, exceeds the available passing bandwidth of the TV-RX, therefore at least two frontends are required.
To reduce the computational power needed, the smallest sampling frequency that allows

to capture a channel segment, with a little guard interval, has been chosen.
With one daughterboard and a sampling frequency of 1.6 MHz I/Q it is possible to receive one portion of the available channels. The other portion, used only by the coast station to transmit on duplex channels, is left to an optional TV-RX, since it is not really interesting for a coast station to monitor its transmissions. The only valid reason to monitor the high part of VHF spectrum is to receive the two AIS channels present on channels 87 and 88. This task can be accomplished with a modified version of the receiver designed in this thesis.

During the design process of the monitoring system, an interesting subject was the evaluation of how many channels is possible to receive at the same time.
The first option is to monitor the spectrum until a communication appears, then follow it until it finishes and restart the radio scanning. This is the easiest way to proceed and can be accomplished with a normal radio with scanning features and does not really require a software radio.
The opposite approach consists in listening to every frequency and potentially receive every communication on every channel simultaneously. This means that an "unlimited" number of communications can be received at the same time and this can be a good reason for using a software defined radio: with a hardware solution a dedicated radio interface is needed for every channel to be monitored, instead with a software receiver everything can be done with only a radio interface, leading to huge savings in space and money.

Another important feature to include in the monitoring system was the ability of receiving both voice and data. Since the GMDSS uses both MF/HF and VHF for the transportation of its messages, it is useful to include in the system an FSK demodulator for receiving DSC messages and to process emergency requests directly on a computer system. To accomplish this task, a dedicated application receives, demodulates and decodes the messages on data channel 70.

Talking from the GNU Radio perspective, everything mentioned before can be realized with one block that processes all the acquired bandwidth, senses for communications in realtime and makes them available to the users.

Analyzing it with more detail, it is possible to see that a module with 56 outputs cannot be handled easily on a single PC, and a big module that does all by itself is not a scalable solution.
The need to separate tasks started also from the demands of the data channel that are different from the ones of voice channels.

As the processing power required for splitting channels from the initial samples is of moderate magnitude, it should be accomplished by a dedicated machine, further on called server: this machine usually stays in a communication equipment room, close to the antennas and away from the people, and distributes its products via a local network.
The operation of selecting and listening to a voice channel should be done by networked client machines, where a single user can operate independently from others.
With a such structure, the number of operators can be increased or decreased depending on coast station's traffic or particular demands.
Finally, the decoding of DSC messages can be done by clients or by one or more machines for redundancy.
The figure below shows a schematic view of the solution adopted with an extra component, the router, that is actually required when a complex or a large monitoring system have to be deployed.

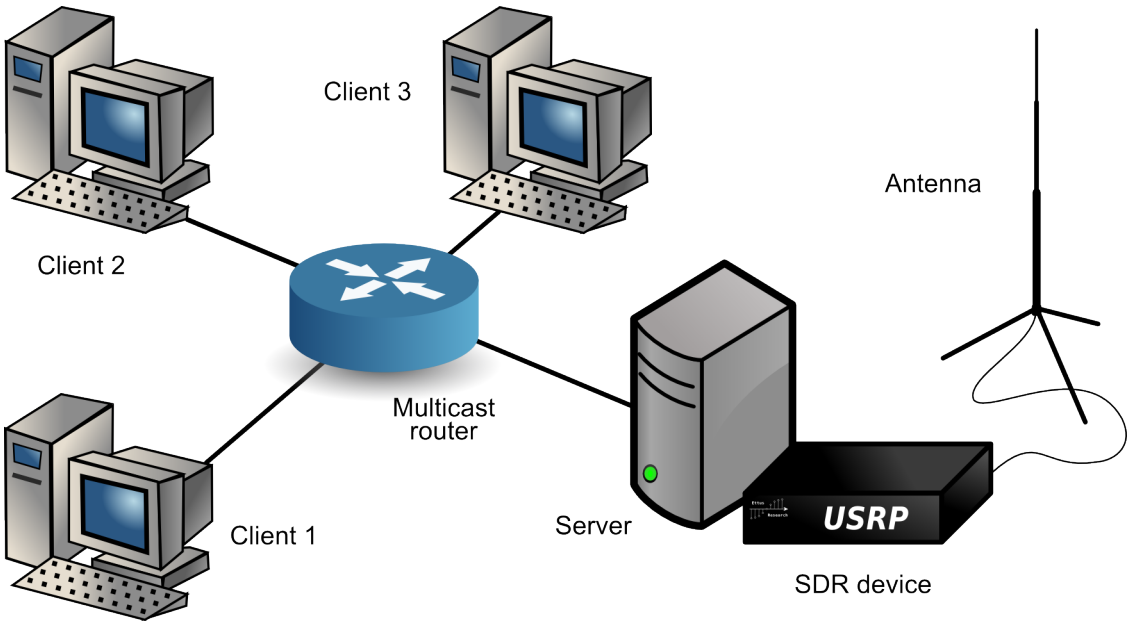


Figure 3.1: Monitoring system architecture

3.1 Multichannel receiver

The server, connected to the USRP, acts as a multichannel receiver. A GNU Radio block has been written to accomplish this task.

The hardware part of the receiver, the USRP and a VHF antenna, is illustrated in the two pictures below (shots taken at Wiser S.r.l.).

In the module, the acquired radio bandwidth is processed channel by channel to recover the original audio signal and make it available to the clients.

The choice of implementing a new standalone module, instead of combining together many existent ones, has been driven by the computational power needed for the operations to be performed. Using many small C++ modules and linking them with python leads to a sub-optimal solution than cannot be acceptable on a realtime system, therefore one standalone module, entirely written in C++, has been written.

In the following sections, every part of the receiver is analyzed, with particular focus on the solutions found to solve some implementation problems.

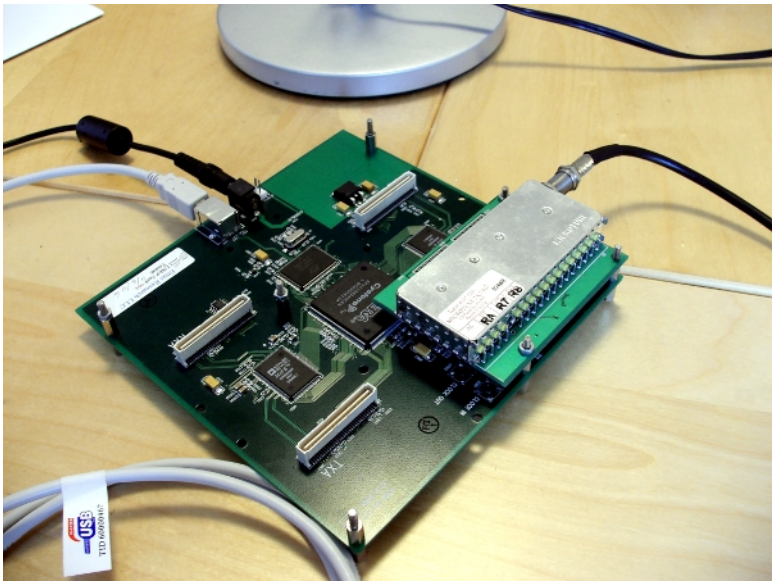


Figure 3.2: The VHF receiving interface (an USRP with TVRX daughterboard)

Figure 3.3: VHF antenna used for tests

3.1.1 Functional blocks

The receiver is composed of many independent blocks, for almost every of them there is a description in the sections below that explains the design choices and the implementation. It is useful to see a block diagram that illustrates the connections between the blocks and how the signal bandwidth decreases.
The input (the arrow on the left) is a complex stream that comes directly from the USRP, that is tuned to the center frequency of maritime VHF channels. This represents the 1.6 MHz portion of radio spectrum that includes the 56 VHF channels.
Next, the whole bandwidth is subdivided in channels and processed. As the power level of a channel is computed, it enters a threshold that is used to trigger the further elaboration of the signal. The FM demodulation is accomplished with a quadrature demodulator followed by the de-emphasis filter and a band pass filter. Then, the audio signal, ranging from 300 to 3000 Hz, is resampled to 24 kHz to reduce network bandwidth requirements and still keep a good audio quality. The last step, as shown on the diagram, consists in sending audio samples to the clients via UDP. An exception to the behavior specified before is the treatment of the DSC modulated channel, that is sent to the network at original 25 kHz rate in complex form and unprocessed. This is required to keep the signal quality at the best.

3.1.2 Bandwidth splitter

The main advantage of using SDR technology is to acquire a large portion of radio spectrum and then extract all the information with software processing. In this case, the receiver have to separate and demodulate maritime VHF channels from a unique sample stream. This task is obtained through a bandwidth splitter that is the hearth of simultaneous multichannel receiption. A common approach to split channels consists in frequency shifting and filtering the input stream a number of times equal to the number of channels to be received.
There are two ways to do this operation, as shown on Figure 3.5. One consists in applying a band pass filter to the initial signal and then move it to the baseband region with a

frequency shift (up or down).

The other, instead, firstly frequency shifts all the spectrum up or down to make the center frequency (or carrier) match the origin of complex frequency plane and then filter the previous signal with a low pass filter that restrict the output bandwidth to a single channel bandwidth.

The first approach, for N channels, requires the design and use of N band pass filters and N frequency shifts, while the second one requires only one low pass and N frequency shifts. However, since the channel occupation is fixed, the N-th channel shift can be obtained as the of result of N sequential channel shifts, requiring the implementation of only one frequency shifter. This solution is really more efficient than the first and has been selected as a starting point. The operation of frequency shifting, in time domain, is done with a simple arithmetic multiplication of the input data by a complex exponential. A GNU Radio block is specifically designed to accomplish this task and other ones implement low pass and band pass filtering with windowed FIR computation.

The first prototype of channel splitter was designed with these components. A few offline tests with 2 MS/s data reached a really poor target: the splitter was able to handle the demodulation of only 5 channels in realtime.

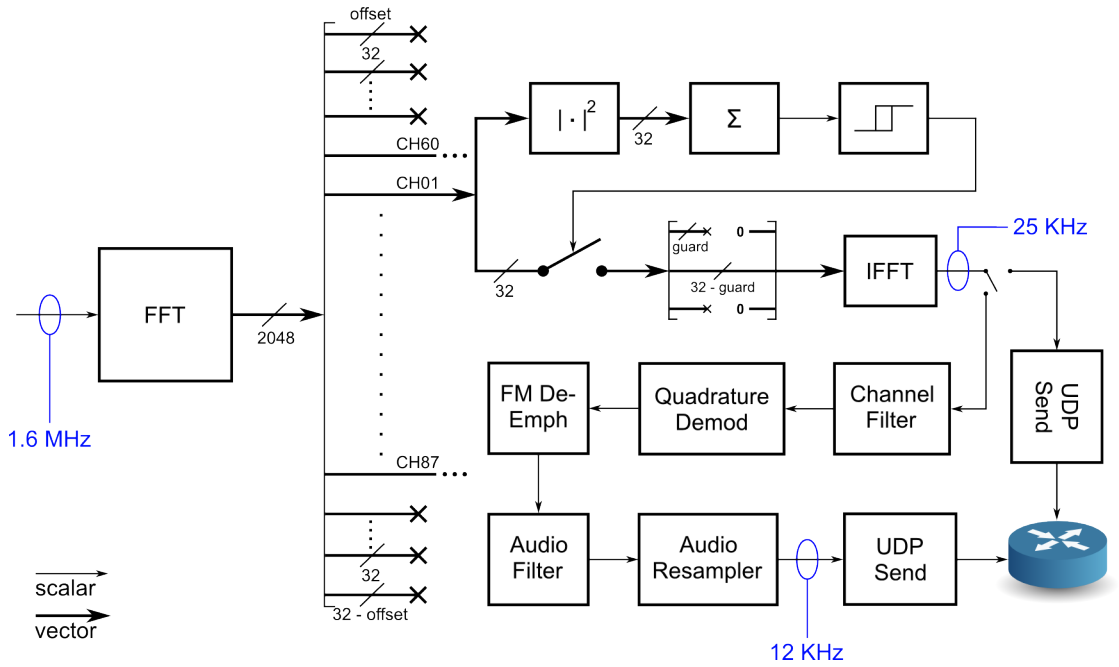


Figure 3.4: Receiver module block diagram

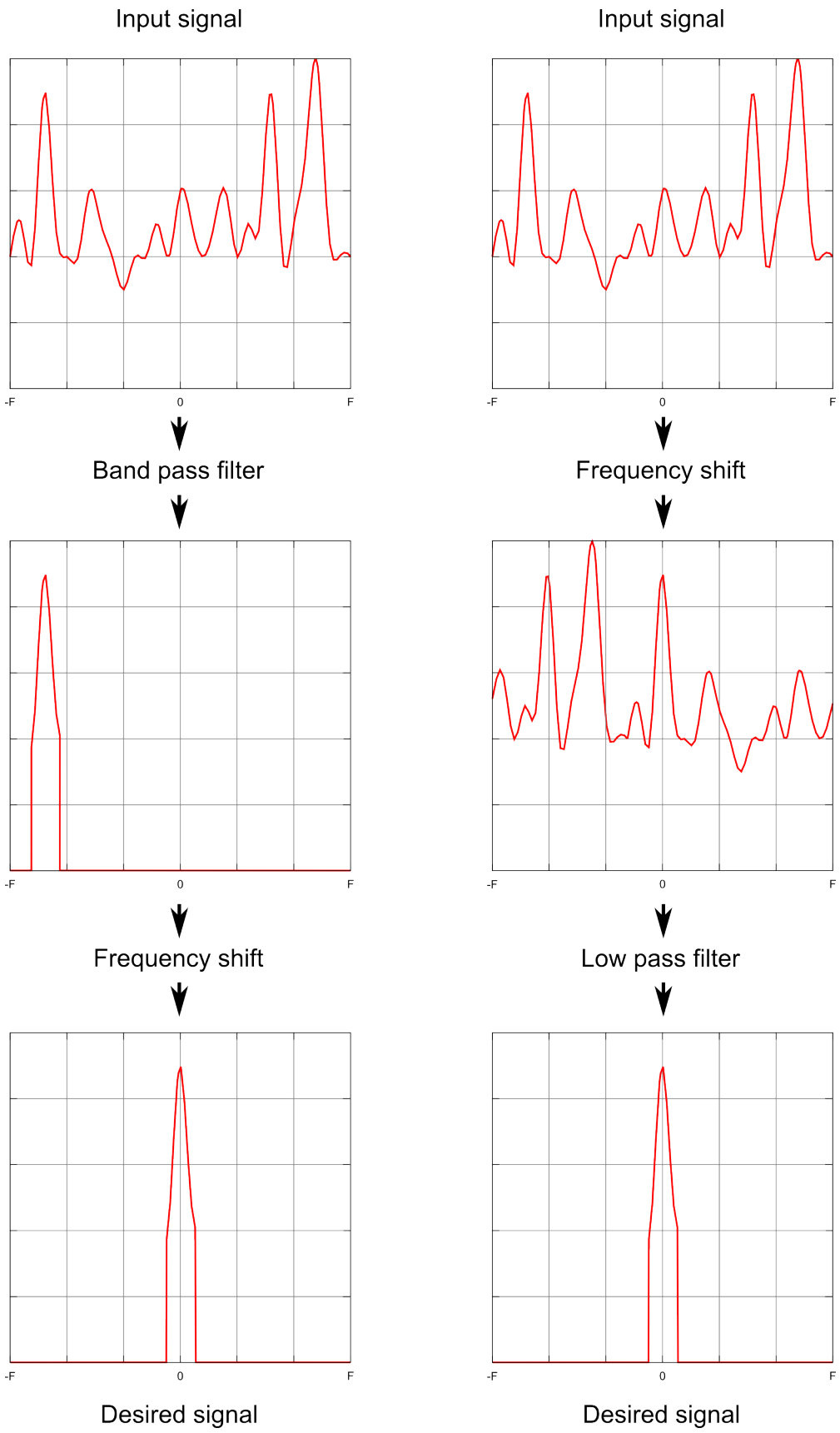


Figure 3.5: Two ways of selecting a channel

The goal of building a receiver for an "unlimited" number of channel seemed to be unreachable. After a little briefing, the idea of using a FFT to separate channels came out.

The main idea is to transform the input data in time domain into its frequency representation, take a group of FFT bins that covers approximately one channel and reconvert it to the time domain with a smaller IFFT. Thus, applying this procedure for every channel will result in our expectations, all available channels will be separated at the same time with two "simple" operations.

One of the first problems was the choice of FFT length. The choice is not easy cause there are two facts to be concerned to: it would be good to have a high time definition, obtained with a short FFT length, to reproduce the original channel signal without sparkling; then it would like to have a high frequency definition, obtained with a large FFT size, to avoid a lost of information in the channel spectrum, where there is a frequency modulated signal. Clearly, these two requirements impose a tradeoff which have to be studied carefully.

A common approach tells to use about 10 to 13 FFT bins for each channel IFFT. In practice this means to consider an IFFT size of the nearest upper power of two (for using the optimized code), in this case 16. Then, considering 56 channels is possible to calculate the required input FFT length, that should be rounded to the upper power of two like the previous one. Therefore $56 * 16 = 840$ which means an FFT size of 1024 samples.

The designed system were operating in realtime and splitting all the 56 channels but the audio quality of demodulated channel was very poor and, most important, the data channel was severely damaged.

Due to these facts the only solution was to double both the input and output FFT sizes. Enlarging input FFT size could lead to some time definition problem that can be subsequently solved with overlapped FFTs. Since the GNU Radio block is called with an arbitrary length of input data and we need to reuse part of the previous FFT execution to take advantage of the overlap technique, a circular input buffer has been implemented. The choice of a circular buffer against a linear buffer is driven by the better efficiency and performance that derives from it: significantly time improvements can help to reach the realtime when working with a high bit rate. The same structure is present in output

processing, where a part of previous computed samples is added to the new ones. This represents the so called overlap-add technique. However, after the code was finally running there were no improvements on audio quality but only more processor load. Subsequently the overlap method was abandoned.

The growing of FFT sizes leads to an increase of arithmetic operations with an order of magnitude of $O(n \cdot log_n)$ considering the best FFT algorithms. This fact should not be neglected when designing a realtime system.
Some debugging code was thus added to the receiver block to monitor the CPU time required for FFT functions. To have an input rate of 2 MS/s (complex samples) and a FFT size of 2048 means that about 976 transforms should be computed in a second. It also means that an FFT bin corresponds to about 976 Hz of the input spectrum. Since the VHF channel bandwidth is 25 kHz and is not an integer multiple of 976.5625, I planned to adjust the input rate to a reasonably good one that makes 25000 divisible by the FFT frequency definition. After some simple computations, I reached the value of 1.6 MS/s that still includes all the 56 channels and leads to a number of 32 bins per channel (exactly the IFFT size) which is the best obtainable result.
Other positive goals have been reached: the bin now corresponds to 781.25 Hz which is smaller than before and means more frequency definition for the transformation, and also the number of FFTs that have to be computed in a second decreased to about 781.
In other terms, we have now 1280 μs to compute an input FFT of 2048 samples and at most 56 output IFFTs of 32 samples. The debugging code for time calculation has shown a good result: on my machine, a Pentium-M 1.86 GHz based laptop with Gentoo Linux and all system and applications optimized for my CPU, the input FFT takes an average of about 30 μs and the output IFFT only 0.3 μs. This result is shown on Figure 3.6 with two density functions of FFT computation times. These short times have been reached on my machine with the latest version (3.2) of FFTW library and are not reachable with the previous one (3.1).

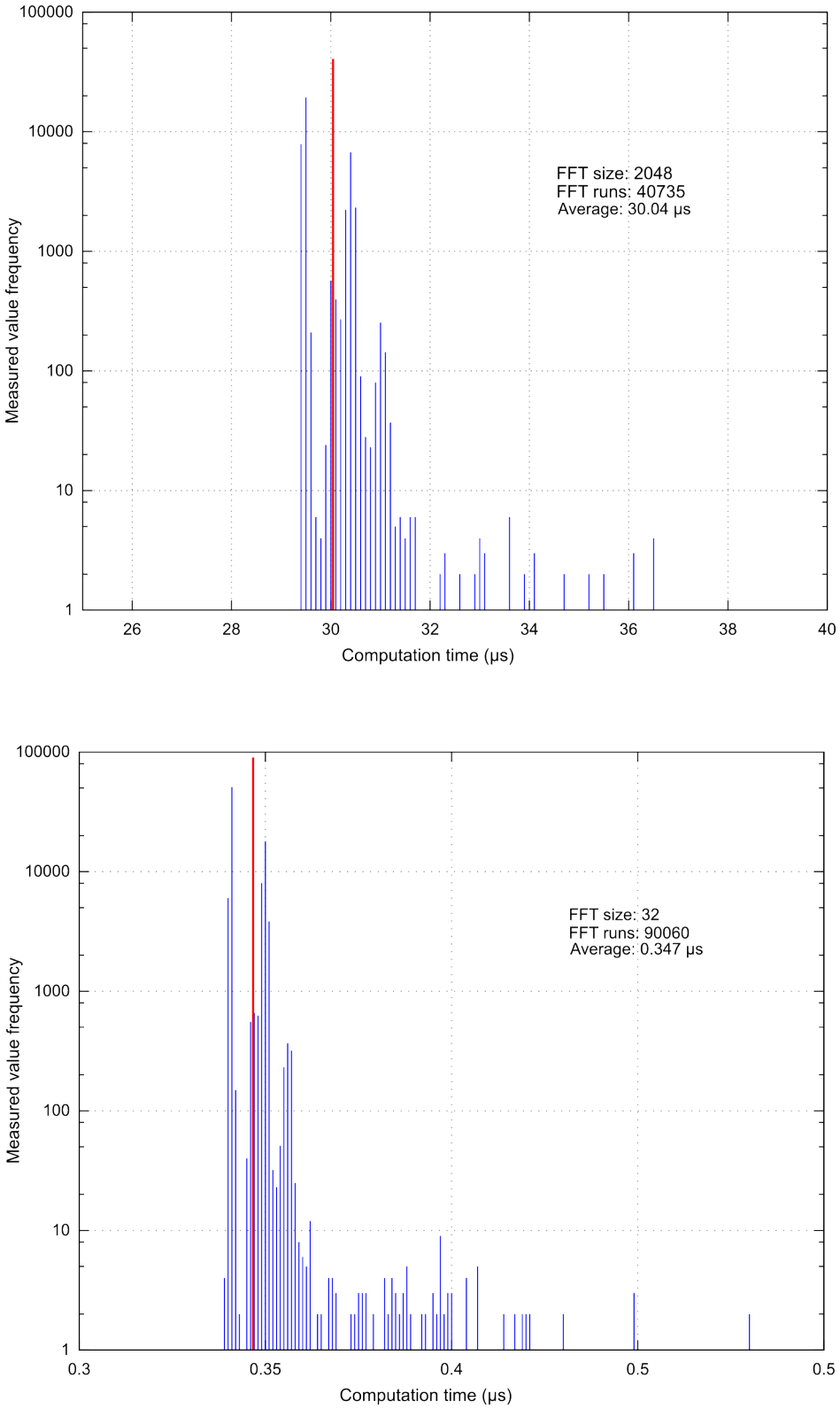


Figure 3.6: Results of FFT computation time tests

3.1.3 Channel sensing

The first operation to be done after obtaining the signal of a single channel is to verify the presence of an active transmission. It must be the first operation simply cause if a user is not transmitting it is useless to proceed with the demodulation of the channel and its diffusion over the network.
This does not includes those particular channels marked as "raw", for instance digital channels, where the modulation cannot be done if synchronization sequences are lost. Hence a "type" flag is used to bypass channel sensing and to allow a direct transmission of channel data trough the network.

A common approach to sense for a transmission is the measurement of signal energy. Taking a sampled signal x[n] and its DFT X[k] is possible to compute the energy and average power of N samples using the following relations (derived from Parseval's theorem):

$$E_x = \sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2 \tag{3.1}$$

$$P_x = \frac{E_x}{N} = \frac{1}{N} \sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N^2} \sum_{k=0}^{N-1} |X[k]|^2 \tag{3.2}$$

The output of a N-sized FFT is a sequence of N complex numbers.
The magnitude of a complex number, expressed in rectangular form is defined as follows:

$$|C| = |a + j \cdot b| = \sqrt{a^2 + b^2} \tag{3.3}$$

Since we need square magnitudes, that represents the Energy Spectral Density (ESD) of the samples, we can exploit the norm() function of the math library:

$$norm(C) = norm(a + j \cdot b) = a^2 + b^2 = (\sqrt{a^2 + b^2})^2 = |C|^2 \tag{3.4}$$

The total energy is then computed as the sum of the norms ignoring the scaling factor N. A little error, derived from some imperfections of the TV-RX frontend, causes an

improper estimation of the energy on the center channel (28). This is probably due to a DC offset entering in the ADC and can be corrected with a reduction of the affected FFT components. As an approximation it is corrected with a division by 10, but in a production environment it should be measured and corrected properly.

A simple threshold on estimated channel energy (or power) can be used to determine if there is activity of the channel. However, temporary bursts of noise can reach the threshold and make the channel to be detected active while it is not. A partial solution can be the rising of the threshold to the maximum value that ensures a correct detection of all important conversations.

Unfortunately, on VHF bands there is enough signal fading to break this assumption.

The simple comparator has no memory and, since it cannot take a decision on past outputs, it can fail with weak signals. A more sophisticated solution has been found.

The solution is a hybrid threshold, between a low pass filter (average) and a Schmidt's trigger (comparator with hysteresis). The low pass filter is required to limit fast changes on power level that typically occurs when there is a channel interference or some missing samples from the USRP. This operation can be tough as a moving average that follows the real power level.

A comparator with hysteresis is then needed to ensure that a temporary signal fade, due to the physical properties of the channel, does not influence the channel state. In particular, an integer counter and three thresholds are used to take a decision: the counter is incremented or decremented depending on the current estimated power level, its value is in the range 0,255; the other two thresholds are used to mark the channel active or inactive basing on the counter value.

These last thresholds are quite asymmetric and must be adjusted to match the particular conditions found where the monitoring system is placed.

To make this task easier, a power logging function is available in the receiver. In debugging options is possible to chose the metric to be logged: the instantaneous power computed after FFT, the value of the counter and the state of the channel. The values logged to file are easily plottable with a graphing software to understand what is happening. A simple multi-function script for GNU Plot is included.

3.1.4 Channel demodulation

Once a communication on a channel is detected it have to be demodulated. Voice channels are processed inside server module, others, marked as data channels, are left untouched. The frequency modulation used in maritime VHF channels is a little different from that one used in broadcasts: modulation index is 5/3, max deviation 5000 Hz, audio bandwidth is filtered above 3 kHz and pre-emphasis filter has a time constant of 530 μs. All these parameters are used to recover the voice properly.
The first step of demodulation process consist in low pass filtering the channel to exclude the noise that surrounds useful FM spectrum. To calculate the filter cut frequency we need the approximated occupation of FM. This is simple to obtain with Carson's rule:

$$B_C = 2 \cdot (\beta + 1) \cdot B_{audio} = 2 \cdot (\Delta f + B_{audio}) = 16kHz \tag{3.5}$$

The 16 kHz value is a delta between the minimum and the maximum frequencies at FM modulator, in other words it means 8 kHz side bands around the carrier frequency.
A 8 kHz Butterworth low pass filter can be easily obtained with a IIR design. After some attempts, an order 32 filter, composed of standard second order cells, has been designed with Matlab help.
The ideal and estimated filter response are illustrated below.

Then, the filtered signal is ready to enter the quadrature demodulator, the base step in FM demodulation. Since it is possible to physically realize it in different ways, but some of them are indicated for hardware implementation only, we analyze two of the most convenient methods to be written in software.
As proposed in [7], the signal recovery can be done computing the first derivative of the phase of received signal. This method comes from the concept of a phase modulation. Frequency modulation can be tough as a modulation of the carrier phase with the time integral of the modulating signal. Starting with the received complex signal,

$$C(t) = x_C(t) + j \cdot x_S(t) = A(t) \cdot e^{j \cdot \theta(t)} \tag{3.6}$$

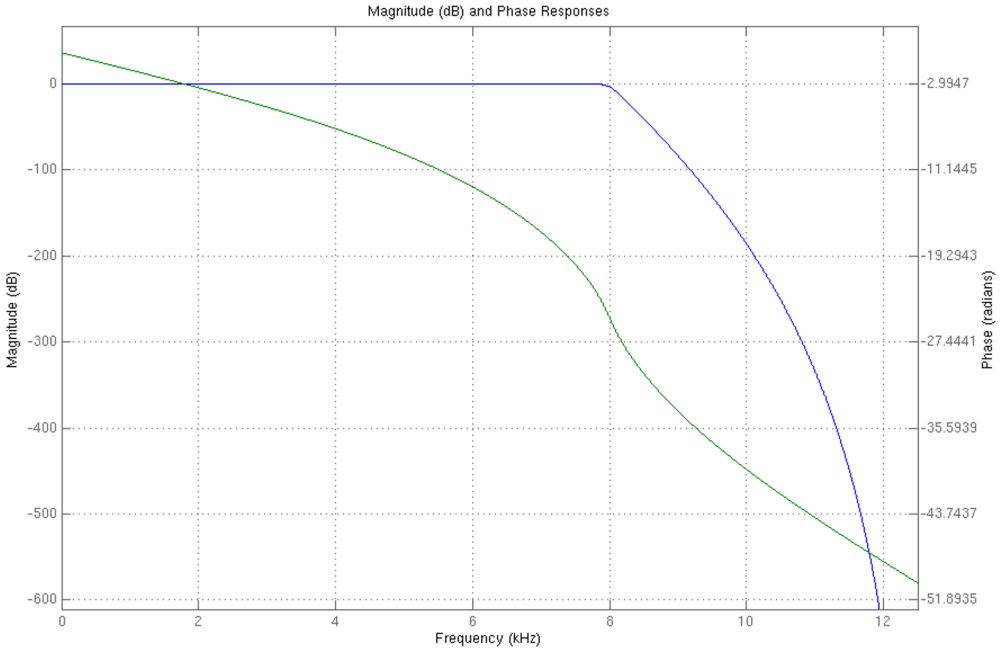


Figure 3.7: Ideal filter response

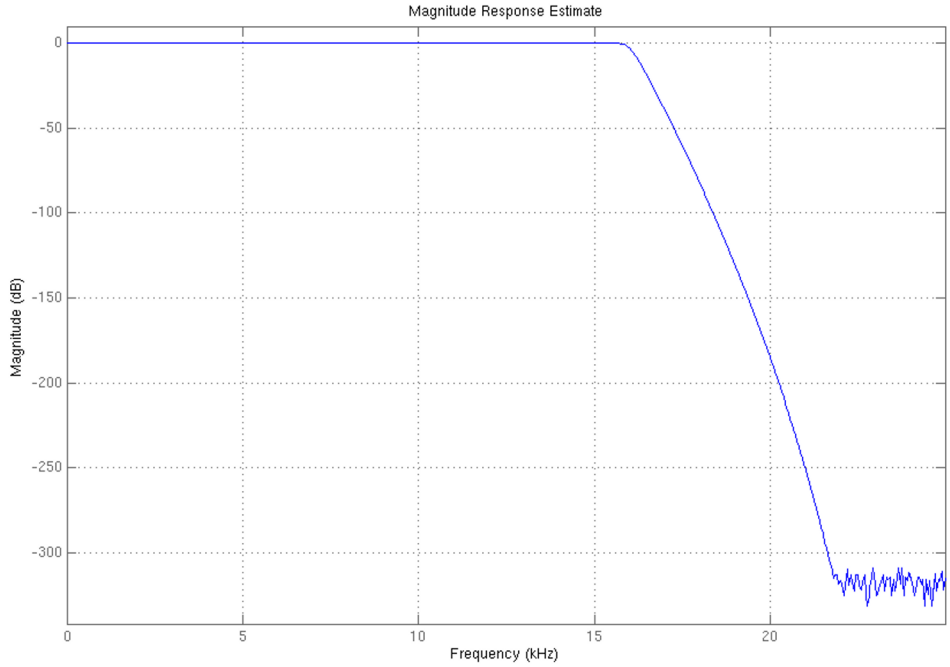


Figure 3.8: Estimated filter response

its instantaneous phase can be written as

$$\theta(t) = \arg \frac{x_S(t)}{x_C(t)} \tag{3.7}$$

and the first derivative of $\theta(t)$ is

$$\theta'(t) = \frac{\partial}{\partial t} \arctan \frac{x_S(t)}{x_C(t)} = \frac{x'_C(t) \cdot x_S(t) - x'_S(t) \cdot x_C(t)}{x_S^2(t) + x_C^2(t)} \tag{3.8}$$

In discrete time domain, a derivative, tough as an incremental ratio, is simply a difference between the current and the previous sample.

This approximation is not always correct. The preliminary hypothesis of having a over-sampled signal, at least of 10 times the Nyquist frequency, must be preserved.

In [7], the sampling frequency for FM data was designed to retain the correctness of derivative approximation. In the actual design, this is not possible in this thesis. In other words, the quadrature demodulator, also called frequency discriminator, cannot be implemented according to this solution.

Fortunately, GNU Radio framework has its own version of quadrature demodulator that has been implemented with another method.

First we compute a complex product of current and previous sample,

$$cp[n] = x[n] \cdot x^*[n-1] \tag{3.9}$$

then, a particular value

$$q[n] = \frac{Fs}{2\pi \cdot \Delta f} \arctan \frac{Im(cp[n])}{Re(cp[n])} \tag{3.10}$$

which is proportional to the modulating signal. The scale factor is not important for voice transmissions and can be adjusted at the latest step.

The obtained signal is ready to enter the FM de-emphasis filter. This operation is the reverse of pre-emphasis and must be done to restore the original relationship between spectral components. Since pre-emphasis is done with a high pass filter with previously specified time constant, we need to design the complementary low pass filter.

Always with IIR technique is possible to implement a first order filter with the required attenuation of 6dB/octave. In this case is possible to use the bilinear transformation to

convert an analog designed filter to the digital domain. The only fault of this procedure is the result: the simple filter with just a pole in the analog domain is converted to a set of coefficients to be used in a direct form type 1 digital filter which need two delay elements instead of one. The more optimized direct form type 2, which has also its problems [5, 1], can be obtained with Matlab designer leading a save of a delay line in implementation. Converting it to code produced a very short function, shown below:

```
inline float deemph(float in, float *z){
 float out=z[0];
 in+=0.927307768331003*z[0];
 z[0]=in;
 return in+out;
}
```

The output signal of de-emphasis filter however has a lot of noise. Especially in bands near DC there are powerful components that disturb the voice listening. Also, at high frequencies there are useless components that can be removed.
To make the sound clear as the one of a real radio, a band pass filter has been designed. Still with the help of Matlab, after some tests a Butterworth order 60 filter has been chosen.
The output sound is very clear and the computation time seems to be acceptable. The only fault, that forced the total bypass of processing for data channels, is the non linear phase of IIR filters: one of the previously named filters can be used on a digital channel channel because they heavily distort the signal. The implementation of audio band pass is similar to the other filters: a chain of second order sections (SOS) produces the required infinite impulse response. A sample SOS is shown below.
All the coefficients are stored in different constant arrays in channel_filter file. For each channel there is a state variable that keeps the required z-states for both FM, de-emphasis and audio filters.

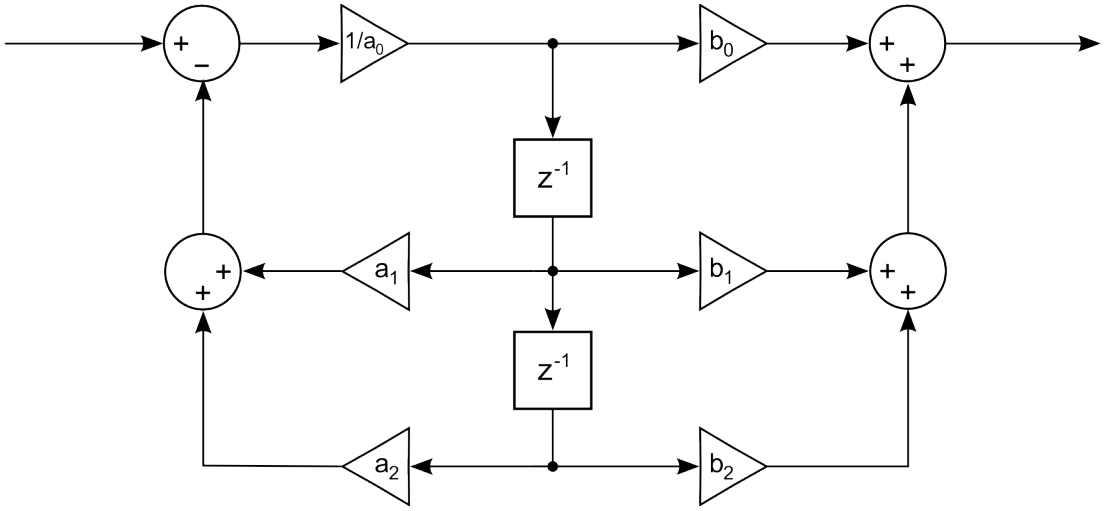


Figure 3.9: A second order section (SOS)

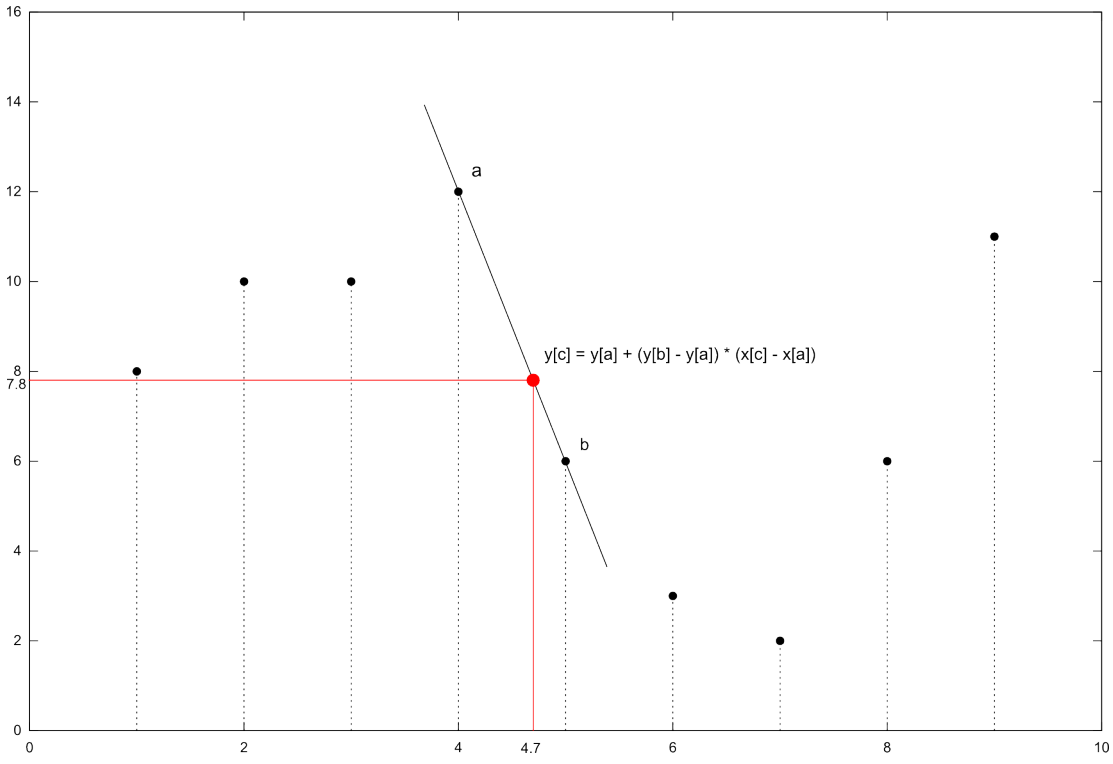


Figure 3.10: Linear interpolation of sample (c) between (a) and (b)

Filter input and output rates are always of 25 kHz. With complex streams this means 25 kHz of available signal bandwidth, while in real streams 12.5 kHz (Nyquist condition).

Since 25 kHz is not a widely used sampling frequency we need to convert it to a proper value. Common sample rates are 8 kHz and its multiples, and 11025 Hz and its multiples.

As a first attempt, I chose 8 kHz that was the minimum rate to allow the correct representation of the 3 kHz filtered audio signal. Another advantage was the very low byte rate requirements to distribute it to the clients. At client side the result was very poor.
After some tests, I found that only with 24 kHz the audio quality reaches a good level.

A simple way to reduce sampling rate consists in downsampling. Sample rate conversion, which includes upsampling and downsampling, is the process of converting a signal from one sampling rate to another trying to keep the information carried by the signal after the conversion.
In practice, every 25 input samples we need to output 24 samples computed with a linear interpolation when needed. Linear interpolation is quite simple, but provides enough precision for our work. An example is shown on Figure 3.10.

A floating point counter is incremented of a downsampling step each time and the output sample is picked from input stream at the counter position. If the counter has a non-integer value, the two adjacent samples are used to interpolate. In some cases, the counter is between 0 and 1 and the interpolation requires the last sample of previous audio block (32 samples per block). This value is stored at every cycle in the channel_info structure.

Every produced new sample is converted from floating point to 16 bit signed integer to save space and to make it easily playable on a PC. Buffer length is of 512 samples and is appropriate for audio applications.
This length, 1024 bytes, is also convenient to fill the payload of a UDP packet without reaching the MTU and not giving much overhead (as it happens for short packets).
When buffer length is reached, all buffered data is sent to clients via UDP, buffer is flushed and if needed audio data or FM samples are logged to file (debug enabled).

3.1.5 Data distribution

As said above, the server application streams audio and statistical data through sockets. There is a logical channel, called statistics channel, that informs the clients of the power

levels of the 56 VHF channels plus some other information (last active channel, sequential number). The other logical channels are associated to each real VHF channel, and are streamed only when the relative channel become active. In these logical channels the raw audio is transmitted in bursts of 512 samples.

Since the channel transmission is one-way and realtime (and can lost some data), the most suitable connection type is UDP, not connection oriented and not lossless. Instead, the use of TCP can be dangerous cause it requires more computational and memory power at the server, and makes useless retransmissions of realtime audio data.

Another opportunity to save network resources comes from IP multicasting. Instead of keeping track of how many clients are listening and send them a copy of audio and statistics data, a single transmission to the multicast group is done. Clients join only the groups they really need (the statistics transmission is mandatory), avoiding the network to be heavily loaded. Each logical channel corresponds to a group that has its own UDP destination port.

This solution can be used both in local area networks, without the need of a router (a switched ethernet for example), and also in large networks (Internet or VPNs), taking advantage of the selective forwarding capabilities of multicast routers.

3.2 Human interface

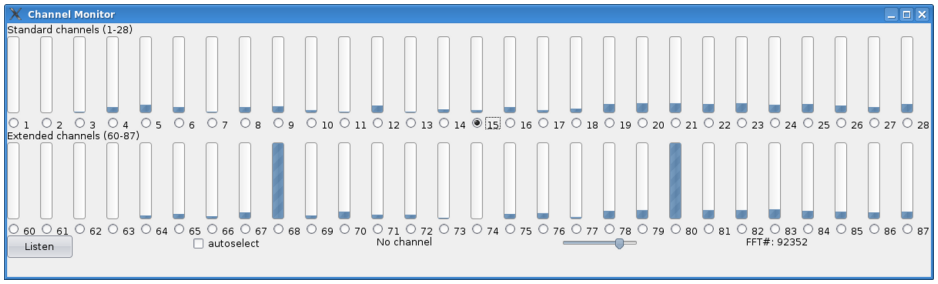


Figure 3.11: Monitoring system user interface

Another important part of the monitoring system is the human interface. Since all other operations seen before are done by a "blackbox" (the server), the work is useless if not combined with a output interface for the user. Its functionality is easy to imagine: receive data from the server and make it available to the user in a such form that is human friendly.

The receiption includes statistical data of the channel, which enables the user to see is there is some activity on channels, and when needed the audio stream of the channel of interest.

All the operations should be available from a graphical user interface that allows a quick and easy selection of what the operator want to listen.

To find and listen immediately new conversations an autoselect function is available.

For convenience the audio is present only when there is an active conversation on channel, otherwise a disturbing noise would be heard.

A mute-like button and a volume selector complete the user controls.

3.2.1 Software portability

One of the driving principles of the human interface implementation is the portability. Starting from the server (receiver) code, it was almost obvious to write the client with same programming language. This idea seemed the best for a programmer with good skills in C and C++ programming, but analyzing the problems in depth, there were a lot of things to keep track to make a program run on different OSes.

The need of some higher level language started when looking to network programming, audio card interfacing and displaying data to the user: the differences between Microsoft and Unix like systems, starting from sockets to graphic libraries can be solved using a lot of #ifdef statements, but surely this is not the best way to proceed.

One of the most famous high level portable language is Python: as mentioned before in the GNU Radio description, this language offers a large variety of libraries and wrappers that allows the interfacing with almost every hardware or software. With just a few rows of code it is possible to open a socket or create a GUI, allowing the programmer to save a lot of time.

Another possible choice was Java, but for my skills it was not really the best solution.

The second big problem of the portability was the choice of the graphic library. Some examples of portable libraries are: GTK, Qt and WxWidgets.

All of them support both C(++) and Python, are open source, and are released with a free license. The main requirements for the human interface are: the ability of show a power meter for a channel, the possibility to visually choose a channel to listen and show some control buttons.

Every library support the requirements, with the exception of vertical gauges that are supported only by WxWidgets. Looking at the appearance, the best can be obtained with Qt (version 4), that allows to completely define the aspect of its widgets with CSS. Another interesting aspect is that Wx libraries does not define a new graphical interface but they use one of the available in the operating system they are running.

After some tests with the libraries, their components and their simplicity of installation, I choose WxWidgets. The develop process is fast, high level abstraction makes possible to define a complete GUI in a few rows and the rest is done by Python. The application look is also nice: in Linux for example, the library relies on GTK and the appearance can be controlled with some GTK plugins. The final problem was the portability of the sound. There are not many libraries that makes a general audio interface for different operating systems. The most famous is PortAudio, that fortunately has a Python wrapper that makes things easier. After some code was written, the whole portability has been tested on Linux and Windows. Everything works fine, some adjustments to receive the multicast streams are needed when firewalls are present.

3.2.2 Operations

Actually, the client support the following operations:

- **Manual channel selection**: once the program is started, the user can view the power level of each channel on the related indicator. Based on his preference, he can listen to the wanted channel simply clicking on the button below the indicator. In this mode, the application will remain on the selected channel until the user changes and the listening process keeps running also if there is no traffic on the channel.

- **Automatic channel selection**: as an alternative to the manual mode, it is possible to let the application chose automatically a channel that where there is some traffic. Activating this mode through clicking the "autoselect" checkbox will make the program to start looking for the activity on channels. When a new channel become active (the power goes beyond a threshold), the user can immediately listen to what is happening.

- **Start/Stop listening**: a required feature is obviously the possibility to start or stop receiving the audio. Normally the program listens to the channel statistics to update the level indicators but does not receive any audio. When the "Start" button is pressed the audio thread receives and process the audio coming from the server. Viceversa, pressing the "Stop" button will stop it. This can be alsa viewed as a "Mute" function.

- **Volume control**: since every radio has a different way to send the voice over the air, it can happen that listening to some channel the audio level differs so much that can disturb the operator. Therefore, it is possible to increase or decrease the listening volume by using the horizontal slider placed below the channel indicators.

- **Communications logging**: the client by default logs everything is listened in a raw file.

3.3 DSC decoder

One component of the monitoring system is dedicated to the handling of DSC messages on VHF channel 70. This is the DSC decoder.

As explained in Chapter 2, DSC messages are transmitted with a 1200 baud binary FSK modulation over FM and are compliant to the ITU-R M.493-11 recommendation.

The messages are sent by mobile or ground stations asynchronously and have different length depending on the information they carry.

To make them robust to noise and channel fading, symbols are transmitted twice, with a 3 bit per symbol parity, and a final error check code at the end of message.

These features are exploited during the synchronization process as will be shown further.

The DSC application, besides the DSC decoder block, is composed of a set of GNU Radio blocks that handles the FM demodulation and resampling. As stated in the receiver section, channel data is distributed via UDP, therefore it must be picked from the network on the client that needs it. Complex samples can be received with a dedicated block, always part of this thesis, called MC receiver. Its output is then fed to a FM demodulation chain that process raw data to extract the audio baseband. After a bandpass filter centered on FSK modulation the signal is resampled to 24000 Hz as required by DSC decoder. This rate is interesting cause it can be further on downsampled of an integer factor 20 to reach the 1200 bps of DSC messages.

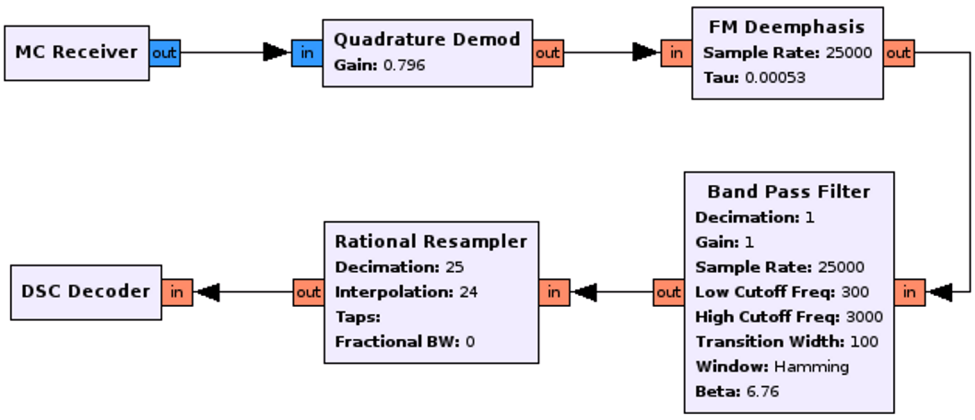


Figure 3.12: Design of DSC application flowgraph with GRC

Each part of the DSC decoder is quite complex and needs to be seen in detail to understand what it really does. Firstly, a DFT-based FSK demodulator process recover bits from the input audio signal, then a cross-correlation based synchronizer assures the optimal sampling timing comparing the analog bit stream with the dot pattern of DSC message (specified in [8]). The produced bit stream enters a buffer where it is collected until it reaches a minimum length threshold. The bits are furthermore analyzed by a message parser to recognize valid messages. When a message is correctly received it is displayed to the user.

3.3.1 FSK demodulator

Like for the server, a preliminary analysis on available modules has been made to avoid to rewrite a module that has already been developed by other people.
Unfortunately, there is a block for modulating a byte stream into a continuous-phase FSK but there is no one for the reverse operation.
As a first attempt the FSK demodulation was realized with GNU Radio available blocks. As illustrated in [11, 13], there are two type of FSK demodulators: one is coherent and another is not. They can be implemented both with a correlator or a matched filter, and differ in error probability. Non coherent design is penalized by the number of components it requires, but however it is the one suggested by [7]. Actually, the non coherent implementation with available blocks works fine but is really slow and cannot be used in a production system. After an attempt to optimize it with handmade blocks, I saw that it was still unusable.
The ultimate breakthrough still came from Fourier transform. In particular in its discrete form (DFT).
In fact, defining a two-bin DFT centered on FSK B and Y carriers, with exactly the decimation factor as size, the demodulation is almost done.
Two constant arrays containing complex exponentials for frequencies f_B and f_Y are separately multiplied with input samples and summed (dot product and sum). The second sum is then subtracted to the first. This process generates an analog signal which represents a distorted version of the original baseband signal. Subsequently, a simple threshold

can recover digital ones and zeros.

What is missing is the choice of the optimal sampling time of received analog signal.

Among various types of synchronizators, the one based on signal correlation seemed good for our purposes. Exploiting DSC message properties, we can correlate the analog signal with an alternating bit pattern, that always comes before real data. Looking to the correlated output is possible to clearly locate peaks that point to message start sequences (see figure below).

In software, this is accomplished with a cycle that finds the absolute maximum of a region and signal to the following blocks that the synchronization is acquired.

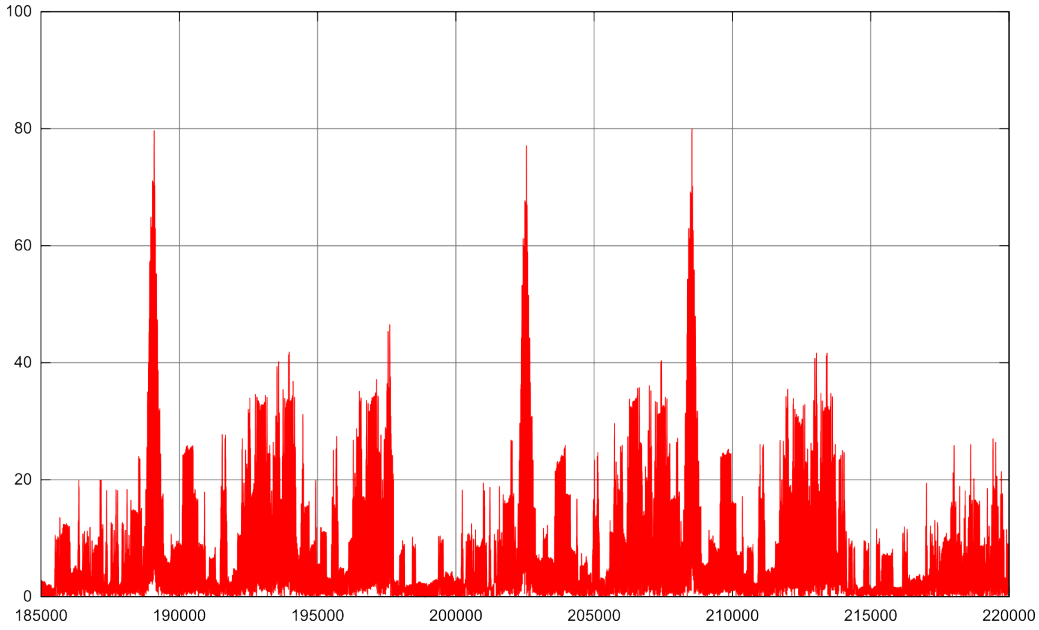


Figure 3.13: Correlation with focus on message peaks

The final problem was the data buffering. To make message synchronization and parsing easier is convenient to have all the samples we need in a buffer, for simplicity a linear buffer.

Buffer length must be computed carefully cause it should be large enough to contain a message but not too large for memory and efficiency constraints. Looking to the correlation output is possible to find the maximum length of a message simply seeing the distance of peaks. A good estimation of message length (in samples) is 11000, however, at this time,

only distress messages have been sampled. Once a peak is found the buffer is aligned to the message and in_sync flag is set. Buffer is then filled to a minimum threshold to ensure that a complete message in present and other operations can start.

3.3.2 Message synchronization

Every message transmission starts with a dot pattern, an alternating bit sequence, of 20 or 200 bits depending on the transmission frequency (MF or VHF). The purpose of this sequence is to allow the receiver to understand that a new message is arriving and synchronize itself with the sender.
To recognize the pattern, a new 10 bit symbol has been added to the table of known ones. This makes easier to find the sequence because a utility function that matches DSC symbols has been written for further message decoding. The searching process follows these two simple steps: read 10 bits from the bit buffer, try to match the dot pattern symbol (number 128 in the table) and, if positive match go to phase sequence verification otherwise skip one bit from the buffer a restart from the beginning. The next step, phase verification, has a role similar to synchronization but its purpose is to prevent false dot pattern matches to be recognized as true message beginnings.

3.3.3 Phase sequence verification

Since a random bit sequence can have enough alternating 0/1 bits, it is mandatory to verify if we are in the beginning of a message. As stated before (Section 2.8), when the dot pattern ends, a sequence of DX and RX symbols should be present to achieve symbol synchronization.
The standard specifies that at least 3 correct phasing symbols must be detected in order to proceed. Therefore a counter of DX and RX good symbols and a state variable keep track of where we are in the message. A sanity check prevents the detection of good RX symbols after a previous RX symbol with lower value has been detected (e.g. a RX6 should not be found after a RX5). Keeping track of the position inside the phasing sequence make

easier to jump to the format symbol when enough correct symbols have been found.

A problem encountered in practical tests was giving a lot of incorrect message interpretations. Looking in detail, I found that the dot pattern followed by the first DX symbol creates a very unlucky bit configuration: dot pattern ends with 0101010101, the following DX is 1011111001, all together 01010101011011111001.
The bit coding of RX2 is 0101011011, that unfortunately is present inside the previous. When the procedure read a correct RX2 expects a DX or RX1 that are never present in this case cause we are out of phase.
To avoid this condition a new check in the sequence verification jumps forward to see if there is a valid format identifier, if not we have found a bogus RX2 that must be skipped.

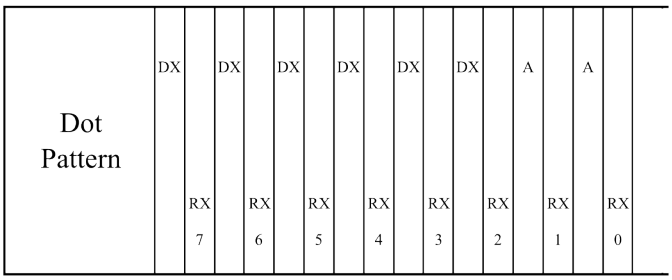


Figure 3.14: Phasing sequence detail

Stepping forward we have another difficult situation to solve. Format symbols are interleaved with RX ones and must be checked in parallel to understand if the message have to be processed futher. In particular, when only one good phasing symbol is recognized before format, there is only one combination of symbols that make the phasing end with success. The last chance is given by RX1 and RX0 (symbols 105 and 104) that are interleaved with the two format symbols. Therefore while checking phase is also required to save the format identifier and check it twice in case of distress message. The format symbol has also to pass another check. It must be a service command (symbol between 100 and 127) and can have only one of the 7 defined values. In case of failure the message is discarded, otherwise is passed to integrity check.

3.3.4 ECC and EOS verification

Different DSC message types have different length. To be sure that all the necessary information to parse the message is present, a simple but powerful check is done.
Every message ends with a EOS character (symbol 127) repeated many times and also has an Error Check Code that ensure more integrity that symbol parity bits.
The EOS search is done symbol by symbol until we reach 127 that cannot be used as a symbol inside the message. During this search we can exploit the fact that every symbol is read to check the parity of them. Also, since the ECC is computed ad a vertical parity we must keep a partial ECC counter that have to be updated every symbol we read. Then when the EOS is found we have precomputed the ECC and checked every message character.
If something fails, we have another opportunity: the RX symbols, where RX stands for retransmission. A copy of the message present in DX position is transmitted with a 5 symbol delay. The same procedure to check EOS and ECC is done with these symbols. If it fails one more time it means that the message is severely errored and should be discarded. Other tries could be done mixing DX and RX character to find a good ECC condition, but this goes beyond the work of this thesis.

3.3.5 Message parsing

The generic message handler, called to make EOS and ECC verification, has the role of selector. For each message type, a dedicated handler is called basing on format identifier. DSC acks and message retransmissions are not handled since with the current architecture is only possible to receive.
The main focus of the message parser is on distress messages. These are the most important for safety reasons and also are the easiest to find for debugging. With debugging I mean that to test the DSC decoder some real radio transmissions are needed. Listening a few times on VHF channel 70 with the monitor software I found only silence. The only way to get a valid DSC was to generate an emergency call with the VHF-DSC transceiver illustrated in Chapter 2 (obviously, without an antenna).

After pushing the distress button, the radio sent 5 DSC messages with the same content. The interpretation of distress messages is fully supported by the parser. Looking to the tables in the ITU standard [8] is possible to understand how the unique address (MMSI) of the transmitting station is coded: 5 symbols, ranging from 0 to 127, are used to carry 10 decimal digits, 9 for the MMSI and trailing 0.

This notation is used also for time coding with a special value of 8888, indicating no time source available. Other interesting fields of the message are the "Nature of distress" which has a fixed number of choices, and the location which helps to find the ship in danger.

Last field has a particular coding, also present in [8]: latitude and longitude derived from a GPS are truncated to degrees and minutes and a symbol is used to identify to which quadrant (NE, NW, SE, SW) they refers. If a position is older than 24 hours or no GPS is available, all the location symbols are set to 99.

All the information decoded is displayed as output and can be logged to file.

Chapter 4

Conclusions

During the develop process of the system, started from previous studies and simulations, the design often evolved in different directions than expected.

The computation power needs, when processing the acquired bandwidth in the time domain, is one of the most critical problems of SDR. Surely, using general purpose processors there are less limitations in memory and storage in respect to an embedded DSP platform, but the processing power to keep operations in realtime leads to many tradeoffs. A low level programming language with processor optimized instructions can mitigate filtering costs, but make the programmer life harder. The solution of using C++ modules linked together with Python of the GNU Radio framework seems a good choice. The available processing blocks are easy to use and good for prototyping but not usable for a production system.

One of the most useful tools, in the SDR domain, is the Fast Fourier Transform. The complexity of shifting and filtering the acquired signal to channelize the spectrum is a good example: the standard way (time domain) requires to compute frequency shifts and low pass filterings for every channel while with FFTs this task is accomplished with only one operation. The same optimization is used for demodulating the FSK signal of DSC: a standard non-coherent demodulator requires synchronized signal sources and many filters that have been simply substituted by a Discrete Fourier Transform.

An interesting update for the monitor system can be the ability to decode AIS (Automatic Identification System) messages. Frequencies used by this system are in the maritime VHF range (near 162 MHz) thus with a second daughterboard centered in the upper channels' region, it is possible to receive the AIS messages.
AIS equipped ships (usually high tonnage ships) send an identification message, containing speed, position and MMSI, about every 10 seconds on a free slot of the AIS channels. Collecting these messages it is possible to locate neighboring ships and in earth stations it can be used for traffic control and navigation aid, and on board to avoid ship collisions. The digital modulation used is GMSK with a maximum bandwidth occupation of 25 kHz and can be collected from the multichannel receiver with just a few modifications.
Creating a decoder as a client of the monitoring system should be a simple task, for example using the GMSK decoder integrated in GNU Radio.
Finally, a new GUI that integrates voice, DSC and AIS can complete and make more effective the monitoring operation.

This work can be also adapted to receive and monitor aircraft VHF communications without changing the main architecture: the frequencies are quite close, so the same daughterboards can be used; the modulation is different (AM instead of FM), but is quite simple to implement; and finally, the ACARS (Aircraft Communication Addressing and Reporting System) messages, similar in concept to the AIS system, are modulated with FSK, the same of maritime DSC.

Another important improvement, as suggested by Elman's products, can be the transformation of the monitoring system in a complete two-way maritime communication system that permits an operator to interact with ships (and not only listen).
Since we already have a client/server architecture, the main idea is to use a VOIP protocol to transfer the information in a synchronized way. Therefore, the required work is to write a channel aggregator that exploits FFT benefits and connect it to a VHF transmitting daughterboard. A suitable protocol for voice transmission from/to clients can be SIP, allowing the communications to be centralized with a software like Asterisk.

As stated, the work of this thesis can be used as a starting point for many future applications: with only a few modifications, it is possible to cover many different systems and tasks... this is the power of Software Defined Radios.

Bibliography

[1] Digital filtering, fft and other stuff. http://www.dsprelated.com.

[2] Ettus research llc. http://www.ettus.com.

[3] Gmdss. http://www.imo.org/TCD/mainframe.asp?topic_id=389.

[4] Gnu radio project. http://gnuradio.org/trac.

[5] Introduction to digital filters. http://www-ccrma.stanford.edu/~jos/filters/.

[6] An overview of the global maritime distress & safety system. http://www.navcen.uscg.gov/marcomms/gmdss/.

[7] Andrea D'Agostino. Studio di fattibilità di un ricevitore multicanale vhf-dsc in tecnologia software radio. Master's thesis, Università di Pisa, April 2008.

[8] ITU-R M.493-11. *Digital selective-calling system for use in the maritime mobile service.*

[9] Naveen Manicka. Gnu radio testbed. Master's thesis, University of Delaware, 2007.

[10] Vincenzo Pellegrini. Soft-dvb: A fully-software dvb-t modulator based on the gnuradio framework. Master's thesis, Università di Pisa, September 2008.

[11] John G. Proakis. *Digital Communications.* McGraw-Hill, 4th edition, 2001.

[12] Danilo Valerio. Open source software-defined radio: A survey on gnuradio and its applications. Technical Report FTW-TR-2008-002, . Forschungszentrum Telekommunikation Wien, August 2008.

[13] Fugin Xiong. *Digital modulation techniques.* Artech House, Inc, 2nd edition, 2006.