gr-isdbt: An ISDB-T 1-segment Receiver Implementation on GNU Radio

Federico Larroca, Pablo Flores Guridi, Gabriel Gómez Sena, Víctor González-Barbone and Pablo Belzarena
Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay.
Email: {flarroca,pablof,ggomez,vagonbar,belza}@fing.edu.uy

***Abstract*—Several countries in the world are undertaking an immensely challenging task: successfully perform the so-called "analogical blackout" of television. In Latin America in particular, the chosen transmission scheme is, in most cases, ISDB-T (the Japanese standard, later adapted by Brazil). Key to the success of this blackout is a thorough understanding of the technology of choice. The present paper intends to be a contribution to this understanding: an open, free and software-based ISDB-T 1-segment receiver. Such receiver may for instance be used to evaluate improvements to the standard (or different algorithms for the receivers), or even be used as a measurement tool (since one has access to the whole receiving chain). In addition to presenting our implementation, we discuss the technology that enables the software-based receiver: Software Defined Radio, and in particular the software framework GNU Radio.**

Keywords*—*Digital television, ISDB, software defined radio.

I. Introduction

The importance of television is undeniable. Most households in the world include (at least) a television set. Although in recent years it has been gradually replaced by other, more interactive, devices, the time spent by the typical "multiscreen user" consumer watching television is 113 minutes a day according to a recent worldwide study [1] (where smartphones, the most important screen medium, is consumed 147 minutes a day).

In this context, several countries have undertaken a very challenging, albeit necessary, task: the so-called "analogical blackout". That is to say, the abandonment of the over fifty years old analogical television transmission scheme, in favor of a digital counterpart. As it usually happens in these situations, different proposals were made and more than one standard is currently in use: the American ATSC (Advanced Television Systems Committee) [2], the Chinese DTMB (Digital Terrestrial Multimedia Broadcast) [3], the European DVB (Digital Video Broadcasting) [4] and the Japanse ISDB (Integrated Services Digital Broadcasting) [5].

Of special interest to us is ISDB. When Brazil was evaluating which Digital Television (DTV) system to adopt, they decided to take the opportunity and perform a Technology Appropriation. To this end, and based on the ISDB standard, they developed the SBTVD (Sistema Brasileiro de Televisão Digital-Terrestre) [6]. The ensuing improvements over ISDB were later incorporated into the original standard, resulting in the so-called "ISDB-T International" (or ISDB-Tb). The latter was then adopted by most South American countries, in addition to some Central American and Asian countries.

This article bears on a further step on this appropriation: the development of an open and free receiver for ISDB-T. Such implementation could for instance, and just to name two examples, be used to perform measurement campaigns (where these measurements may be taken anywhere on the receiving chain) or ease the evaluation of possible improvements to the standard (a hypothetical ISDB-T2) by the scientific community.

A key technology to achieve this goal is Software Defined Radio (SDR). The basic idea is to implement as much as possible of a transmitter and/or receiver in a software which runs on a PC (or an embedded system). Current implementations consist of a variable-frequency oscillator, a mixer and a filter, which together move a portion of the spectrum to baseband, which is then sampled by an Analog-to-Digital converter (the case corresponding to a transmission is analogous). These samples are then fed to a PC, which may process them arbitrarily (the only limitation being mostly the processing power of the PC). The result is that it is possible to have, using the same hardware, from a simple FM radio [7] to a complete GSM base-station [8], simply by running different programmes on the PC.

Several hardware platforms are available on the market. We would like to highlight three: USRP (Universal Software Radio Peripherical, from Ettus Research) [9], BladeRF (from Nuand) [10] and HackRF (from Great Scott Gadgets) [11]. All of them are open to a certain degree, since the drivers used by the PC as well as the code used by the FPGA/CPLD in them are open and freely available. Moreover, the hardware schematics are also available. In any case, there are several models available, from prices ranging typically from 350USD to 1500USD. Regarding the FPGA (or CPLD in the HackRF), it is used to perform the basic baseband processing (such as filtering). However, and as mentioned before, its code may be modified so as to include advanced functionalities, such as FFTs or decoders.

Regarding software, the most prominent development kit (and by the far the most popular) is GNU Radio [12], which in addition to being completely free and open, also supports all the hardware mentioned above. GNU Radio basically provides a framework where the different blocks that compose the transceiver may be implemented and interconnected with relative ease. Moreover, it includes a growing base of already

This work was partially funded by Uruguay's national research agency (ANII) and the telecommunications and audiovisual media services (DINATEL) under grant FST_1_2013_1_13179, and also by "CSIC Grupos I+D 2014" program.

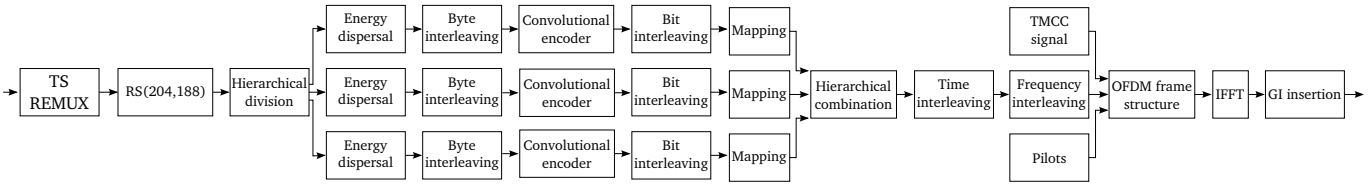

Fig. 1. ISDB-T transmission system block diagram.

implemented blocks, ranging from multipliers to demodulators.

Regarding DTV and SDR, there have been some implementation efforts in the past. In what concerns ISDB-T in particular, and to the best of our knowledge, the only software receiver for ISDB-T was presented in [13]. However, it was not based on GNU Radio, and most importantly, was not released to the public. Moreover, the project seems abandoned, and our attempts to contact the authors have failed. We have thus turned our attention to DVB-T, which is relatively similar to ISDB-T. The first work in this direction was [14]. However, it shares the same flaws as [13]: it was not released to the public (and was not based on GNU Radio). On the other hand, gr-dvbt [15] is an out-of-tree module (a GNU Radio component which is not part of the original source tree, typically developed by third-party programmers) which implements a DVB-T compliant transmitter and receiver, and which is both open and free.

This paper presents gr-isdbt, a GNU Radio out-of-tree module which implements an ISDB-T 1-segment receiver, capable of demodulating the signal and displaying the multimedia content in real-time. Its code is open and free, available at https://github.com/git-artes/gr-isdbt/. The rest of the paper is structured as follows. After a brief overview of ISDB-T in the next section, Sec. III presents in more detail GNU Radio, particularly those components we used for our implementation. Section IV then discusses gr-isdbt, highlighting those sections particular to ISDB-T, where the code of gr-dvbt could not be re-used or had to be heavily modified. Naturally, there are several aspects of the receiver which are not complete or may be improved. A discussion on them is presented in Sec. V. Conclusions in Sec. VI end this paper.

II. The ISDB-T standard

ISDB-T stands for Integrated Services Digital Broadcasting - Terrestrial, and is the Japanese digital television standard. It is based on DVB-T, and was ratified in the early 2000's, after the European and the American standards were adopted. This delay allowed the designers to take into account the experience gained with the previous digital television systems, resulting in the most complex, but also the most robust and versatile standard (although this is obviously the subject of much debate). As we mentioned in the last section, it was later adopted and adapted by Brazil: most importantly a new interactivity middleware named *Ginga* was defined (instead of the original *BML*) and MPEG-4 replaced MPEG-2 for source coding. The new version of the standard was named ISDB-Tb, and was later adopted by most South American countries.

Figure 1 shows the entire block diagram of the ISDB-T transmission system. In what follows we briefly discuss the specifications of each of the blocks, paying special attention to those aspects particular to ISDB-T. For more details, the interested reader should consult [5].

Let us begin by the **modulation scheme**. Orthogonal Frequency Division Multiplexing (OFDM) is used over a 6 MHz bandwidth channel. After the OFDM modulation, a cyclic prefix (CP) is added which length is expressed as a fraction of the active symbol's length, T_s. There are four possible values: $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ or $\frac{1}{32}$. This CP is a copy of the last part of the OFDM symbol which is prepended to it. As we will discuss later, it will be used in reception for symbol synchronization and frequency correction. Moreover, over multipath propagation channels, this prepending will ease equalization (see for instance [16, Sec. 12.4]).

Regarding the number of carriers in one OFDM symbol, it can be either 2^{11}, 2^{12} or 2^{13} (fixed at a power of 2 so as to be able to use the FFT algorithm). However, the sampling rate (termed f_{FFT} in the standard) is always equal to $512/63 \approx 8.126\,\text{MHz}$. This means that keeping the total data rate constant, the operator may choose to use more carriers but slower symbols in order to immunize the radio signal from multipath propagation effects, or less carriers but faster symbols in order to immunize the signal from Doppler effect. This choice is termed **Transmission Mode**, and may be either 1, 2 or 3, although generally mode 3 is used.

Focusing on mode 3 from now on, not all the 8192 carriers are used, but rather 5617, enough to meet the bit-rate and bandwidth requirements (where a guard interval and zero-padding is used in the rest of the carriers). This useful spectrum is in turn sub-divided into 13 sub-bands named **segments**, of 432 carriers each. These 13 segments may be used independently from one another, a feature first implemented in ISDB-T and called *Band-Segmented Transmission OFDM* (BST-OFDM). In particular, in this case, they can be combined in up to three so-called *hierarchical layers* (A, B and C), which transmit different Transport Streams. Moreover, these groups of segments can be configured to use different forward error correction (FEC) rates, time interleaving lengths and modulation schemes.

Although up to three different Transport Streams may be thus transmitted in the same channel, the typical configuration uses one segment with very robust transmission parameters (which for instance allows visualization by mobile users), while the rest is used with a configuration resulting in a high data-rate (with HDTV generally in mind). Thus, handheld receivers (such as cellphones, a market which particularly interested the ISDB-T designers) should only tune and demodulate this single segment, making it possible to work with lower sample and data rates, and thus, less CPU requirements. This feature is known as **1-segment** (or 1-seg for short). Receivers capable of tuning and demodulating all segments

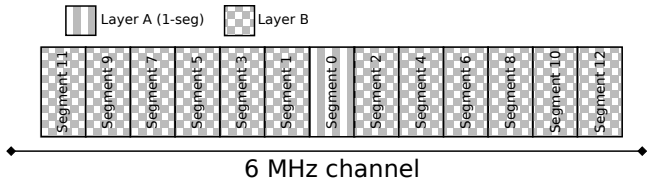

Fig. 2. ISDB-T segmented spectrum and the 1-seg layered configuration.

are known as *full-seg*. Figure 2 illustrates this segment-based spectrum division (and the numbering used by the standard) and the typical 1-seg configuration.

From the 5617 active carriers, there are several which are used as **pilots** to assist the receiver in the equalization process. These so-called scattered pilots (SP) change position from symbol to symbol to avoid pathological situations (such as a permanent deep fade in the spectrum), although their payload is known (which in turn depends on their position). A particularity of ISDB-T is the virtual absence of continual pilots (there is a single CP in the biggest carrier). As we will further discuss later in the article, standard frequency correction algorithms that rely on this kind of pilots are thus not adequate in this case.

In addition to these pilots, several carriers are devoted to transmitting the modulation parameters at use. There are a total of 204 bits to be transmitted: the so-called **TMCC** (Transmission Multiplexing Configuration Control). They occupy fixed carriers in the OFDM symbol, each of them corresponding to the same TMCC's bit, as DBPSK modulation scheme is used. Although they naturally change from symbol to symbol, we will use these carriers with identical information to perform part of the necessary frequency correction as we discuss later. In any case, 204 OFDM symbols are thus required in order to receive the complete TMCC, completing a so-called **OFDM frame**.

The rest of the blocks are somewhat standard: frequency and time interleaving of the complex symbols; mapping, bit interleaving and convolutional encoder are applied to bits; byte interleaver, energy dispersal and Reed-Solomon encoding may be regarded as applied to bytes. It is important to highlight that most of these algorithms are applied separately to each layer (thus the three parallel paths in Fig. 1), and each layer may use its own set of parameters.

Table I summarizes what was described in this section and adds some other extra information. The parameters used in each layer for the convolution encoding, time interleaving and modulation scheme for any particular transmission are all specified in the TMCC.

TABLE I. ISDB-T TRANSMISSION SYSTEM AVAILABLE PARAMETERS.

Parameters	Values
Total Bandwidth	6 MHz
Number of segments	13
Segments bandwidth	$6000/14 \approx 428.57\ kHz$
Number of carriers	1405 (mode 1) 2809 (mode 2) 5617 (mode 3)
Active symbol duration	252 μs (mode 1) 504 μs (mode 2) 1004 μs (mode 3)
Guard interval duration	1/4, 1/8, 1/16, 1/32 (of active symbol duration)
Convolutional Code Rate	1/2, 2/3, 3/4, 5/6, 7/8
Time interleaving parameter	0, 1, 2, 4 (mode 1) 0, 2, 4, 8 (mode 2) 0, 4, 8, 16 (mode 3)
Modulation schemes	DQPSK, QPSK, 16QAM, 64 QAM

III. GNU RADIO

GNU Radio [12] is a free and open-source software development toolkit that provides signal processing blocks to implement software radios. GNU Radio is licensed under the GNU General Public License (GPL) version 3. All of the code is copyright of the Free Software Foundation.

As we discussed before, a software radio is a radio system which performs the required signal processing in software instead of using dedicated integrated circuits in hardware. As the software can be easily replaced in the radio system, the same hardware can be used to create many kinds of radios for many different standards.

Since GNU Radio is software, it can only handle digital data. For a real deployment an external analog RF hardware must be used which can shift back and forth the signal to the desired center frequency and perform the basic filtering. GNU Radio can be used to write applications to either receive and process data from digital streams received by the RF hardware or to push data into digital streams, which are then transmitted using the RF hardware.

GNU Radio is designed to support signal processing on continuous data streams from a source to a sink passing through different signal processing blocks. The stream is a flow of basic types like bytes, integers or complexes. Each GNU Radio block defines input and output signatures which specify the number of input and output streams and their type. The designer can choose which blocks are needed and how they are connected to build a flow graph for a particular implementation. For this purpose, the GNU Radio Companion [17], which is a graphical user interface tool which allows the creation of signal processing applications by combining blocks with a simple drag-and-drop user interface, may be used.

When a flow-graph is executed, the GNU Radio internal scheduler invokes sequentially a predefined method defined in each block. This method processes the data stream according to the block functionality. Communication between blocks is made through shared memory. The size of the shared memory buffers is defined by data type and rate of flow.

A software radio system can be constructed by combining some of the general purpose blocks already provided by GNU Radio and its contributors. Common elements typically found in radio systems are available, for instance: filters, channel codes, synchronization elements, equalizers, demodulators, coders, decoders. Moreover, GNU Radio provides a relatively easy way to extend the functionality by writing the specific blocks needed for a particular implementation. For instance, our ISDB-T 1-seg receiver, as will be explained in Sec. IV, uses some generic blocks like filters or FFT, but we needed to create or adapt some other required blocks such as: OFDM symbol acquisition, frequency synchronization,

channel estimation, TMCC decoding, frequency deinterleaver, time deinterleaver, Viterbi decoder, Reed Solomon decoder, among other auxiliary blocks. Some of them were similar to the ones used in gr-dvbt implementation [15], but others had to be written from scratch.

GNU Radio applications are primarily written using the Python programming language (the GNU Radio Companion is actually a front-end that automatically generates Python code). However, most blocks are written internally in C++: in this sense, Python may be regarded as the scripting language used by GNU Radio. Thus, real-time, high-throughput radio systems can be developed using GNU Radio.

Besides the data flow traversing the graph, normally a communication system needs to signal some events to other blocks in the graph. For instance, one block detects the beginning of a OFDM frame in a radio receiver and other blocks must be notified about this event. Recent versions of GNU Radio provide both a stream tag and a message passing system to handle meta data and control information. The stream tag mechanism allows to add additional information to a particular sample of a flow. On the other hand, the message passing system has two main goals: to allow downstream blocks to communicate back to upstream blocks, and to provide an easier way to communicate between external applications and GNU Radio.

Our implementation uses the stream tag mechanism to signal other blocks the point in the stream where one block detects the beginning or ending of a frame, the need of resynchronization after missing a OFDM symbol, and other similar events. Those requirements could have been implemented by using the message passing system but this option would have involved adding a message queue to all the downstream blocks and it could also imply a performance penalty. Moreover, it is easier to indicate these events with tags, since they are associated to particular samples.

Another interesting feature useful for a real time system, where performance is an essential aspect, is the recent Volk Library [18]. The so-called *Vector-Optimized Library of Kernels* (VOLK) project from GNU Radio provides a simple to use, extensible, and architecture-independent programming tool to enable vectorized mathematical operations. The Volk library makes extensive use of the SIMD (Single instruction, multiple data) capabilities provided by moderns CPUs. The SIMD enabled processors have multiple processing elements that perform the same operation on multiple data points simultaneously. Thus, such machines exploit data level parallelism which is particularly applicable for processing vectors of data. For instance, using Volk two arrays of the same length may be multiplied entry-wise (and the result stored) much faster than using the standard `for` loop.

Some of the blocks needed by our ISDB-T 1-seg receiver frequently need to apply the same operation to a data vector, thus the use of the Volk library provides a great performance improvement. To verify this performance improvement, we tested our OFDM Symbol Acquisition block. The time spent on the block for processing a data stream was measured, obtaining a system 54.9% faster with the use of the Volk library.

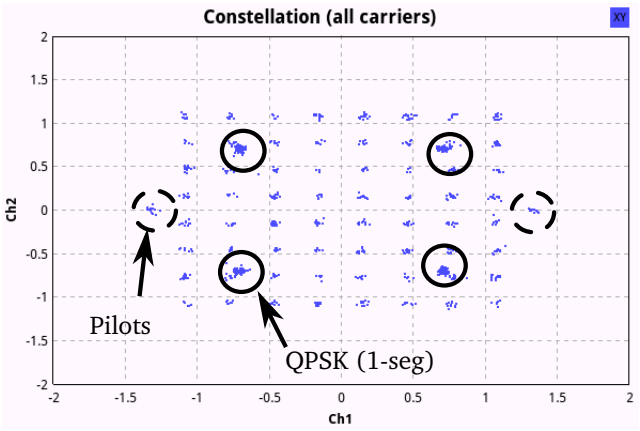

Fig. 4. An example of the received complex symbols. Please note that some of the points corresponding to the 64QAM are very near to those corresponding to the QPSK and were also encircled in the figure.

IV. A 1-SEG RECEIVER IMPLEMENTED IN GNU RADIO

Now that we have introduced both the ISDB-T standard and GNU Radio, let us discuss our implementation. The 1-seg receiver's block diagram is shown in Fig. 3 (a screen capture of the corresponding GNU Radio companion's flow graph). As we mentioned before, it takes advantage of some general purpose blocks such as the `Low Pass Filter`, the `FFT` and the `Vector to Stream`, but most of them had to be implemented or adapted by us. We first give an overview of the complete flow graph, and then focus on the most interesting blocks and their operation.

The first step is naturally to receive the data from the corresponding hardware (in this particular case a USRP). As these devices usually cannot sample at any arbitrary rate, a rational resampler is used to obtain a sampling rate of f_{FFT} (cf. Sec. II). Depending on the situation, it may be necessary to filter-out neighboring channels. For this purpose, a `Low Pass Filter` block with a cut-off frequency $f_c = 2.8\,MHz$ may be used. After that, the `OFDM Sym Acquisition` block performs symbol time synchronization and pre-FFT carrier frequency offset correction. Once this was performed, as in every OFDM system, the FFT of the incoming vector has to be calculated. The `Sync And Channel Estimation` block is then in charge of performing the post-FFT carrier frequency offset correction, channel estimation and equalization. Finally, the `TMCC Decoding` block will detect the end of every OFDM frame (the grouping of 204 OFDM symbols that transmit a complete TMCC) and perform the TMCC decoding.

It is important to highlight that until this point, our receiver operates as a full-seg one. That is to say, we are sampling the whole 13 segments and performing synchronization and the rest of the tasks for the complete signal. Figure 4 shows an example where the complex symbols (after synchronization and equalization) corresponding to all segments are displayed by our receiver. There are two layers in this case: the one corresponding to 1-seg (modulated in QPSK) and another made up of the other 12 segments (modulated in 64QAM). Pilots, which are modulated in BPSK (and boosted), may also be appreciated in the figure.

The next block (`Subset of carriers`) outputs only

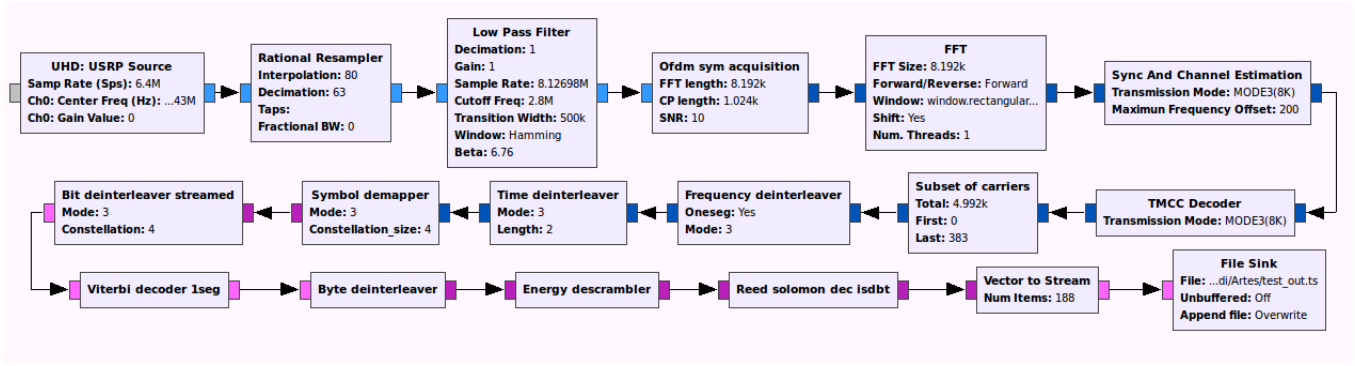

Fig. 3. The GNU Radio Companion flow graph of our ISDB-T 1-seg receiver implementation.

those carriers corresponding to the 1-seg signal. It may thus be regarded as a hierarchical divisor. A complete full-seg receiver would roughly include three parallel branches of the blocks that follow: various deinterleavers, decoders and demapper. Finally, when the processed bytes arrive to the `File Sink` block, the result is a Transport Stream file. It may either be saved and played (or analyzed) off-line, or be played in real-time by first creating the file as a pipe, running the flow graph, and feeding it to, for instance, MPlayer [19].

Some of the most interesting blocks are discussed in detail in the following sections.

A. `Ofdm sym acquisition`

When the receiver is processing the samples one by one, two uncertainties appear: the arrival time of the OFDM symbol (θ) and the carrier frequency offset (ϵ), which due to a difference between the local oscillators in the transmitter and the receiver may cause a shift of all the sub-carriers. These two uncertainties lead us to model the received signal $r[k]$ as

$$r[k] = s[k-\theta]e^{j2\pi\epsilon k/N} + n[k],$$

with $s[k]$ the transmitted signal and $n[k]$ the AWGN added by the channel.

A classic and relatively simple algorithm that estimates both θ and ϵ was proposed by van de Beek et al. in [20]. This algorithm considers an observation interval of $2N+L$ consecutive samples, with N the number of samples in one OFDM symbol ($N = 2^{13}$ in our mode 3 case) and L the number of samples of the cyclic prefix. Then, two sets are considered:

$$\mathcal{I} = \{\theta, \ldots, \theta + L - 1\},$$
$$\mathcal{I}' = \{\theta + N, \ldots, \theta + N + L - 1\}.$$

So, $\mathcal{I}$ indicates the sample indices corresponding to the cyclic prefix, and the set $\mathcal{I}'$ those corresponding to the original data that is copied in the CP. Naturally, there is a strong correlation between these samples:

$$\forall k \in \mathcal{I} : E\{r[k]r^*[k+m]\} = \begin{cases} \sigma_s^2 + \sigma_n^2 & m = 0, \\ \sigma_s^2 e^{-j2\pi\epsilon} & m = N, \\ 0 & \text{otherwise.} \end{cases}$$

The remaining samples $r[k]$ such that $k \notin \mathcal{I} \cup \mathcal{I}'$ are mutually uncorrelated.

Based on this observation, [20] proposes the following log-likelihood function for θ and ϵ

$$\Lambda(\sigma, \epsilon) = |\gamma(\theta)| cos(2\pi\epsilon + \angle\gamma(\theta)) - \rho\Phi(\theta) \quad (1)$$

where $\angle$ denotes the phase of the complex number,

$$\gamma(m) = \sum_{k=m}^{m+L-1} r[k]r^*[k+N],$$
$$\Phi(m) = \sum_{k=m}^{m+L-1} \frac{|r[k]|^2 + |r[k+N]|^2}{2},$$

and $\rho = \sigma_s^2/(\sigma_s^2 + \sigma_n^2) = SNR/(1+SNR)$.

Finally, the maximization of Eq. (1) in the observation interval will solve the receiver uncertainties. In particular, a two-stage optimization may be performed, where ϵ is optimized first as a function of θ:

$$\hat{\epsilon}_{ML} = -\frac{1}{2\pi}\angle\gamma(\hat{\theta}_{ML}), \quad (2)$$

and the resulting θ is thus

$$\hat{\theta}_{ML} = \arg\max_{\theta} |\gamma(\theta)| - \rho\Phi(\theta). \quad (3)$$

Please note that owing to the periodicity of the cos function, there are actually infinitely many maxima of ϵ. We have assumed here that $0 < \epsilon < 1$. This is thus called *fractional frequency* correction. The integer part will be corrected in a different block.

Our implementation works as follows. When synchronization has not yet been attained (e.g. the first time the flow graph is run), Eq. (3) is calculated for the whole observation period, its maximum is found, ϵ is calculated through (2), samples are derotated accordingly (as we mentioned before, the performance for these operations was remarkably improved by Volk), and the samples corresponding to the actual symbol are output.

Once synchronization is established, the position of the maximum should not change significantly from symbol to symbol (as well as the value of (3)). We thus only calculate (3) for a small interval around the previous θ, and proceed accordingly. If this is not the case (e.g. the obtained maximum is too small when compared to previous values), a re-synchronization is triggered. We first look for a reasonable maximum in the

complete observation period. If it is not found, we skip some samples and proceed as if synchronization was not attained.

Notice that at this point of the receiver we still do not know the Transmission Mode or the CP length that are being used, i.e. N and L in the above algorithm. Strictly speaking, the maximum likelihood estimator has to be calculated for every Transmission Mode and for every cyclic prefix length, being the one generating the greater value and a perfect peak in $\Lambda(\sigma, \epsilon)$ the correct configuration. Naturally, since these values do not change frequently (if at all), they may be found previously, and the block configured accordingly

B. `Sync and channel estimation`

For large carrier frequency offsets, we can distinguish between an "integer" carrier frequency offset Δf_I, being a multiple of the sub-carrier spacing $1/T_s$, and the "fractional" carrier frequency offset ϵ we calculated before. The latter, if left unattended would provoke Inter Carrier Interference. The former, the one addressed by the present block, represents an ambiguity regarding which of the 2^{13} carriers output by the FFT block correspond to each of the transmitted ones, as they are now shifted by the unknown Δf_I.

In other broadcast technologies such as DVB-T, the presence of continual pilots (certain fixed carriers which transmit an alternating complex symbol) makes it possible to correlate two consecutive OFDM symbols in order to solve this problem. However, as we mentioned before, in this case that would not be possible as ISDB-T has no continual pilots. On the other hand, we may take advantage of the multiple TMCC carriers that are constantly transmitted at fixed frequency positions. Although we still do not know what they are transmitting (i.e. time interleaving parameter, convolutional code rate, modulation scheme), in a given OFDM symbol they are all transmitting the same bit.

Suppose there are M TMCC carriers per OFDM symbol ($M = 52$ for mode 3), and denote as $T[i] : i \in \{0, \ldots, M-1\}$ their positions when there is no frequency offset (i.e. at transmission), and $Y[k]$ the output of the FFT block. Although the bits transmitted at the carriers corresponding to the TMCC are the same, the modulation used is DBPSK, where the initial value for the differential modulation depends on $T[i]$. This means that, depending on their position, certain symbols are either 4/3 or -4/3. Let us then further denote as $w(T[i])$ this initial value. It should be clear that the following correlation is maximized when $m = \Delta f_I$:

$$\Gamma[m] = \sum_{i=0}^{M-2} w(T[i])Y[T[i]+m].w(T[i+1])Y^*[T[i+1]+m] \tag{4}$$

Once we know which carrier is each, it is time for channel estimation and subsequent equalization. That is to say, if $X[i]$ is the symbol transmitted at the i-th carrier, then $Y[i] = X[i]H[i]$, where $H[i]$ is the channel gain on the corresponding frequency [16, Ch. 12]. We thus have to estimate $H[i]$ for every carrier and divide accordingly.

This task shall be performed by using the scattered pilots (SP), as they have predefined positions and values. Nevertheless, these positions vary with every new symbol cyclically. In particular, there are four different possible carrier configurations. A very similar algorithm to the one discussed before regarding continual pilots is applied to the SPs in order to detect the current arrangement. Then, $H[i]$ is calculated for the carriers corresponding to the SPs. The value for the rest is estimated by a simple linear interpolation.

C. `TMCC decoder`

At this point, once we have correctly synchronized and equalized the signal, it is time to decode the TMCC in order to set the appropriate values for channel decoding and symbol demapping blocks. We proceed as follows. For any given OFDM symbol, we demodulate the corresponding bit on every carrier, and perform a majority voting to take the final decision of the corresponding bit.

Moreover, in order to decide whether a complete TMCC was received, we can make use of TMCC synchronization signal which consists in a 16-bit word that takes either the form 0011010111101110 or 1100101000010001 depending whether it is an odd or even frame. A buffer with the last 204 bits is constantly filled and bits 1 to 16 are compared to the synchronization words presented above. When matched, the BCH parity code syndrome is computed and if no errors where found, a new frame start is detected and the TMCC can be read. This is signaled downstream by a tag, since other blocks make use of this information (most notably the energy dispersal, which should be reset with every new OFDM frame).

Finally, the `TMCC decoder` block is also responsible for filtering all the control pilots and letting out only the data carriers reordered increasingly by segments number.

D. Further blocks: demapping, interleaving and decoding

The rest of the blocks are relatively standard, so their implementation did not cause major problems. In Fig. 3, after the `TMCC Decoder` block discussed in the previous section, the `Subset of Carriers` block receives every data carrier but only lets out those corresponding to the 1-segment transmission. For those carriers, frequency and time deinterleaving are performed and then symbol demapping followed by bit deinterleaving. A QPSK constellation is assumed in the last two blocks as that is the recommended modulation scheme for handheld receivers.

After the Viterbi decoder there is a byte deinterleaving and an energy descrambler. This last stage may be used to test the correct reception of the signal. As specified in [5, Sec. 3.5], the byte just before the beginning of a new OFDM frame (signaled by the tag we mentioned before) should be the synchronization byte used by the Transport Stream (i.e. 0x47). After that, the Reed Solomon decoder, which is able to correct up to eight corrupted bytes, removes the last 16 bytes of redundancy from every Transport Stream package. Finally, the stream is saved on the hard drive. Both the Viterbi and Reed-Solomon decoder were taken from [15] as they are identical to those used by DVB-T.

Figure 5 shows the full implementation receiving a real digital terrestrial television transmission and displaying it online.

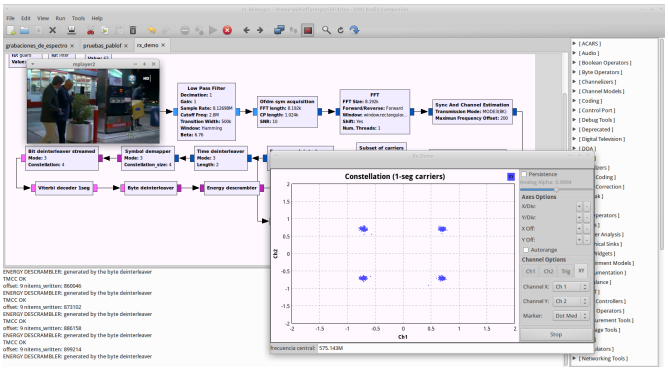

Fig. 5. Our 1-segment ISDB-T receiver working in real time. Note how the corresponding video and constellation are being displayed.

V. Further Developments

The last section presented our current implementation of an ISDB-T receiver on GNU Radio. Naturally, there are several blocks which may be improved. For instance, regarding synchronization, we have used the classic algorithm by van de Beek et al. [20] to perform coarse time and frequency synchronization. However, there have been several proposals since (see for instance [21]), and an evaluation of possible substitutes is in order. For instance, a further estimation of the residual timing and frequency error may be obtained from the post-FFT stage (termed "fine synchronization"), which we have not implemented. The same applies for other blocks, such as the integer frequency correction or the channel equalization (see for instance [22] for a recent survey on the latter subject). It is important to highlight however, that complexity is a crucial factor here, and that any such substitute should not increase it significantly. Another aspect we would like to highlight is that the modular architecture of the receiver (inherited from GNU Radio) makes these substitutions and evaluations a relatively simple task.

An important feature, which we are currently working on, is the ability to decode and display the rest of the segments of the ISDB-T signal. As we explained before, most of the operations (such as time deinterleaving) are performed per segment. Those blocks are thus already capable of handling the rest of the segments with minor modifications. The main challenge with the full-segment receiver is the computing power required to process a bit-stream with a significantly higher rate than the one considered up to now. We are currently evaluating the performance of each of the blocks in order to optimize the most critical ones.

Please note however that for low-end PCs an off-line mode is always possible (either for 1-seg or full-seg), where the signal is recorded, then processed, and finally visualized. For the cases where this is not acceptable, we are working on implementing the first blocks of the receiver on the hardware's FPGA. For instance, the OFDM synchronization block is computationally intensive, and off-loading it to the FPGA would alleviate the PC's CPU. Naturally, in this scenario we would lose in simplicity and generality (the FPGA code would be valid only for certain SDR hardwares), but a high-end PC would not be necessary for online operation of the receiver.

One of the applications that we mentioned in the introduction, in addition to evaluating other algorithms for the different blocks, was the possibility of taking measurements anywhere on the receiving chain. Indeed, there are several measurement equipment specially designed for digital TV, but their prices range roughly from 5.000 USD to 100.000 USD. Just to cite an example which considers only software, Max-Eye Technologies' DVB-T/H signal analysis and monitoring toolkit, a third-party add-on to National Instrument's LabView, which may be used together with the USRP, has a listing price of 6.000 USD [23]. In addition to cost, these measurement equipments present the negativity of being closed. This means that in case of a doubt, the user counts only with the technical support and/or the manual. Moreover, adding features (such as another type of measurement, or a different measurement technique) will imply even more cost (or will simply not be possible).

Given the context above, we are also working on a free and open GNU Radio out-of-tree module for digital modulation measurements, and its integration with gr-isdbt. The first blocks are in beta stage, and may be downloaded from https://github.com/git-artes/gr-mer.

VI. Conclusions

We have presented gr-isdbt, the first fully software-based, free and open ISDB-T 1-segment receiver. We have discussed the particularities of ISDB-T with respect to DVB-T, specially those that represent the biggest challenge to the receiver (such as the absence of Continual Pilots). Although, as discussed in the previous section, there is plenty of room for improvement, the performance of the current implementation is such that the whole signal may be demodulated, transformed into the original MPEG Transport Stream and visualized on real-time.

We consider this work to be a further step in the technological appropriation that the region is performing regarding digital television. We believe that implementations based on the free and open software paradigm are key to an effective appropriation. Moreover, in our particular case, the Software Defined Radio technology plays a fundamental role, assuring generality (by means of frameworks such as GNU Radio) and ease of distribution (with the exception of the general-purpose hardware, the different applications are simply downloaded from the Internet).

References

[1] MillwardBrown, "AdReaction: Marketing in a Multiscreen world. Global report." 2014. [Online]. Available: https://www.millwardbrown.com/adreaction/2014/report/Millward-Brown_AdReaction-2014_Global.pdf

[2] "A/53: Atsc digital television standard." [Online]. Available: http://atsc.org/standard/a53-atsc-digital-television-standard/

[3] "Framing Structure, Channel Coding and Modulation for Digital Television Terrestrial Broadcasting System," Chinese National Standard GB 20600-2006, in Chinese, 2006.

[4] "EN 300 744 V1.6.1. Framing structure, channel coding and modulation for digital terrestrial television. ," 2009. [Online]. Available: www.etsi.org/deliver/etsi_en/300700_300799/300744/01.06.01_60/en_300744v010601p.pdf

[5] "STD-B31. Transmission System for Digital Terrestrial Television Broadcasting." 2014. [Online]. Available: www.arib.or.jp/english/html/overview/doc/6-STD-B31v2_2-E1.pdf

[6] "Normas Brasileiras de TV Digital." [Online]. Available: http://forumsbtvd.org.br/acervo-online/normas-brasileiras-de-tv-digital/

[7] Ettus Research, "How To Build an FM Receiver with the USRP in Less Than 10 minutes." [Online]. Available: http://www.ettus.com/kb/detail/sdr-for-beginners-building-an-fm-receiver-with-the-usrp-and-gnu-radio

[8] Range Networks, "OpenBTS." [Online]. Available: http://openbts.org/

[9] Ettus Research, A National Instruments Company, "Universal Software Radio Peripherical." [Online]. Available: http://www.ettus.com/

[10] Nuand LLC., "bladeRF." [Online]. Available: http://nuand.com/

[11] Great Scott Gadgets, "HackRF." [Online]. Available: http://greatscottgadgets.com/hackrf/

[12] "GNU Radio. The free & open software radio ecosystem." [Online]. Available: http://gnuradio.org

[13] H. Sugano, R. Miyamoto, and M. Okada, "Fully software-based real-time isdb-t 1 segment receiver," in *Broadband Multimedia Systems and Broadcasting (BMSB), 2011 IEEE International Symposium on*, June 2011, pp. 1–5.

[14] V. Pellegrini, M. Di Dio, L. Rose, and M. Luise, "A real-time, fully-software receiver for dvb-t signals based on the usrp," in *6th Karlsruhe Workshop on Software Radios*, 2010.

[15] B. Diaconescu, "gr-dvbt." [Online]. Available: https://github.com/BogdanDIA/gr-dvbt

[16] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[17] "GNU Radio Companion." [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio/wiki/GNURadioCompanion

[18] T. Rondeau, N. McCarthy, and T. O'Shea, "SIMD Programming in GNU Radio: Maintainable und User-Friendly Algorithm Optimization with VOLK," in *2013 Wireless Innovation Forum European Conference on Communications Technologies and Software Defined Radio (SDR'13 - WInnComm - Europe)*. [Online]. Available: https://gnuradio.org/redmine/attachments/download/422/volk.pdf

[19] "MPlayer." [Online]. Available: http://mplayerhq.hu

[20] J.-J. van de Beek, M. Sandell, and P. Borjesson, "ML estimation of time and frequency offset in OFDM systems," *Signal Processing, IEEE Transactions on*, vol. 45, no. 7, pp. 1800–1805, Jul 1997.

[21] M. Morelli, C.-C. Kuo, and M.-O. Pun, "Synchronization Techniques for Orthogonal Frequency Division Multiple Access (OFDMA): A Tutorial Review," *Proceedings of the IEEE*, vol. 95, no. 7, pp. 1394–1427, July 2007.

[22] Y. Liu, Z. Tan, H. Hu, L. Cimini, and G. Li, "Channel estimation for ofdm," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 4, pp. 1891–1908, Fourthquarter 2014.

[23] National Instruments, "DVB-T/H Signal Analysis and Monitoring Toolkit - MaxEye Technologies." [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/en/nid/212594