GNU Radio, an educational tool to teach synchronization and much more
Decoding RDS signal from FM radio

Thomas Lavarenne

Lycée Jean Rostand - Villepinte

29 Mars 2023
Summary

1. FM Radio
2. RDS signal extraction
3. Viewing and slicing frames
4. Decoding and displaying informations
Thomas Lavarenne
(thomas.lavarenne@lycee-jean-rostand.fr)

Teacher of applied physics in BTS ("Brevet de technicien supérieur")
Thomas Lavarenne  
(thomas.lavarenne@lycee-jean-rostand.fr)

Teacher of applied physics in BTS ("Brevet de technicien supérieur")

Equivalent to the second year of a bachelor’s degree
**Presentation**

**THE FRENCH DEGREE SYSTEM:**

<table>
<thead>
<tr>
<th>LEVELS</th>
<th>DEGREES</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 years</td>
<td>2 semesters (1 year for a total of 6 years of postsecondary study)</td>
</tr>
<tr>
<td>8 years</td>
<td>6 semesters (1 year for a total of 7 years of postsecondary study)</td>
</tr>
<tr>
<td>7 years</td>
<td>6 semesters (1 year for a total of 8 years of postsecondary study)</td>
</tr>
<tr>
<td>6 years</td>
<td>2 semesters (1 year for a total of 9 years of postsecondary study)</td>
</tr>
<tr>
<td>5 years</td>
<td>Master (4 years)</td>
</tr>
<tr>
<td>4 years</td>
<td>Licence (3 years)</td>
</tr>
<tr>
<td>3 years</td>
<td>Licence professionnelle (professional bachelor)</td>
</tr>
<tr>
<td>2 years</td>
<td>BTS (4 semesters)</td>
</tr>
<tr>
<td>1 year</td>
<td>Licence (bachelor)</td>
</tr>
</tbody>
</table>

**DOCTORATE**

- State diploma of Doctor of medicine
- Doctoral degree awarded by doctoral boards affiliated with France’s universities to successful candidates who have previously earned a French master’s degree or the equivalent

**Bachelor (Licence)** > **Master** > **Doctorate**

---

**Thomas Lavarenne**
(thomas.lavarenne@lycee-jean-rostand.fr)

Teacher of applied physics in BTS ("Brevet de technicien supérieur")

Equivalent to the second year of a bachelor’s degree

Students mainly study computer science, from the physical layer to application creation.
Why I use GNU Radio?

GNU Radio fits particularly well for the study of digital wireless communications.
Why I use GNU Radio?

GNU Radio fits particularly well for the study of digital wireless communications

Many do not have the mathematical experience needed in signal processing
Why I use GNU Radio?

GNU Radio fits particularly well for the study of digital wireless communications

Many do not have the mathematical experience needed in signal processing

GNU radio is rather simple handling and it is quickly possible to do very interesting things
Why I use GNU Radio?

GNU Radio fits particularly well for the study of digital wireless communications

Many do not have the mathematical experience needed in signal processing

GNU radio is rather simple handling and it is quickly possible to do very interesting things

Practical sessions that allow you to work on a lot of interesting skills (building an antenna, FM analog demodulation, filtering, synchronization, phase modulation...etc.)
Why I use GNU Radio?

GNU Radio fits particularly well for the study of digital wireless communications

Many do not have the mathematical experience needed in signal processing

GNU radio is rather simple handling and it is quickly possible to do very interesting things

Practical sessions that allow you to work on a lot of interesting skills (building an antenna, FM analog demodulation, filtering, synchronization, phase modulation...etc.)

Isolate difficulties
1. FM Radio

Each FM station consists of a multiplex containing:

- Mono informations in baseband
- A pilot at 19 kHz
- Stereo information around a 38 kHz sub-carrier
- The RDS signal around the subcarrier at 57 kHz
1. FM Radio

Each FM station consists of a multiplex containing:

- Mono informations in baseband
1. FM Radio

Each FM station consists of a multiplex containing:

- Mono informations in baseband
- a pilot at 19 kHz
1. FM Radio

Each FM station consists of a multiplex containing:

- Mono informations in baseband
- a pilot at 19 kHz
- Stereo information around a 38 kHz sub-carrier
1. FM Radio

Each FM station consists of a multiplex containing:

- Mono informations in baseband
- a pilot at 19 kHz
- Stereo information around a 38 kHz sub-carrier
- the RDS signal around the subcarrier at 57 kHz
Listen to the radio:

Options
- Output Language: Python
- Generate Options: QT GUI

Variable
- ID: samp_rate
- Value: 2.1168M

Throttle
- Sample Rate: 2.1168M

File Source
- File: fichier
- Repeat: Yes
- Add begin tag: 0
- Offset: 0
- Length: 0

WBFM Receive
- Quadrature Rate: 2.1168M
- Audio Decimation: 2

Low Pass Filter
- Decimation: 6
- Gain: 1
- Sample Rate: 2.1168M
- Cutoff Freq: 100k
- Transition Width: 100k
- Window: Hamming
- Beta: 6.76

LT GUI Waterfall Sink
- FFT Size: 1024
- Center Frequency (Hz): 0
- Bandwidth (Hz): 176.4k

Low Pass Filter
- Decimation: 4
- Gain: 1
- Sample Rate: 176.4k
- Cutoff Freq: 15k
- Transition Width: 1k
- Window: Hamming
- Beta: 6.76

Audio Sink
- Sample Rate: 48 kHz
2. RDS signal extraction

Filter around 57 kHz and get the modulated RDS signal:
BPSK Modulation?

Superimposing a reference signal at 57kHz with the signal source block:
Transfer the signal to baseband: frequency translation

- Frequency Xlating FIR Filter
  - Decimation: 40
  - Taps: firdes.low_pass(40, 0.5, 0.1)
  - Center Frequency: 56.9974kHz
  - Sample Rate: 1.0584MHz

- AGC
  - Rate: 100u
  - Reference: 1
  - Gain: 1
  - Max Gain: 65.536kHz

- Complex to Arg
- Symbol Sync
  - Timing Error Detector: Mueller and Müller
  - Samples per Symbol: 11.1411
  - Expected TED Gain: 1
  - Loop Bandwidth: 45MHz
  - Damping Factor: 1
  - Maximum Deviation: 1.5
  - Output Samples/Symbol: 1
  - Interpolating Resampler: MMSE, 8 tap FIR

- QT GUI Time Sink
  - Number of Points: 200
  - Sample Rate: 26.46kHz
  - Autoscale: No

- QT GUI Constellation Sink
  - Number of Points: 1.024K
  - Autoscale: No
Transfer the signal to baseband: frequency translation

- Extract the phase with the **Complex to Arg** block

---

**Frequency Xlating FIR Filter**
- Decimation: 40
- Taps: firdes.low_pass(40, s...)
- Center Frequency: 56.9974k
- Sample Rate: 1.0584M

**Complex to Arg**
- Input: in
- Output: out

**AGC**
- Rate: 100us
- Reference: 1
- Gain: 1
- Max Gain: 65.536k

**Symbol Sync**
- Timing Error Detector: Mueller and Müller
- Samples per Symbol: 11.1411
- Expected TED Gain: 1
- Loop Bandwidth: 45m
- Damping Factor: 1
- Maximum Deviation: 1.5
- Output Samples/Symbol: 1
- Interpolating Resampler: MMSE, 8 tap FIR

**QT GUI Time Sink**
- Number of Points: 200
- Sample Rate: 26.46k
- Autoscale: No

**QT GUI Constellation Sink**
- Number of Points: 1.024k
- Autoscale: No

---

TL (LJR)
Transfer the signal to baseband: frequency translation

- Extract the phase with the **Complex to Arg** block
- Plot the constellation after symbol synchronization
GNU Radio, an educational tool to teach synchronization and much more.

Symbol Sync
- Timing Error Detector: Mueller and Müller
- Samples per Symbol: 11.1411
- Expected TED Gain: 1
- Loop Bandwidth: 45m
- Damping Factor: 1
- Maximum Deviation: 1.5
- Output Samples/Symbol: 1
- Interpolating Resampler: MMSE, 8 tap FIR

QT GUI Constellation Sink
- Number of Points: 1.024k
- Autoscale: No
Conclusion:

- Manual synchronization has only an educational purpose
Conclusion:

- Manual synchronization has only an educational purpose
- We use the Costas Loop block
Conclusion:

- Manual synchronization has only an educational purpose
- We use the Costas Loop block
- Illustrate and understand notions of synchronization without going into great mathematical details
3. Viewing and slicing Frames

To retrieve and visualize the frames, first we add a threshold and try to synchronize the data stream on the station code, in this example it is "France bleu Alsace" whose code is F405.
GNU Radio is open source!
Everything is on github:
Results:

Various frames are detected.

The 'packet' tag added by the correlate-access code block permit to synchronize the visualization on the Time Sink block.
To process the frames, we choose to cut them and send them to separate files on the hard disk for a post-processing in python.
To process the frames, we choose to cut them and send them to separate files on the hard disk for a post-processing in python.

Tagged File sink block works with a 'burst' tag:
To process the frames, we choose to cut them and send them to separate files on the hard disk for a post-processing in python.

Tagged File sink block works with a ‘burst’ tag:

if it detects a ‘burst’ tag with PMT ‘True’, it opens a new file and saves all the data until it detects a new ‘burst’ tag with PMT ‘False’.
To process the frames, we choose to cut them and send them to separate files on the hard disk for a post-processing in python.

**Tagged File sink block** works with a 'burst' tag:

if it detects a 'burst' tag with **PMT 'True'**, it opens a new file and saves all the data until it detects a new 'burst' tag with **PMT 'False'**.

Adding the 'burst' tags is the role of the **RDS Packet divider**.
RDS Packet divider is a Python block:

class blk(gr.sync_block): # other base classes are basic_block, decim_block, interp_block
    
    """Embedded Python Block that put a tag with key burst and pmt:True when a tag is recieved and key burst and pmt:False 178 samples after that""

    def __init__(self, tag_name="packet"): # only default arguments here
        
        """arguments to this function show up as parameters in GRC""
        gr.sync_block.__init__(
            self,
            name='RDS packet divider', # will show up in GRC
            in_sig=[np.int8],
            out_sig=[np.int8]
        )
        
        # if an attribute with the same name as a parameter is found,
        # a callback is registered (properties work, too).
        self.tag_name = tag_name
        self.max_block_length = 178

    def work(self, input_items, output_items):
        """example: multiply with constant""
        output_items[0][:] = input_items[0]
        tags = self.get_tags_in_window(0, 0, len(input_items[0]))

        for i in range(len(tags)):
            
            print("We got a tag: ", tags[i].key, tags[i].value, tags[i].offset)
            #if tags[i].key == self.tag_name:
            self.add_item_tag(0, tags[i].offset, pmt.intern("burst"), pmt.PMT_T,
            pmt.intern("Blc"))
            self.add_item_tag(0, tags[i].offset + self.max_block_length, pmt.intern("burst"),
            pmt.PMT_F, pmt.intern("Blc"))

            return len(output_items[0])
More details on tags and how to build such a python block?
https://www.youtube.com/watch?v=j4Cn8U2Kl90
Decoding in python:

```python
i=0
while i<=len(byte)-1:
    if (byte[i] != byte[i+1])
        trame_dec += '0'
    else:
        trame_dec += '1'
```

No major difficulties!

- Open the files
4. Decoding and displaying Information

Decoding in python:

```python
i=0
while i< len(byte)-1:
    if (byte[i] != byte[i+1])
        trame_dec += '0'
    else:
        trame_dec += '1'
```

No major difficulties!

- Open the files
- Read the bytes
Decoding in python:

```python
i=0
while i<=len(byte)-1:
    if (byte[i] != byte[i+1])
        trame_dec += '0'
    else:
        trame_dec += '1'
```

No major difficulties!
- Open the files
- Read the bytes
- Read the doc...
What is hidden in our files?

```python
import os
import time

Chemin_home = '/home/lavarenn/GNURadiodaysParis2023'
Chemin_dossier = '/home/lavarenn/Trames_RDS'

nom = [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
texte = []

for i in range(0, 65):
    texte.append(' ')

texte = '
while(True):
    time.sleep(0.2)

    fichierhome = []
    for files in os.listdir(Chemin_home):
        fichierhome.append(files)
    listehome = []
    for j in range(0, len(fichierhome)):
        listehome += str(fichierhome[j])

    # déplacer les fichiers trames dans le dossier de travail
    if ('.dat' in listehome) == True:
        os.system('mv {0}/file* {1}/.format(Chemin_home, Chemin_dossier))

    # lecture du nom des fichiers et insertion dans la liste
    for fich in os.listdir(Chemin_dossier):
        fichdict ={}
        for files in os.listdir(Chemin_dossier):
            fichdict.append(files)

    # préambule: access code France Musique F203
    001100110101011010101101011011011000
    # préambule: access code France Inter F201
    0011001101010110101011010110110110
```

---

TL (LJR)
GNU Radio, an educational tool to teach synchronization
29 Mars 2023
20 / 22
RDS and FM Radio is a little old but still in use and interesting
Conclusion

- RDS and FM Radio is a little old but still in use and interesting
- Many notions of physics, signal processing and computer science in a concrete way
Conclusion

- RDS and FM Radio is a little old but still in use and interesting
- Many notions of physics, signal processing and computer science in a concrete way
- Working with real signals is an important source of motivation
Thank you

- Thank you to Jean-Michel Friedt with whom I have been talking a lot about this ...
Thank you to Jean-Michel Friedt with whom I have been talking a lot about this ...

Thank you to Cyrille Morin, for the help on the tutorial on tags and the creation of the python block.
Thank you to Jean-Michel Friedt with whom I have been talking a lot about this ...

Thank you to Cyrille Morin, for the help on the tutorial on tags and the creation of the python block.

Thank you to all the developers and contributors of GNU Radio !!
Thank you to Jean-Michel Friedt with whom I have been talking a lot about this ...

Thank you to Cyrille Morin, for the help on the tutorial on tags and the creation of the python block.

Thank you to all the developers and contributors of GNU Radio !!

Thank you all for your attention!
Thank you

- Thank you to Jean-Michel Friedt with whom I have been talking a lot about this ...
- Thank you to Cyrille Morin, for the help on the tutorial on tags and the creation of the python block.
- Thank you to all the developers and contributors of GNU Radio !!
- See you at the coffee break!