Frame Decoding

The final step in the receiver is the decoding. It is performed in multiple sub-steps, as follows. Demodulation: The OFDM Decode MAC block receives vectors of 48 constellation points in the complex plane, corresponding to the 48 data subcarriers per OFDM symbol. According to the used modulation scheme, these constellations are mapped to floating point values, representing the soft-bits of the employed modulation. Deinterleaving: Dependent on the Modulation and Coding Scheme (MCS), the bits of a symbol are permuted. The permutation is the same for all symbols of a frame.

Convolutional Decoding and Puncturing: For decoding of the convolutional code and puncturing, the IT++ library is again utilized.

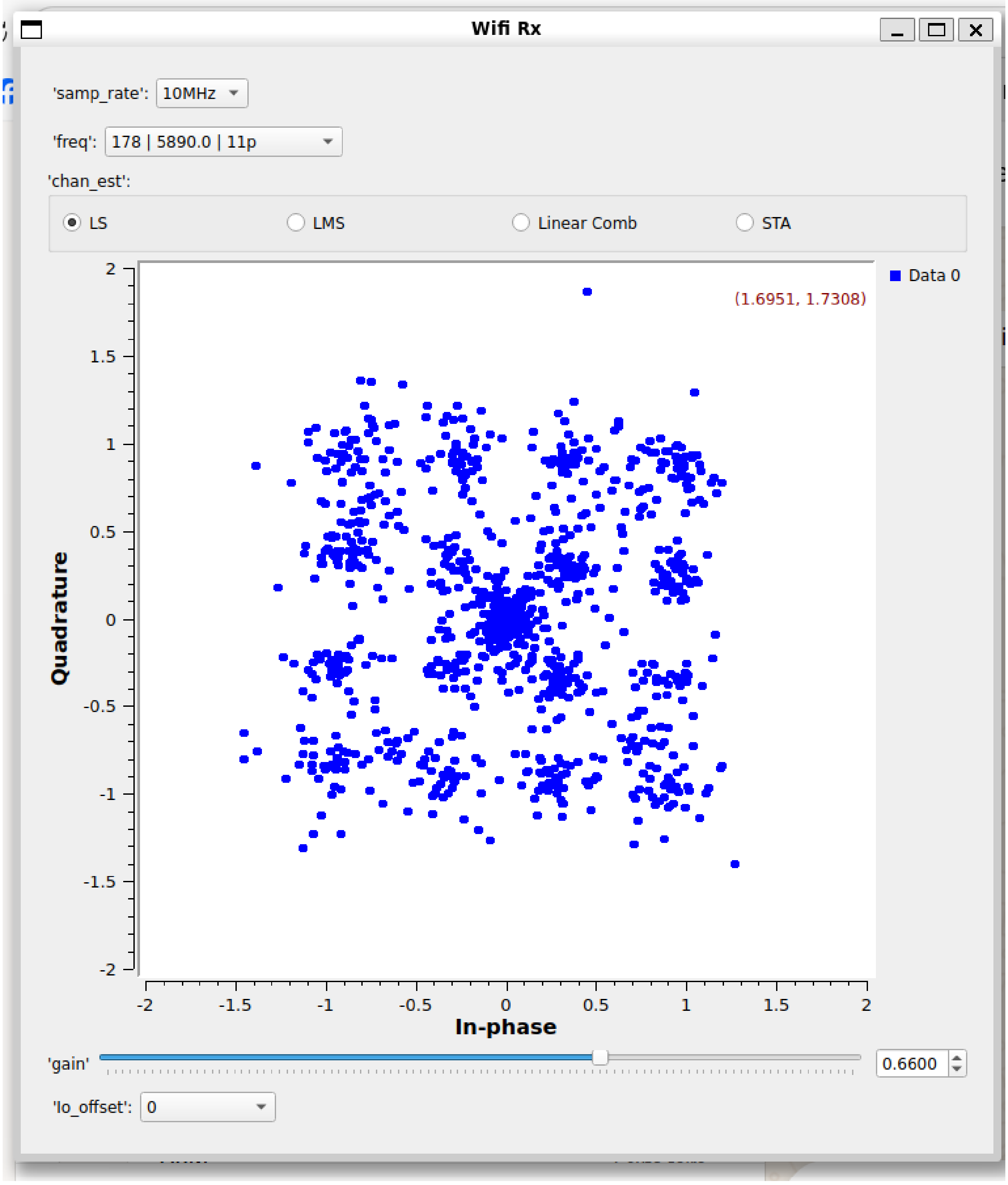
Descrambling: The final step in the decoding process is descrambling. In the encoder the initial state of the scrambler is set to a pseudo random value. As the scrambler is implemented with a seven bit feedback shift register, $2^7 = 128$ initial states are possible. The first 7bit of the payload are part of the service field and always set to zero, in order to allow the receiver to deduce the initial state of the scrambler. The mapping from these first bits to the initial state is implemented via a lookup table. Output: After the decoding process, the payload is packed into a GNU Radio message and passed to subsequent blocks in the flow graph. As final endpoint of the flow graph, we use a Socket PDU block of type UDP Server that sends the payload reencapsulated in a User Datagram Protocol (UDP) datagram. A user can then receive the datagrams, e.g., with netcat and see the payload appearing in their terminal. Therefore, the f lowgraph can be easily extended with custom applications.

How constellation symbols are demodulated

The demodulator operates on **one OFDM symbol at a time** (64-point FFT, 52 used subcarriers).
This is necessary because:

- each OFDM symbol carries a 48-subcarrier data vector
- the modulation (BPSK/QPSK/16-QAM/64-QAM) is *per subcarrier*, not across symbols
- interleaving and FEC coding operate across **bits**, not samples

Thus, demodulating one OFDM symbol at a time preserves the OFDM frame structure and ensures alignment with the PHY pipeline.



Wifi Rx

'samp_rate': 10MHz

'freq': 178 | 5890.0 | 11p

'chan_est':

● LS ○ LMS ○ Linear Comb ○ STA

'gain' 0.6600

'lo_offset': 0

De-interleaving

De-interleaving reverses the interleaving performed at the transmitter to combat burst errors.
In IEEE 802.11, bits are permuted across subcarriers to distribute the effect of channel fades and interferers.

The de-interleaver reconstructs the original order using the inverse permutation defined in the standard.

Descrambling

Scrambling randomizes transmitted bit patterns to:

- avoid long sequences of identical bits
- ensure DC balance
- aid clock recovery
- mitigate spectral lines

IEEE 802.11 uses a 7-bit LFSR polynomial:

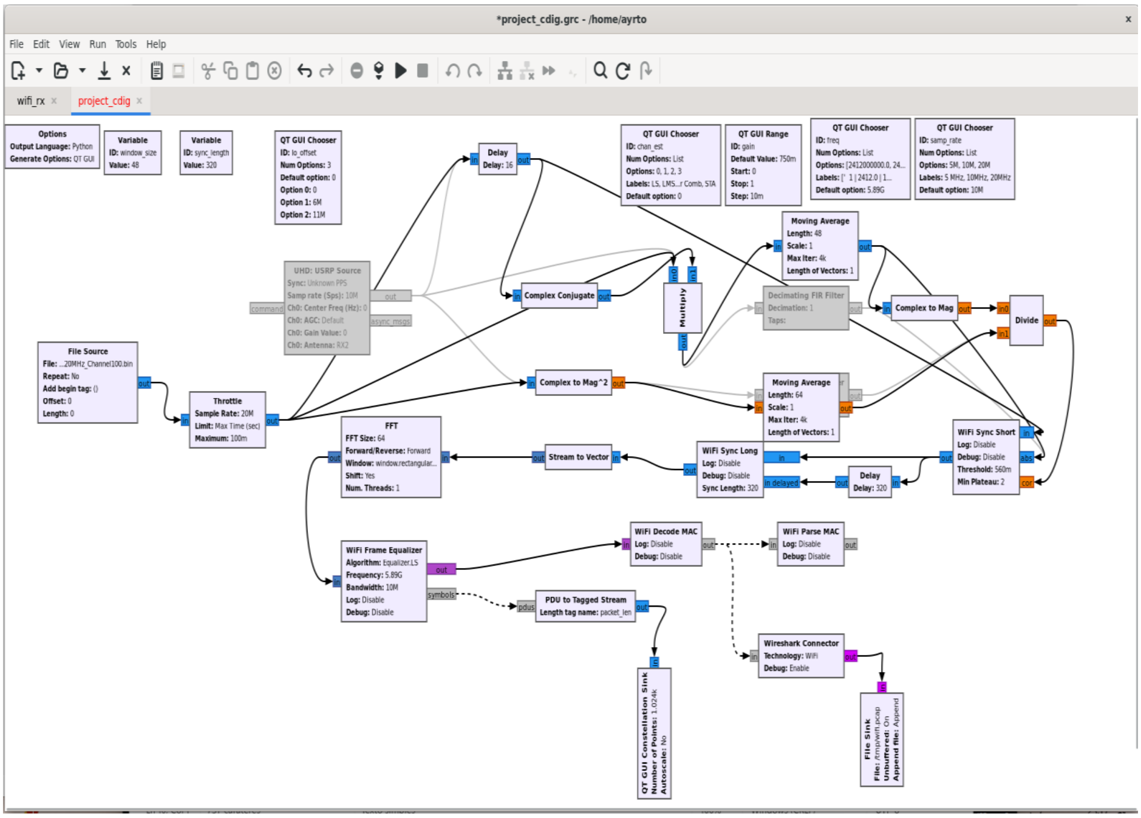
$$s_n = s_{n-7} \oplus s_{n-4}$$

MAC Frame Reconstruction

After de-interleaving, FEC decoding, and descrambling:

1. The MAC header fields (Frame Control, Duration, Addresses, Sequence Control) are parsed.
2. The payload is reassembled.
3. A CRC32 Frame Check Sequence is computed and compared with the received FCS.
4. Valid frames are emitted as PDUs (Protocol Data Units) to upper-layer sockets (Wireshark).

These fields are used to classify frame type (Data, Control, Management), verify integrity, and extract payload information.



Options
Output Language: Python
Generate Options: QT GUI

Variable
ID: window_size
Value: 48

Variable
ID: sync_length
Value: 320

QT GUI Chooser
ID: lo_offset
Num Options: 3
Default option: 0
Option 0: 0
Option 1: 6M
Option 2: 11M

Delay
Delay: 16

QT GUI Chooser
ID: chan_est
Num Options: List
Options: 0, 1, 2, 3
Labels: LS, LMS...r Comb, STA
Default option: 0

QT GUI Range
ID: gain
Default Value: 750m
Start: 0
Stop: 1
Step: 10m

QT GUI Chooser
ID: freq
Num Options: List
Options: [2412000000.0, 24...
Labels: [' 1 | 2412.0 | 1...
Default option: 5.89G

QT GUI Chooser
ID: samp_rate
Num Options: List
Options: 5M, 10M, 20M
Labels: 5 MHz, 10MHz, 20MHz
Default option: 10M

UHD: USRP Source
Sync: Unknown PPS
Samp rate (Sps): 10M
Ch0: Center Freq (Hz): 0
Ch0: AGC: Default
Ch0: Gain Value: 0
Ch0: Antenna: RX2

Moving Average
Length: 48
Scale: 1
Max Iter: 4k
Length of Vectors: 1

Complex Conjugate

Multiply

Decimating FIR Filter
Decimation: 1
Taps:

Complex to Mag

Divide

File Source
File: ...20MHz_Channel100.bin
Repeat: No
Add begin tag: ()
Offset: 0
Length: 0

Throttle
Sample Rate: 20M
Limit: Max Time (sec)
Maximum: 100m

Complex to Mag^2

Moving Average
Length: 64
Scale: 1
Max Iter: 4k
Length of Vectors: 1

FFT
FFT Size: 64
Forward/Reverse: Forward
Window: window.rectangular...
Shift: Yes
Num. Threads: 1

Stream to Vector

WiFi Sync Long
Log: Disable
Debug: Disable
Sync Length: 320

Delay
Delay: 320

WiFi Sync Short
Log: Disable
Debug: Disable
Threshold: 560m
Min Plateau: 2

WiFi Decode MAC
Log: Disable
Debug: Disable

WiFi Parse MAC
Log: Disable
Debug: Disable

WiFi Frame Equalizer
Algorithm: Equalizer.LS
Frequency: 5.89G
Bandwidth: 10M
Log: Disable
Debug: Disable

PDU to Tagged Stream
Length tag name: packet_len

Wireshark Connector
Technology: WiFi
Debug: Enable

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

File Sink
File: /tmp/wifi.pcap
Unbuffered: On
Append file: Append