EUROPEAN UNION

Interreg EUROPEAN UNION

Austria-Czech Republic

European Regional Development Fund

LoRa(WAN) Webinar

LoRa – Modulation/Encoding

Author: Harald Eigner (TU Wien)

These slides give an overview on the LoRa modulation and encoding. While the first slides are intended for general audience, the second part will explain modulation/encoding on a very detailed level, targeting advanced audience which want to generate/receive signals with their own hardware, like software defined radios.

TU WIEN TECHNISCHE UNIVERSITÄT WIEN Vienna | Austria

EMCE

Contents

Contents

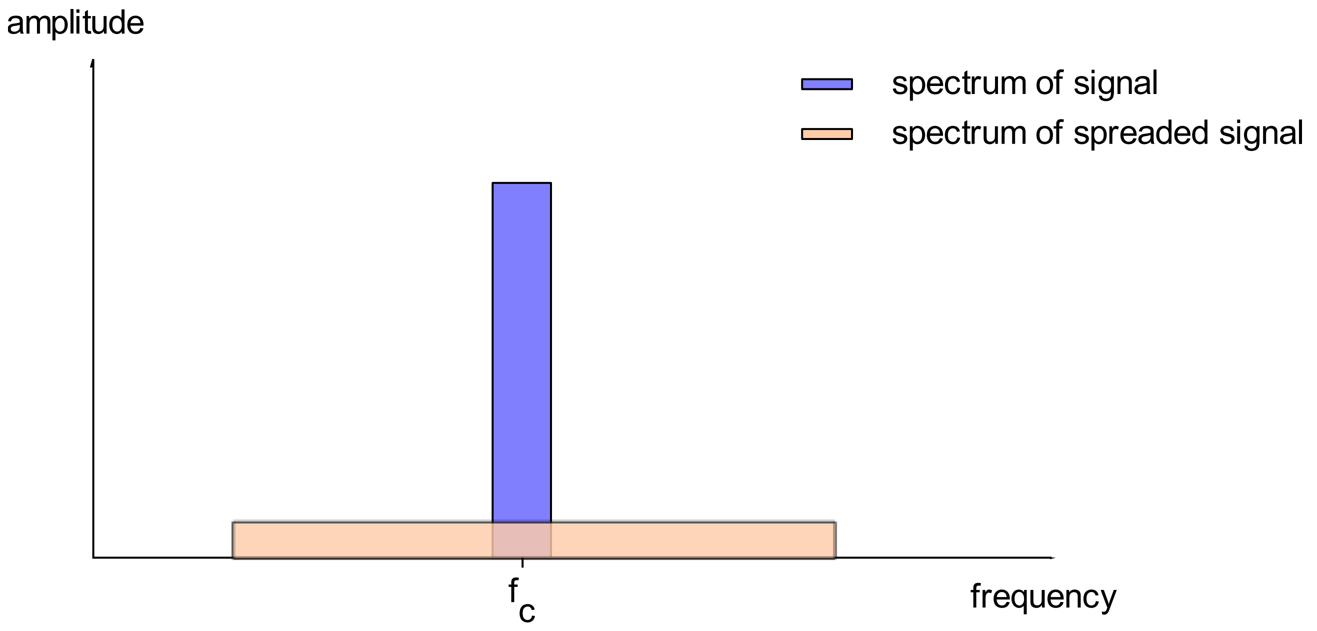
Spread Spectrum Basics

- Information is spread over frequency, high bandwidth
- Trade-off between data rate, bandwidth, and power

$$C = BW \log_2\left(1 + \frac{S}{N}\right)$$

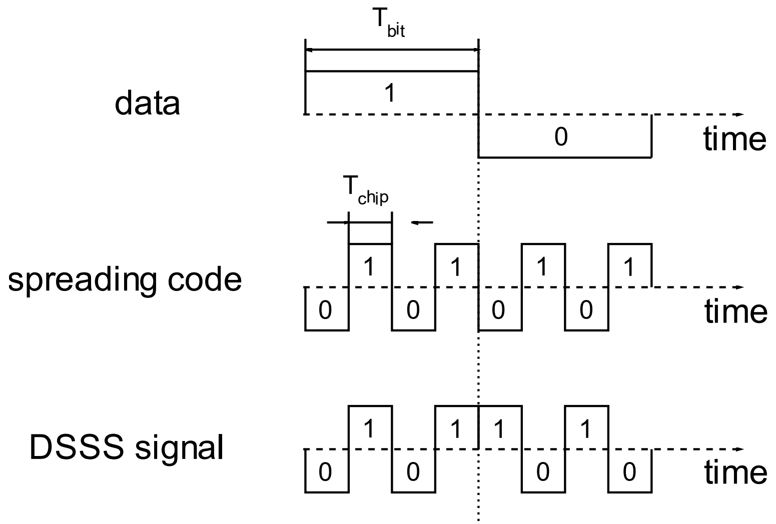
with

- C ... Channel Capacity $\left(\frac{bits}{s}\right)$
- BW ... Bandwidth (Hz)
- $\frac{S}{N}$ ... Signal to Noise ratio (1)



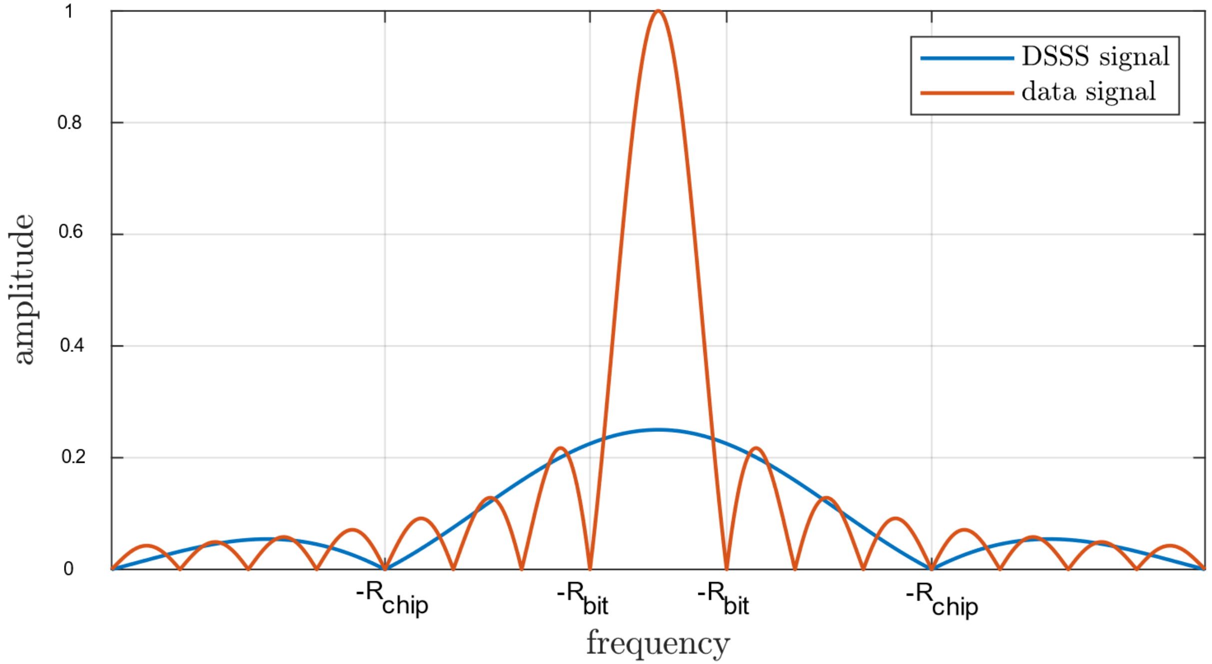
Direct Sequence Spread Spectrum

- Data signal is expanded by multiplying with pseudo random sequence – spreading code
- Information is spread over frequency



$$R_{bit} = \frac{1}{T_{bit}}$$

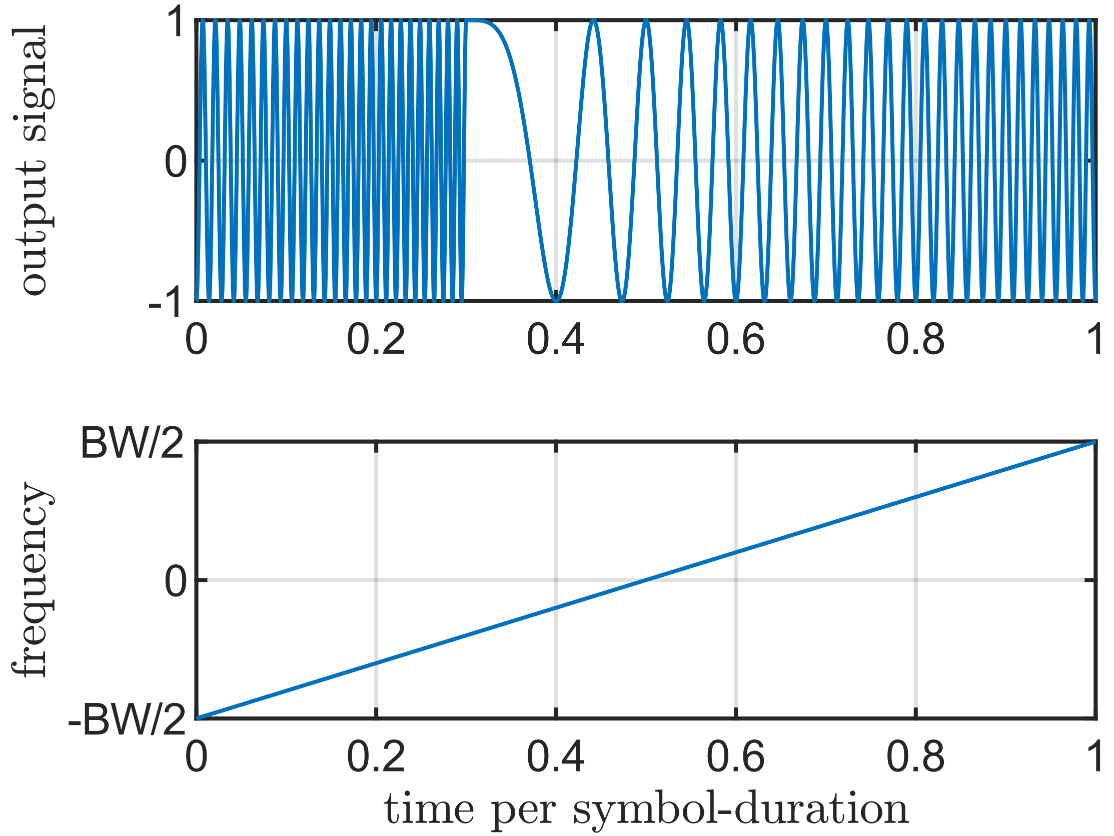
$$R_{chip} = \frac{1}{T_{chip}}$$



Contents

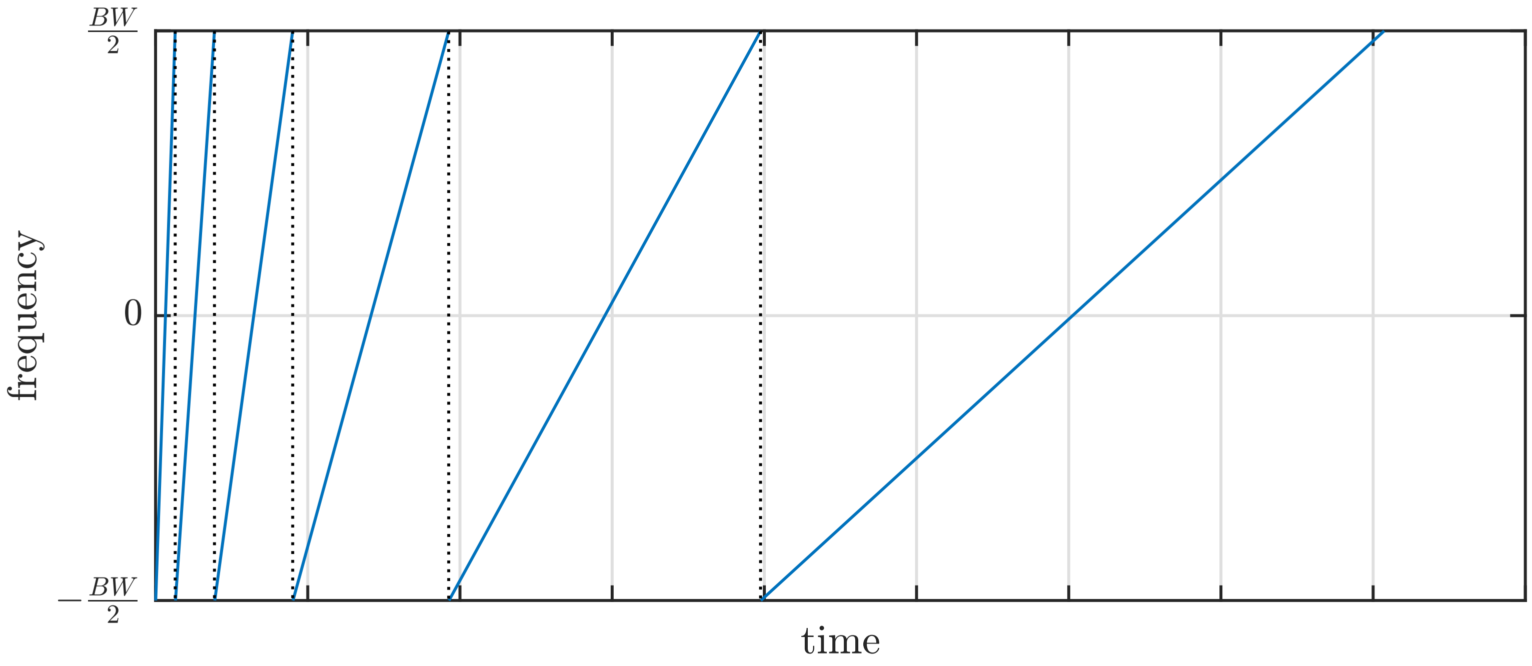
LoRa Chirped Spread Spectrum Modulation

- LoRa Chirp: signal with linearly increasing frequency
- Chirp length is determined by the spreading factor SF
- Information is spread over time



LoRa Chirped Spread Spectrum Modulation

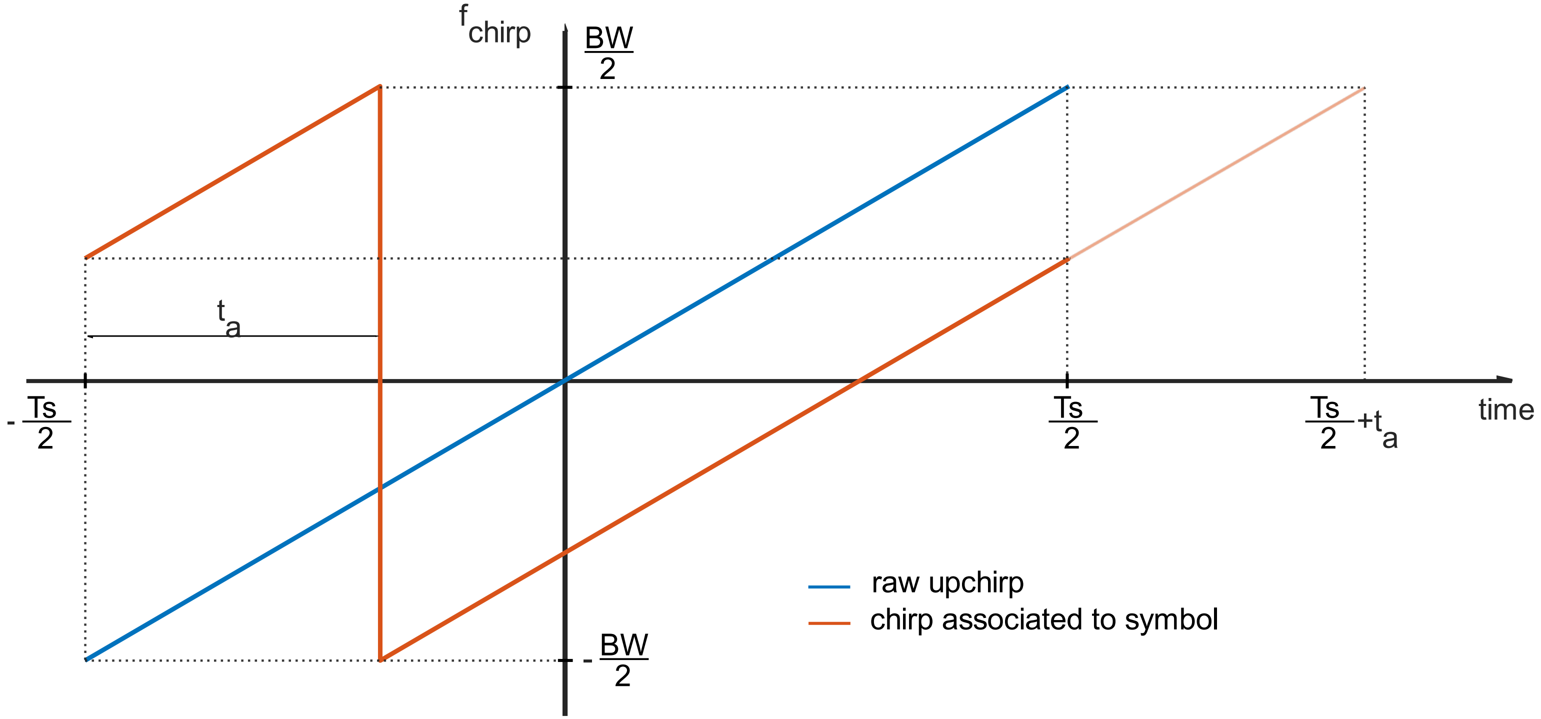
- Parameters defining the chirp:
  - Spreading factor $SF = 7 \dots 12$
  - Bandwidth $BW = 125,\ 250,\ 500\ kHz$
  - Code Rate $CR = {}^{4}/_{5}, {}^{4}/_{6}, {}^{4}/_{7}, {}^{4}/_{8}$
  - Chirp length $T_S = \frac{2^{SF}}{BW}$



LoRa chirps with $SF = 7 \dots 12$

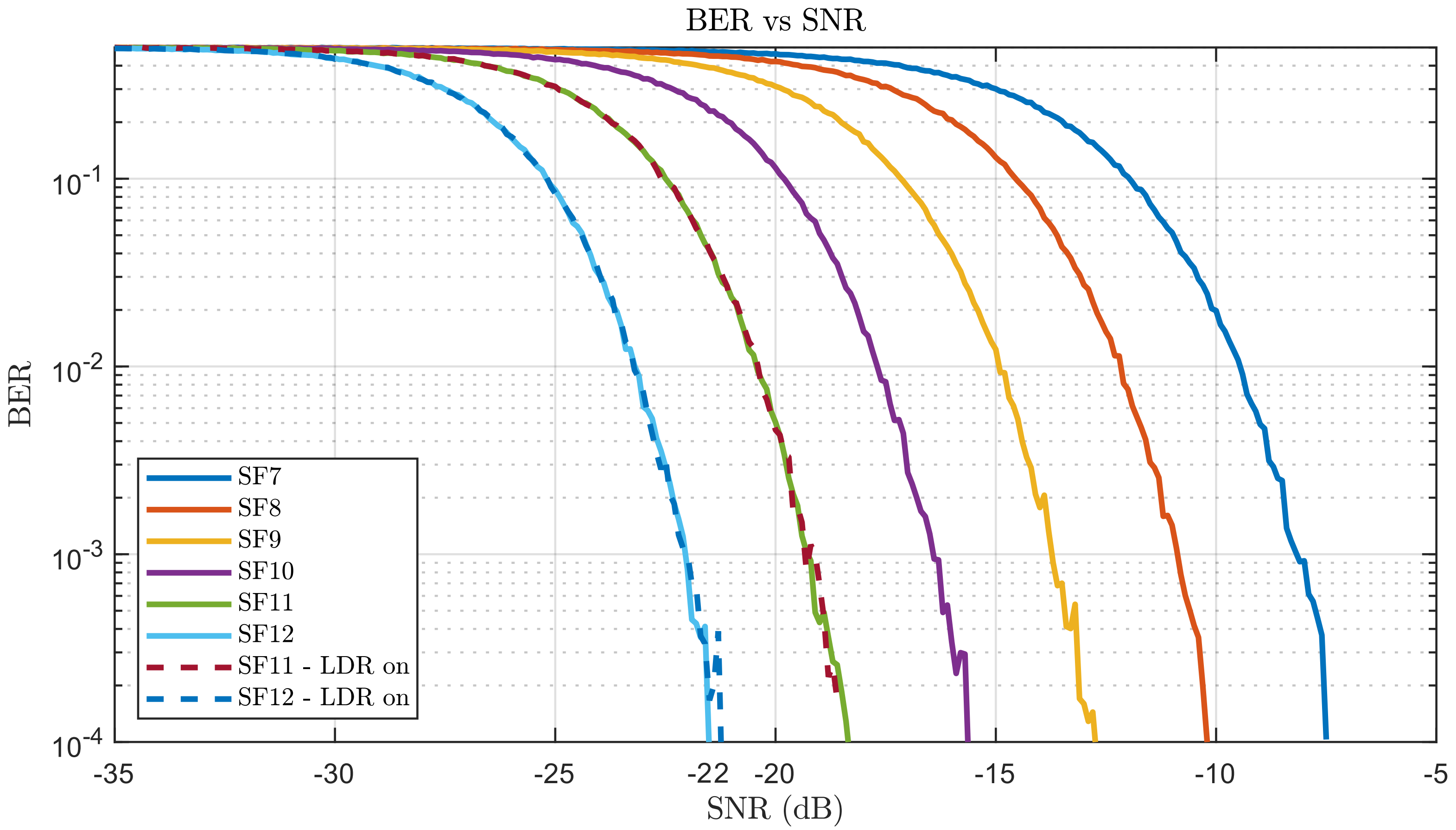
LoRa Chirped Spread Spectrum Modulation

- Data encoded in cyclically shifted frequency ramp
- Symbol alphabet size $N = \begin{cases} SF & \text{with disabled low data rate mode} \\ SF - 2 & \text{with enabled low data rate mode} \end{cases}$
- Symbol alphabet $a \in \{0, \dots, M-1\}$



LoRa Chirped Spread Spectrum Modulation

- Bit Error Ratio over Signal to Noise Ratio



- SNR = -22 dB for BER = 10^{-4} with $SF = 12$

LDR … Low data rate mode

LoRa Chirped Spread Spectrum Modulation

- Receiver Sensitivity: Receiving power at which a certain maximum BER is achieved
- Example:
  - $BW = 125\ kHz$
  - Noise Figure $NF = 6\ dB$
  - $SF = 12$
  - $SNR = -22\ dB$ for $BER = 10^{-4}$

$$S = -174 + 10\ \log_{10} BW + NF + SNR$$
$$S = -174 + 10 \log_{10}(125 \cdot 10^3) + 6 - 22 = \mathbf{-139\ dBm}$$

- Link Budget:
  - Antenna transmit power $P = +15\ dBm$

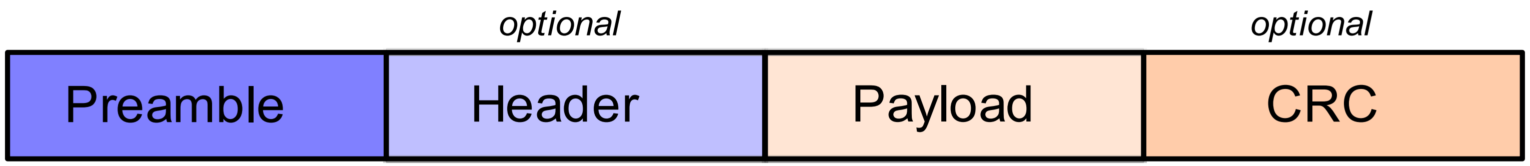
$$Link\ Budget = P - S = 15\ dBm - (-139\ dBm)$$
$$Link\ Budget = \mathbf{154\ dB}$$

Contents

- Spread Spectrum Basics
- LoRa Chirped Spread Spectrum
- LoRa Frame Format
- LoRa Encoding Scheme
- Airtime

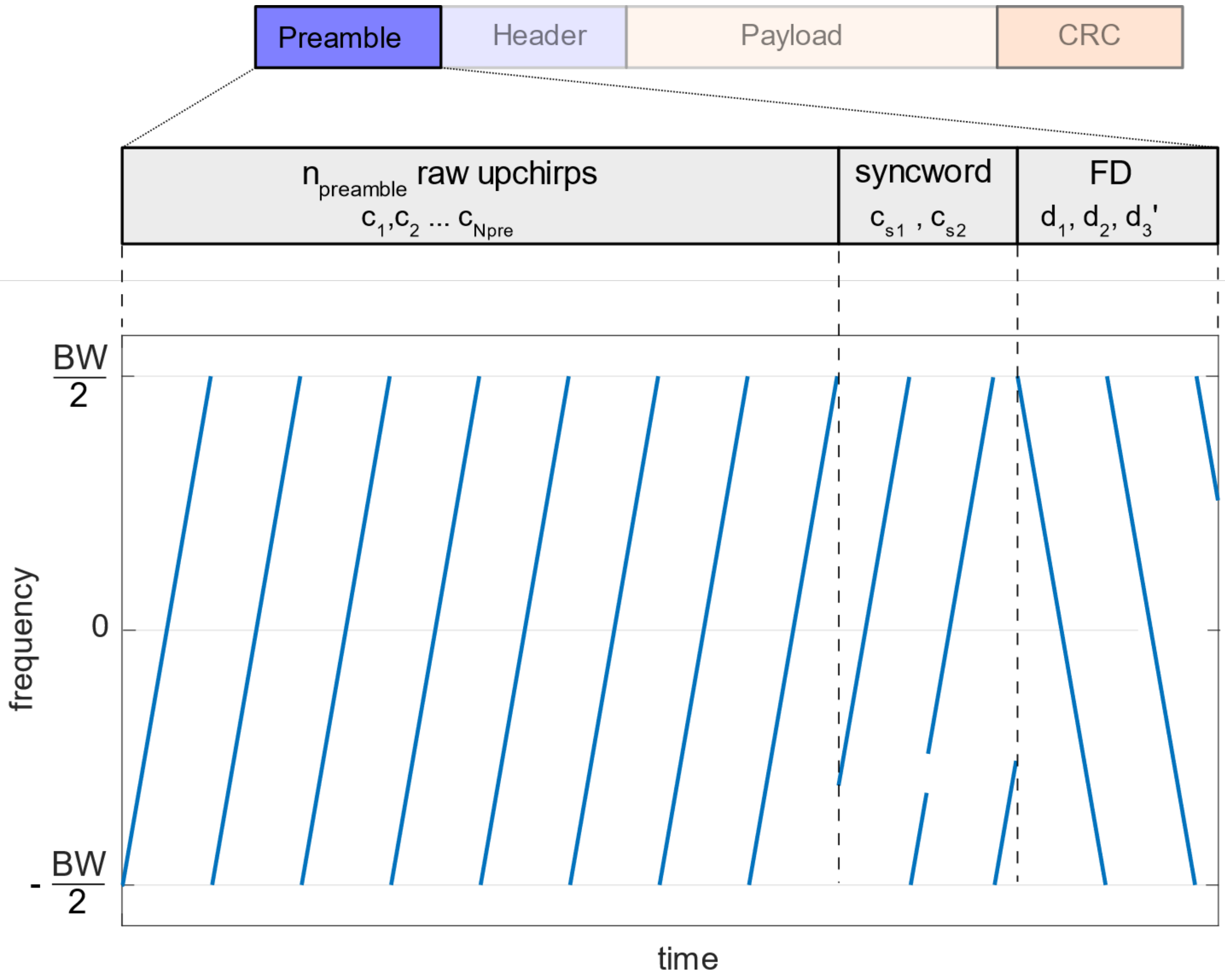
LoRa Frame Format

- Preamble
- Optional header
- Payload
- Cyclic redundancy check CRC



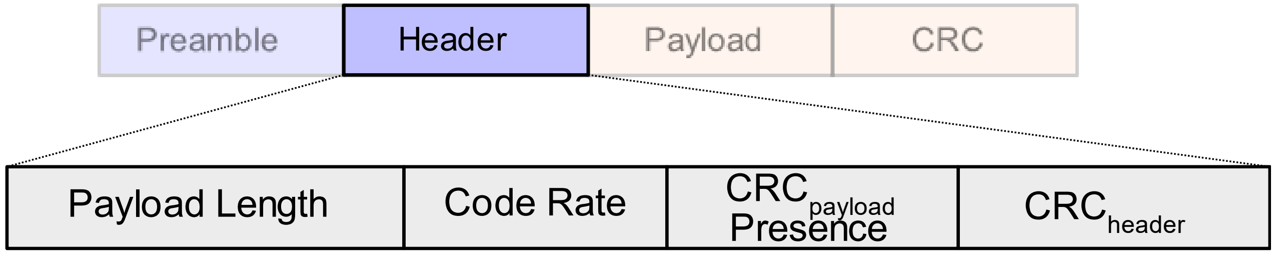
Preamble

- $n_{preamble}$ raw upchirps $c_{1,} c_{2, \dots} c_{N_{pre}}$ for synchronization at receiver
- Syncword: two upchirps $c_{s1,}$ $c_{s2,}$ defining the mode of the communication. Public or private mode
- Frame Delimiter FD: 2.25 downchirps $d_{2,}$ d_2, and $d_3{}'$ to end the preamble
  - $d_3{}'$ describes the chirp with length $\frac{T_S}{4}$



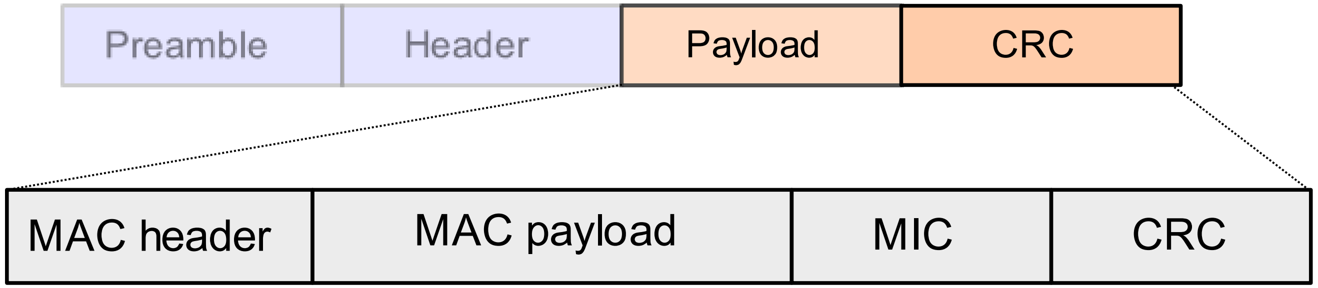
Header

- Information included in Header:
  - Payload length — 2 Bytes
  - Code Rate — 3 Bit
  - Presence of a payload-CRC — 1 Bit
  - Header-CRC — 2 Bytes
- Has a length of 8 symbols and is always encoded with highest code rate of $CR = {}^{4}/_{8}$
- 8 symbols include information of $56 \ldots 96$ bit ,depending on spreading factor
- Unused space in the first 8 symbols is filled with payload data



Payload

- Payload length $0 \ldots 255\ Byte$
- MAC header: defines message type (data exchange, join request, …)
- MAC Payload: either generic or LoRaWAN payload
- Every code rate possible
- Message Integrity Code MIC: Computed with the network session key
- Optional Cyclic Redundancy Check CRC: length 2 Bytes

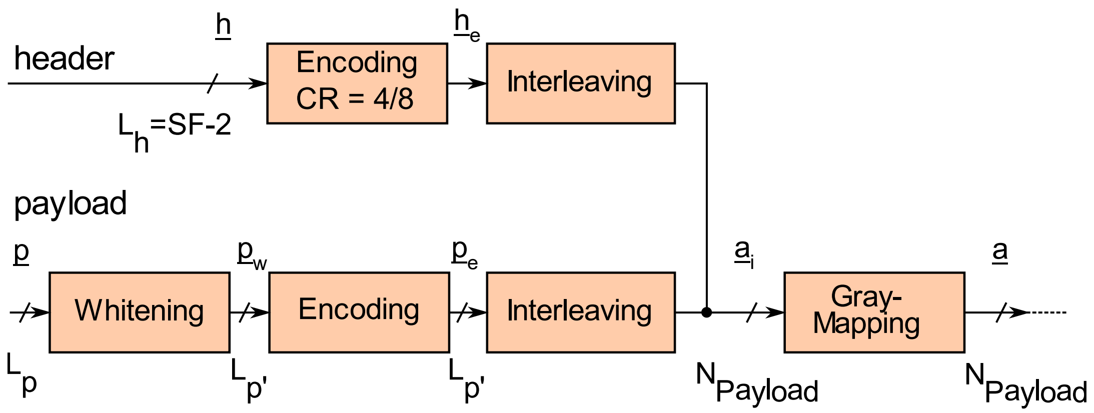


Contents

- Spread Spectrum Basics
- LoRa Chirped Spread Spectrum
- LoRa Frame Format
- LoRa Encoding Scheme
- Airtime

LoRa Encoding

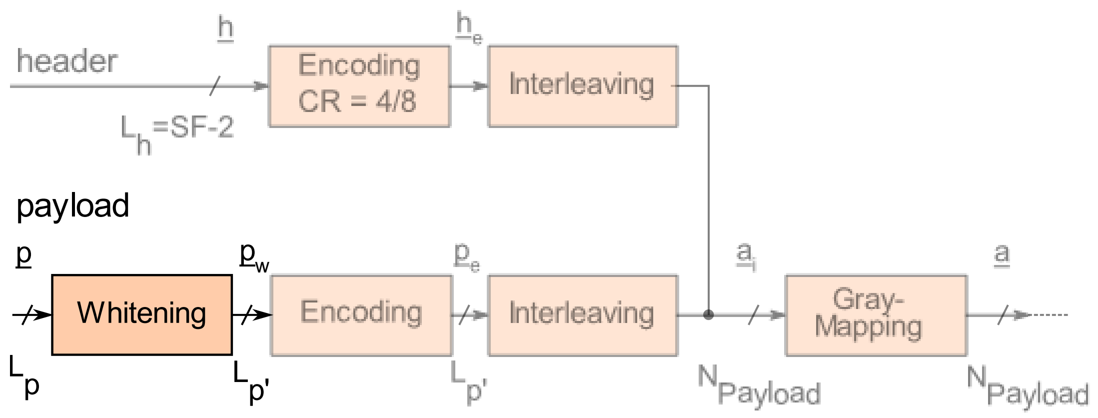
- Several encoding steps to increase resilience against interference
- Different treatment for header and payload
- *Example with* $SF = 8$, $CR = 4/5$, $L_p = 2Byte$, *and payload* $\underline{p} = \begin{bmatrix} 0x\,00 \\ 0x\,00 \end{bmatrix}$

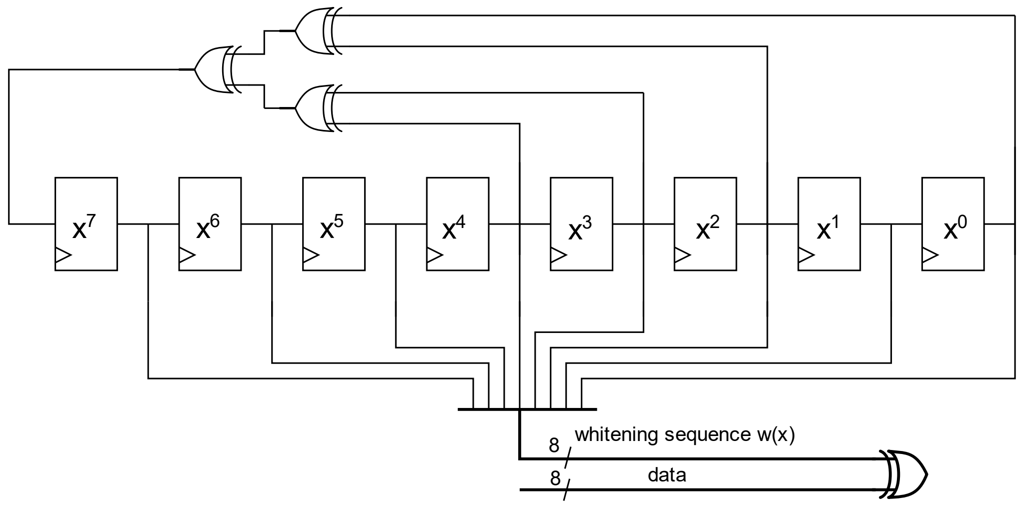


Whitening

- Removes DC-bias in data; only payload gets whitened
- XOR with whitening sequence $\underline{w}$
  - $\underline{p_w} = \underline{p} \oplus \underline{w}$ $\qquad\qquad$ $\underline{p_w} = \underline{p} \oplus \underline{w}$
- Whitening sequence from linear feedback shift register with polynomial:
  - $\underline{w}(x) = x^4 \oplus x^3 \oplus x^2 \oplus 1$

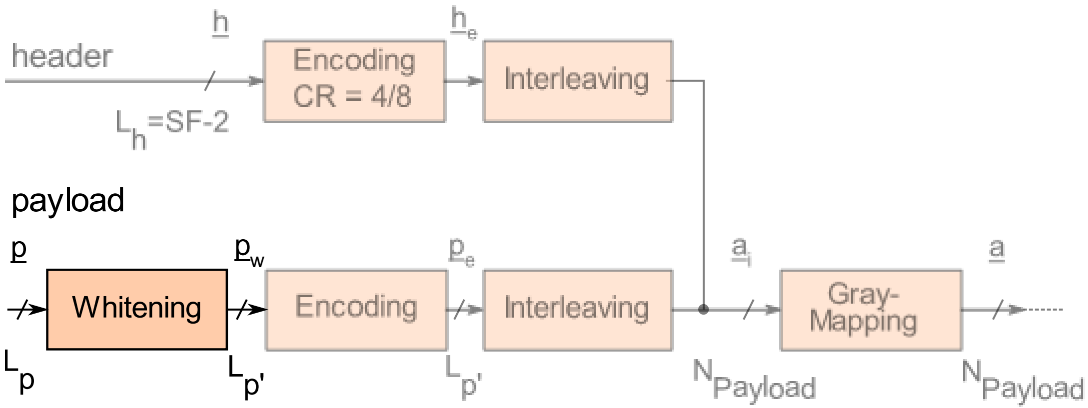
$$\underline{w} = \begin{bmatrix} 0\mathrm{x}\ FE \\ 0\mathrm{x}\ FF \\ 0\mathrm{x}\ FF \\ \dots \end{bmatrix}$$ *first bytes of the whitening sequence*





Whitening Example

- Payload $\underline{p} = \begin{bmatrix} 0\text{x }00 \\ 0\text{x }00 \end{bmatrix}$
- Whitening sequence $\underline{w} = \begin{bmatrix} 0\text{x }FF \\ 0\text{x }FE \end{bmatrix}$
- Whitened payload $\underline{p_w} = \underline{p} \oplus \underline{w} = \begin{bmatrix} 0\text{x }FF \\ 0\text{x }FE \end{bmatrix}$

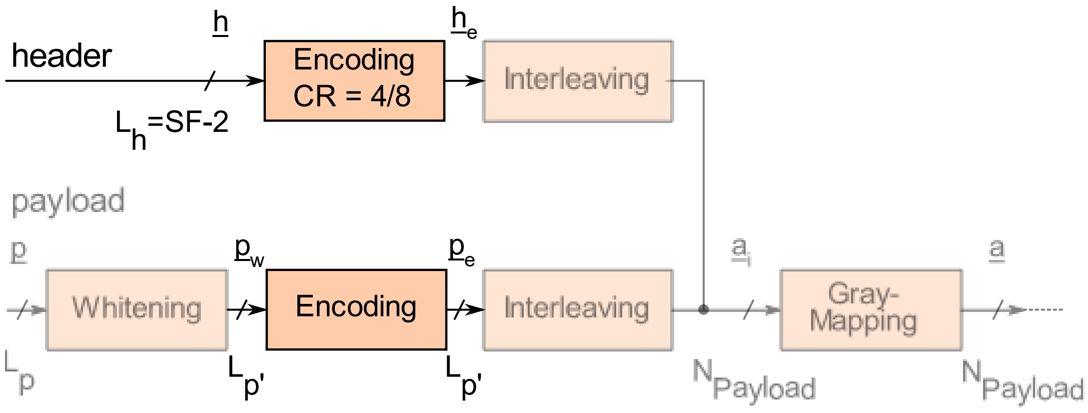


Whitening Matlab Implementation

```
function p = compute_whitening(p)
%COMPUTE_WHITENING performs a whitening of the databytes p
%   The function performs a whitenig of the databits p. The whitening
%   sequence is computed with a LFSR with the polynomial:
%   w(x) = x^4 xor x^3 xor x^2 xor 1 and a starting value of 0x FF


register = double(0xFF); % starting value of 0x FF

%perform whitening
for idx = 1 : length(p)
    p(idx) = bitxor(p(idx), register);
    feedback = mod(sum(dec2binvec(bitand(register, double(0xB8)))), 2);
    register = bitand(register * 2, double(0xFF)) + feedback;
end
```

Encoding

- Four different code-rates
  - $CR = 4/5$ parity check
  - $CR = 4/6$ shortened (4,7) Hamming code
  - $CR = 4/7$ (4,7) Hamming code
  - $CR = 4/8$ extended (4,7) Hamming code
- Header is always encoded with $CR = 4/8$ and low data rate mode -> length $L_h = SF - 2$ nibbles of 4 Bit
- Remaining space is filled with payload data
- Small payload may fit completely in header symbols
- Data $\underline{p_w}$ transformed from Bytes to Nibbles of 4 Bit



Encoding

- Parity bits:

CR = 4/8	p_3	p_2	p_1	p_0	d_3	d_2	d_1	d_0
CR = 4/7		p_2	p_1	p_0	d_3	d_2	d_1	d_0
CR = 4/6			p_1	p_0	d_3	d_2	d_1	d_0
CR = 4/5				p_4	d_3	d_2	d_1	d_0

- Parity bit calculation:

$$
\begin{aligned}
p_0 &= d_0 \oplus d_1 \oplus d_2 \\
p_1 &= d_1 \oplus d_2 \oplus d_3 \\
p_2 &= d_0 \oplus d_1 \oplus d_3 \\
p_3 &= d_0 \oplus d_2 \oplus d_3 \\
p_4 &= d_0 \oplus d_1 \oplus d_2 \oplus d_3
\end{aligned}
$$

- Error correction capabilities:

Error Detection

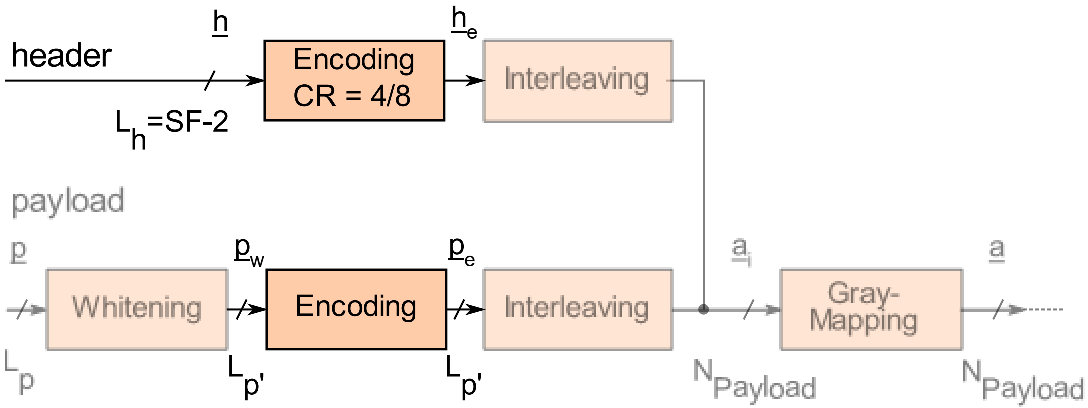
bit error / code rate	1	2	3	4
4/5				
4/6				
4/7				
4/8				

Error Correction

bit error / code rate	1	2	3	4
4/5				
4/6				
4/7				
4/8				

Encoding Example

- Header $L_h = SF - 2 = 6$



$$\underline{h} = \begin{bmatrix} 0\text{x } 0 \\ 0\text{x } 2 \\ 0\text{x } 2 \\ 0\text{x } 0 \\ 0\text{x } E \\ 0\text{x } F \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$L_p = 2$

CRC disabled

$CR = 4/5$

Header CRC

Remaining space is filled with Payload data

- Payload

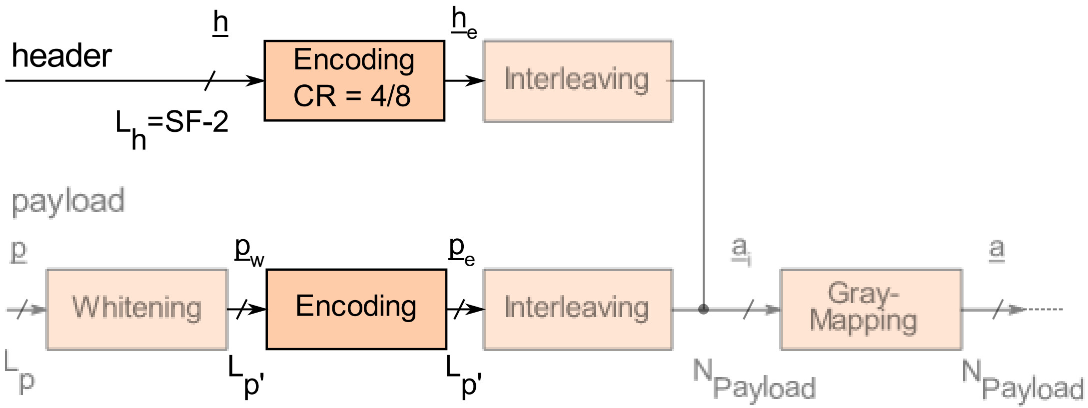
$$\underline{p_w} = \begin{bmatrix} 0\text{x } F \\ 0\text{x } F \\ 0\text{x } E \\ 0\text{x } F \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \rightarrow \quad \underline{p_w{}'} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- Header $CR = 4/8$

$$\begin{array}{cc} & \begin{array}{cc}\text{parity bits} & \text{header bits}\end{array} \\ \underline{h_e} = & \left[\begin{array}{cccc|cccc} 0&0&0&0&0&0&0&0 \\ 0&1&1&1&0&0&1&0 \\ 0&1&1&1&0&0&1&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&0&1&1&1&0 \\ 1&1&1&1&1&1&1&1 \end{array}\right]\end{array}$$

- Payload $CR = 4/5$

$$\begin{array}{cc} & \begin{array}{cc}\text{parity bits} & \text{payload bits}\end{array} \\ \underline{p_e} = & \left[\begin{array}{c|cccc} 0&1&1&1&1 \\ 1&0&1&1&1 \\ 0&1&1&1&1 \end{array}\right]\end{array}$$



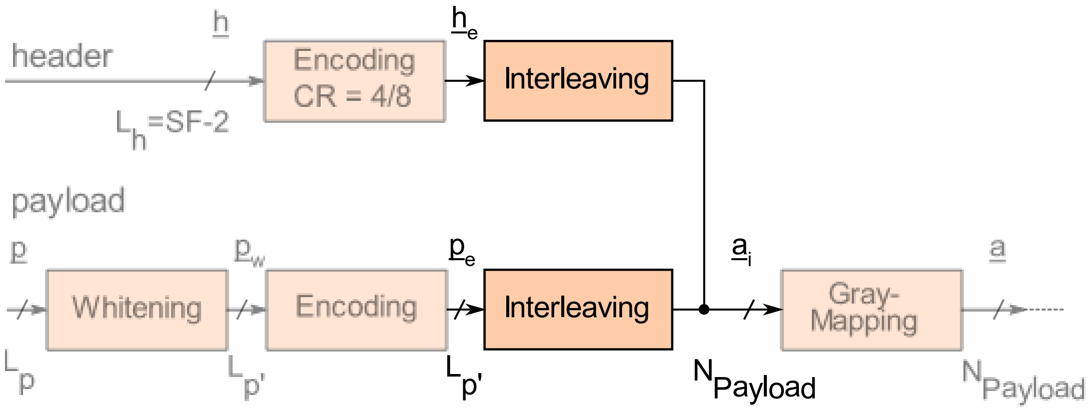
Encoding Matlab Implementation

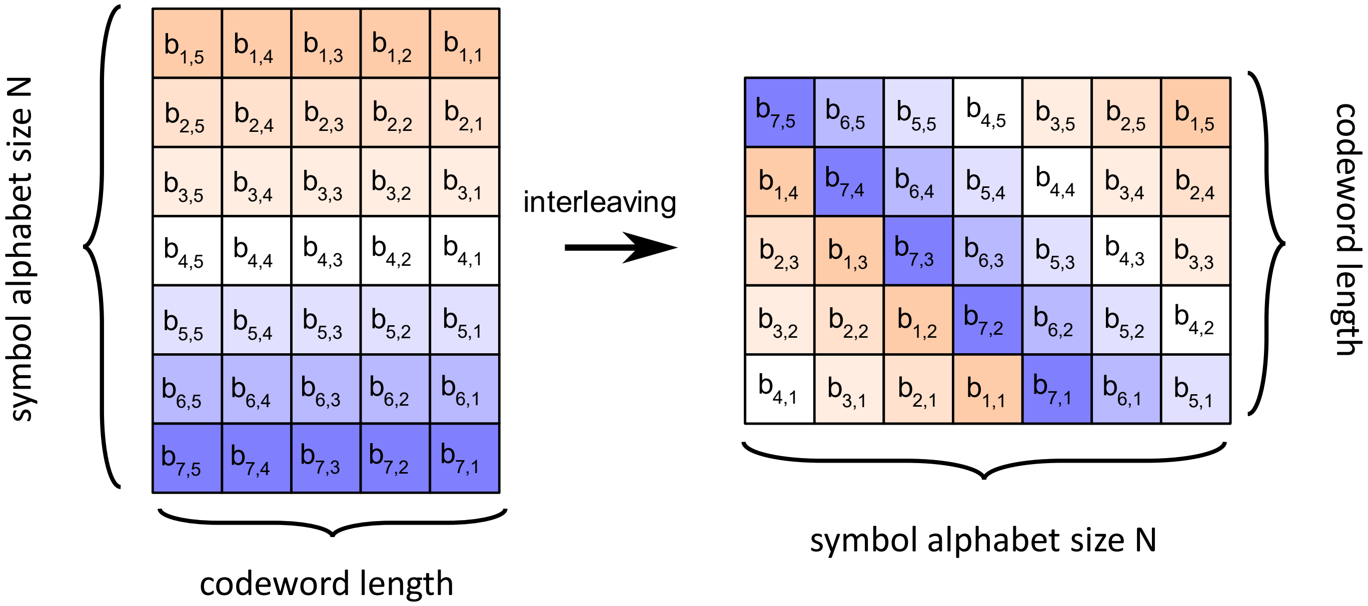
```
function codewords = encoder(B, CR)
%ENCODE Data encoding
%   This function encodes the input data from
%   matrix B with code rate:
%   CR = 1 ... code rate 4/5
%   CR = 2 ... code rate 4/6
%   CR = 3 ... code rate 4/7
%   CR = 4 ... code rate 4/8


len = size(B, 1);
N = CR;
k = 4;
codewords = zeros(len, k);
P = zeros(k, N - k);
I = logical(eye(k));
```

```
%Create generator matrix G
if CR > 5
    for idx = 1 : 4
        P(idx, 1) =  xor(xor(I(idx, 1), I(idx, 2)), I(idx, 3));
        P(idx, 2) =  xor(xor(I(idx, 2), I(idx, 3)), I(idx, 4));
        P(idx, 3) =  xor(xor(I(idx, 1), I(idx, 2)), I(idx, 4));
        P(idx, 4) =  xor(xor(I(idx, 1), I(idx, 3)), I(idx, 4));
    end
    %Create generator matrix for CR = 1 ... 3
    G=[I P(1 : 4, 1 : (CR - 4))]; elseif CR == 5
    for idx = 1:4
        P(idx, 1) =  xor(xor(I(idx, 1), I(idx, 2)), xor(I(idx, 3), I(idx, 4)));
    end
    G=[I P];    %Create generator matrix for CR = 4
end

%Generation of codewords
for idx1 = 1 : len
    for idx2 = 1 : N
        %multiplication data with generator matrix c = u * G
        codewords(idx1, idx2 )= mod(sum(B(idx1, :).*G(:, idx2)'), 2);
    end
end

end
```

Interleaver

- Diagonal interleaver
- Multiple bit errors caused by one symbol are highly correlated
- Spreads multiple bit errors over multiple codewords -> decorrelation





Interleaving example with $SF = 7$ and $CR = 4/5$

<u>symbol alphabet size:</u>

$$N = \begin{cases} SF & \text{disabled low data rate mode} \\ SF-2 & \text{enabled low data rate mode} \end{cases}$$

<u>codeword length:</u>

- 5 bit for $CR = {}^4/_5$
- 6 bit for $CR = {}^4/_6$
- 7 bit for $CR = {}^4/_7$
- 8 bit for $CR = {}^4/_8$

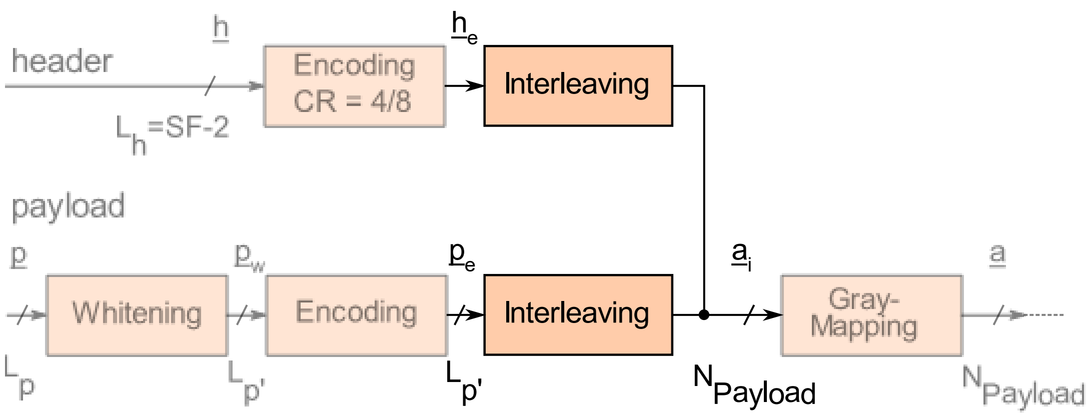
Interleaver Matlab Implementation

```
function B_int = interleave(B)
%INTERLEAVE interleaves the binary codeblock B
%   This function performs the diagonal interleaving of the codeblock B. It transoforms
%   the binary matrix B of size symbol alphabet times codeword length (N x CR + 4) to the binary
%   matrix B_int of size codeword length times symbol alphabet size (CR + 4 x N)

c = size(B, 1);
s = size(B, 2);
bin_interleaved1 = zeros(c, s);
B_int = zeros(c, s);

%perform interleaving
for idx1 = 1 : c
    for idx2 = 1 : s
        bin_interleaved1(idx1, idx2) = B(c - (idx1 - 1), idx2);
    end
end

for idx1 = 1 : c
    for idx2 = 1 : s
        B_int(idx1, idx2)=bin_interleaved1(1 + mod(idx1 - idx2, c), idx2);
    end
end
B_int = fliplr(B_int.');
end
```

Interleaver Example

- Header

$$\underline{h_e} = \left.\overbrace{\begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&1&1&1&0&0&1&0\\ 0&1&1&1&0&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&1&0&1&1&1&0\\ 1&1&1&1&1&1&1&1 \end{bmatrix}}^{8\text{ bit}}\right\}6\text{ bit} \quad \rightarrow \quad \underline{h_i} = \left.\overbrace{\begin{bmatrix} 1&0&0&0&0&0\\ 0&1&1&0&1&1\\ 0&0&1&1&0&0\\ 0&0&0&1&1&0\\ 0&1&1&0&1&0\\ 1&0&1&1&0&1\\ 1&0&0&1&1&0\\ 0&1&0&0&0&0 \end{bmatrix}}^{6\text{ bit}}\right\}8\text{ bit}$$



- Payload CR

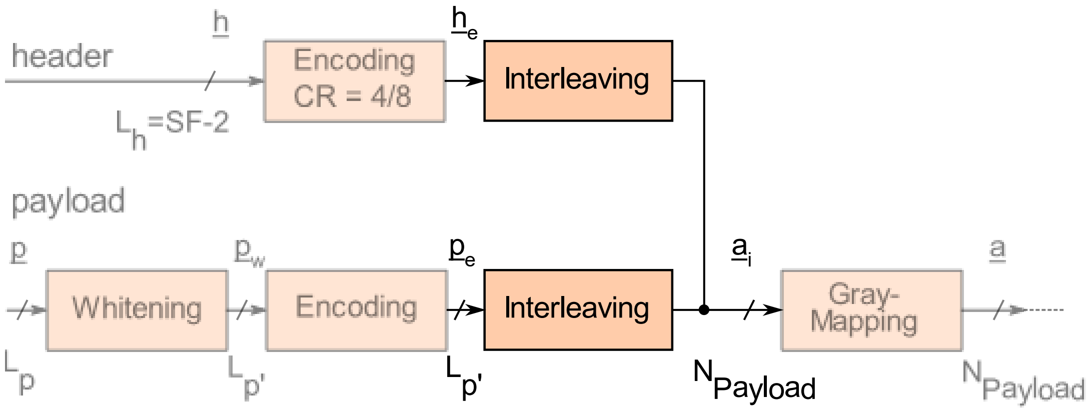
$$\underline{p_e} = \left.\overbrace{\begin{bmatrix} 0&1&1&1&1\\ 1&0&1&1&1\\ 0&1&1&1&1\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{bmatrix}}^{5\text{ bit}}\right\}8\text{ bit} \quad \rightarrow \quad \underline{p_i} = \left.\overbrace{\begin{bmatrix} 0&0&0&0&0&0&1&0\\ 1&0&0&0&0&0&1&0\\ 1&1&0&0&0&0&0&1\\ 1&1&1&0&0&0&0&0\\ 0&1&1&1&0&0&0&0 \end{bmatrix}}^{8\text{ bit}}\right\}5\text{ bit}$$

Payload is extended with random data to get a block of data with symbol alphabet size N times codeword length

Interleaver Example cont.

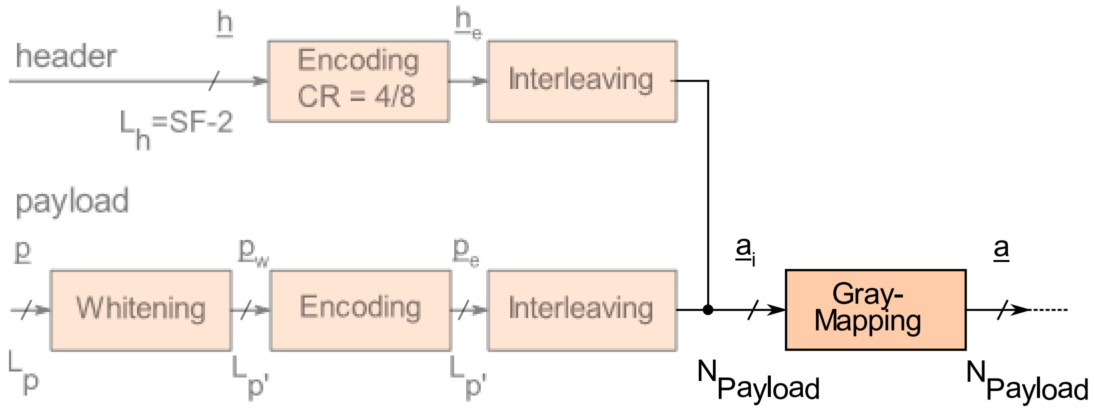
- Header and payload is stacked to form codewords $\underline{a}_i$

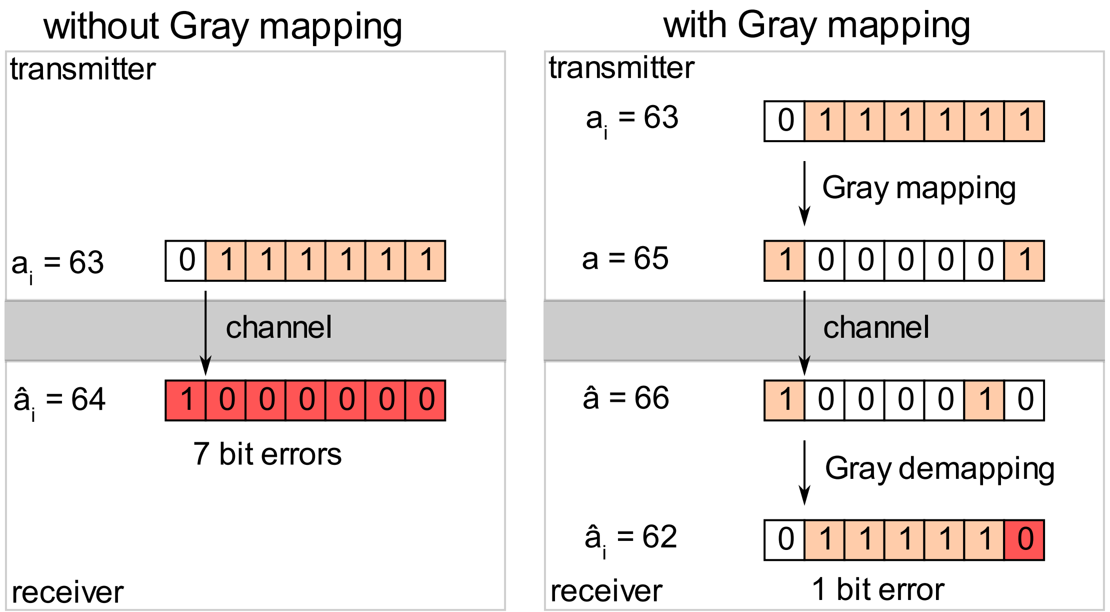
$$\underline{h} = \begin{bmatrix} \underline{h_i} \\ \underline{p_i} \end{bmatrix} = \left[\begin{array}{cccccccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{c} 0x\,02 \\ 0x\,B1 \\ 0x\,C0 \\ 0x\,60 \\ 0x\,A1 \\ 0x\,D2 \\ 0x\,62 \\ 0x\,01 \\ \hline 0x\,20 \\ 0x\,28 \\ 0x\,1C \\ 0x\,0E \\ 0x\,09 \end{array}\right]$$



- Erroneous symbol most likely mistaken with adjacent symbol.
- Gray code maps a bit sequence such that two successive values differ by just one bit

$$a_j = \begin{cases} a_i\,(j,k) & \text{for } k = 0 \\ a_i\,(j,k) \oplus a_i\,(j,k-1) & \text{else} \end{cases}$$

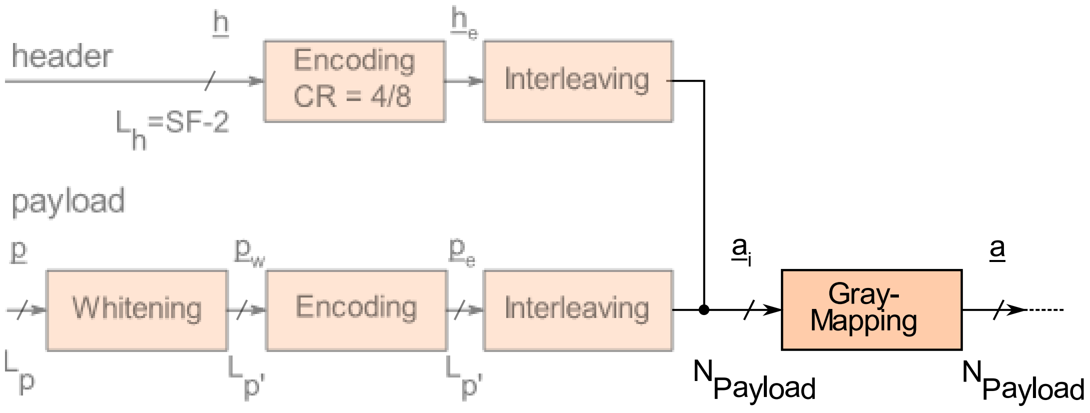




Receiver mistakes symbol a_i with adjacent symbol $\hat{a}_i = a_i + 1$

Gray Mapping Example

$$\underline{a_i} = \begin{bmatrix} 0x\ 02 \\ 0x\ B1 \\ 0x\ C0 \\ 0x\ 60 \\ 0x\ \mathrm{A1} \\ 0x\ D2 \\ 0x\ 62 \\ 0x\ 01 \\ 0x\ 20 \\ 0x\ 28 \\ 0x\ \mathrm{1C} \\ 0x\ 0E \\ 0x\ 09 \end{bmatrix} \Rightarrow \underline{a} = \begin{bmatrix} 0x\ F3 \\ 0x\ 21 \\ 0x\ 40 \\ 0x\ 80 \\ 0x\ 23 \\ 0x\ B1 \\ 0x\ 93 \\ 0x\ E3 \\ 0x\ 06 \\ 0x\ FD \\ 0x\ 18 \\ 0x\ DF \\ 0x\ CF \end{bmatrix} \equiv \begin{bmatrix} 243 \\ 33 \\ 64 \\ 128 \\ 35 \\ 177 \\ 147 \\ 227 \\ 6 \\ 253 \\ 24 \\ 223 \\ 207 \end{bmatrix}$$

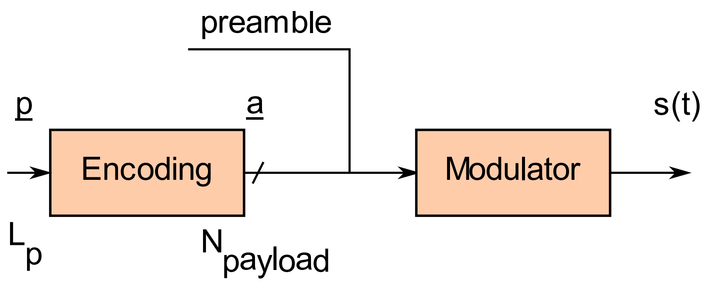


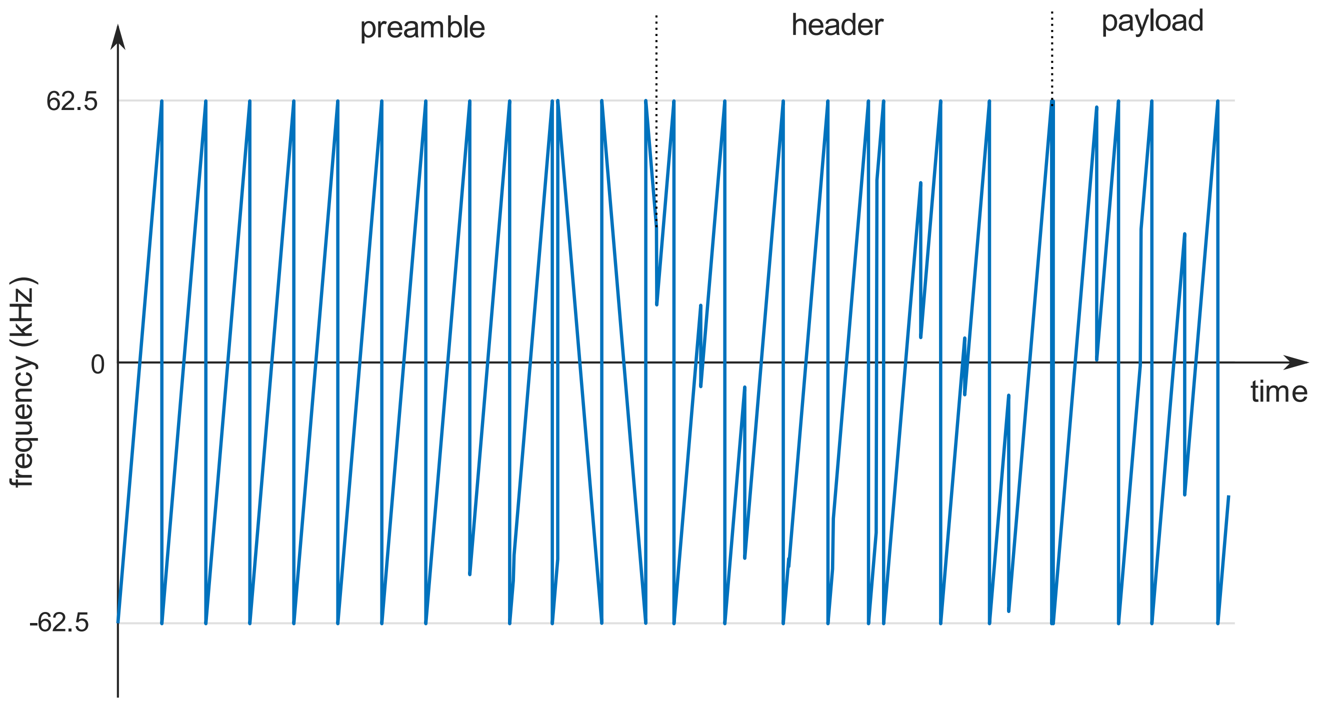
Gray Mapping Matlab Implementation

```
function g = bin2gray(b)
%BIN2GRAY performs a Gray mapping of the binary vector b
%   The function performs a Gray mapping of the binary vector b according
%   to:
%   g_i = b_i               for i = 0
%   g_i = b_i xor g_i-1     else

len = size(b, 2);
g(len) = b(len);

for idx = 1 : len - 1
    x = xor(b(len - idx + 1), b(len - idx));
    g(len - idx) = x;
end
end
```

Modulator

- After encoding steps, the preamble is added
- The symbols are modulated to the signal





LoRa signal s(t)

Modulator Matlab Implementation

```
function iqdata = LoRa_modulator(symbols, SF, PL, syncword, ovs)
%LORA_MODULATOR creates a LoRa signal
%   This function crates a LoRa signal for the vector symbols with the
%   spreading factor SF, the preamble length PL, the syncword, and the
%   oversampling factor ovs.

iqdata = [];
phaseacc = 0;

%create preamble chirps
for preamble = 1 : PL
    [chirp, phaseacc] = lora_chirpsignal(0, SF, 'up', ovs, phaseacc);
    iqdata = [iqdata chirp];
end

%create syncword chirps
[chirp, phaseacc] = lora_chirpsignal(floor(syncword/16)*8 , SF, 'up', ovs, phaseacc);
iqdata = [iqdata chirp];
[chirp, phaseacc] = lora_chirpsignal(mod(syncword,16)*8 , SF, 'up', ovs, phaseacc);
iqdata = [iqdata chirp];

%create 2.25 frame delimiter downchirps
[chirp, phaseacc] = lora_chirpsignal(0, SF, 'down', ovs, phaseacc);
iqdata = [iqdata chirp];
[chirp, phaseacc] = lora_chirpsignal(0, SF, 'down', ovs, phaseacc);
iqdata = [iqdata chirp];
[chirp, phaseacc] = lora_chirpsignal(0, SF, 'down_quarter', ovs, phaseacc);
iqdata = [iqdata chirp];

%create chirps for the input symbols
for sym = symbols
    [chirp, phaseacc] = lora_chirpsignal(sym, SF, 'up', ovs, phaseacc);
    iqdata = [iqdata chirp];
end
```

Chirp-Creation Matlab Implementation

```
function [chirp, phaseacc] = lora_chirpsignal(symbol, SF,
chirpmode, ovs, phaseacc)
%LORA_CHIRPSIGNAL creates a LoRa chirp
%   The function lora_chirpsignal creates a chirp representing
%   the input symbol with the spreading factor SF, oversampling
%   factor ovs, starting phase paseacc, and the chirpmode:
%   chirpmode = "UP"            upchirp
%   chirpmode = "DOWN"          downchirp
%   chirpmode = "DOWN_QUATER"   downchirp with 1/4 symbol length.

N = 2 ^ SF;
NN = N * ovs;
fMin = -pi / ovs;
fMax = pi / ovs;
fStep = (2 * pi) / (N * ovs * ovs);
f0 = (2 * pi * symbol) / NN;
f = fMin + f0;

%define length for mode "DOWN_QUATER"
if isequal(upper(chirpmode), 'DOWN_QUARTER')
    NN = NN / 4;
end
chirp = zeros(1, NN);
```

```
%create upchirp
if isequal(upper(chirpmode), 'UP')
    for idx = 1 : NN
        if f > fMax
            f =  f - (fMax - fMin);
        end
        phaseacc = phaseacc + f;
        chirp(idx) = exp(1i * phaseacc);
        f = f + fStep;
    end
end

%create downchirp
if isequal(upper(chirpmode), 'DOWN') || isequal(upper(chirpmode),
'DOWN_QUARTER')
    for idx = 1 : NN
        if f > fMax
            f =  f - (fMax - fMin);
        end
        phaseacc = phaseacc - f;
        chirp(idx) = exp(1i * phaseacc);
        f = f + fStep;
    end
end
```

Example

<u>Same Example with $SF = 11$:</u>

payload $\underline{p} = \begin{bmatrix} 0\text{x } 00 \\ 0\text{x } 00 \end{bmatrix}$

- ***Whitening***:

whitening sequence $\underline{w} = \begin{bmatrix} 0\text{x } FF \\ 0\text{x } FE \end{bmatrix}$

whitened payload $\underline{p_w} = \underline{p} \oplus \underline{w} = \begin{bmatrix} 0\text{x } FF \\ 0\text{x } FE \end{bmatrix}$

Example

<u>Same Example with $SF = 11$:</u>

- ***Encoding***: $L_h = SF - 2 = 9$

payload $\underline{p} = \begin{bmatrix} 0x\,00 \\ 0x\,00 \end{bmatrix}$

$$\underline{h} = \begin{bmatrix} 0x\,0 \\ 0x\,2 \\ 0x\,2 \\ 0x\,0 \\ 0x\,E \\ 0x\,F \\ 0x\,F \\ 0x\,E \\ 0x\,F \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \longrightarrow \underline{h_e} = \begin{bmatrix} 0x\,00 \\ 0x\,27 \\ 0x\,2A \\ 0x\,00 \\ 0x\,A6 \\ 0x\,FF \\ 0x\,FF \\ 0x\,E2 \\ 0x\,FF \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$\underline{p_w}$

whole payload data fits in remaining space in header

$$\underline{p_w} = \begin{bmatrix} 0x\,F \\ 0x\,F \\ 0x\,E \\ 0x\,F \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \longrightarrow \underline{p_w{}'} = 0 \longrightarrow \underline{p_e} = 0$$

Example

<u>*Same Example with $SF = 11$:*</u>

payload $\underline{p} = \begin{bmatrix} 0\text{x }00 \\ 0\text{x }00 \end{bmatrix}$

- ***Interleaving***:

$$\underline{h_e} = \begin{bmatrix} 0\text{x }00 \\ 0\text{x }27 \\ 0\text{x }2A \\ 0\text{x }00 \\ 0\text{x }A6 \\ 0\text{x }FF \\ 0\text{x }FF \\ 0\text{x }E2 \\ 0\text{x }FF \end{bmatrix} = \left.\overbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}}^{8\text{ bit}}\right\} 9\text{ bit} \longrightarrow \underline{h_i} = a_i = \begin{bmatrix} 0\text{x }471 \\ 0\text{x }3F0 \\ 0\text{x }C70 \\ 0\text{x }C30 \\ 0\text{x }650 \\ 0\text{x }F61 \\ 0\text{x }591 \\ 0\text{x }291 \end{bmatrix} = \left.\overbrace{\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}}^{9\text{ bit}}\right\} 8\text{ bit}$$

Same Example with $SF = 11$:

- ***Gray Mapping***:

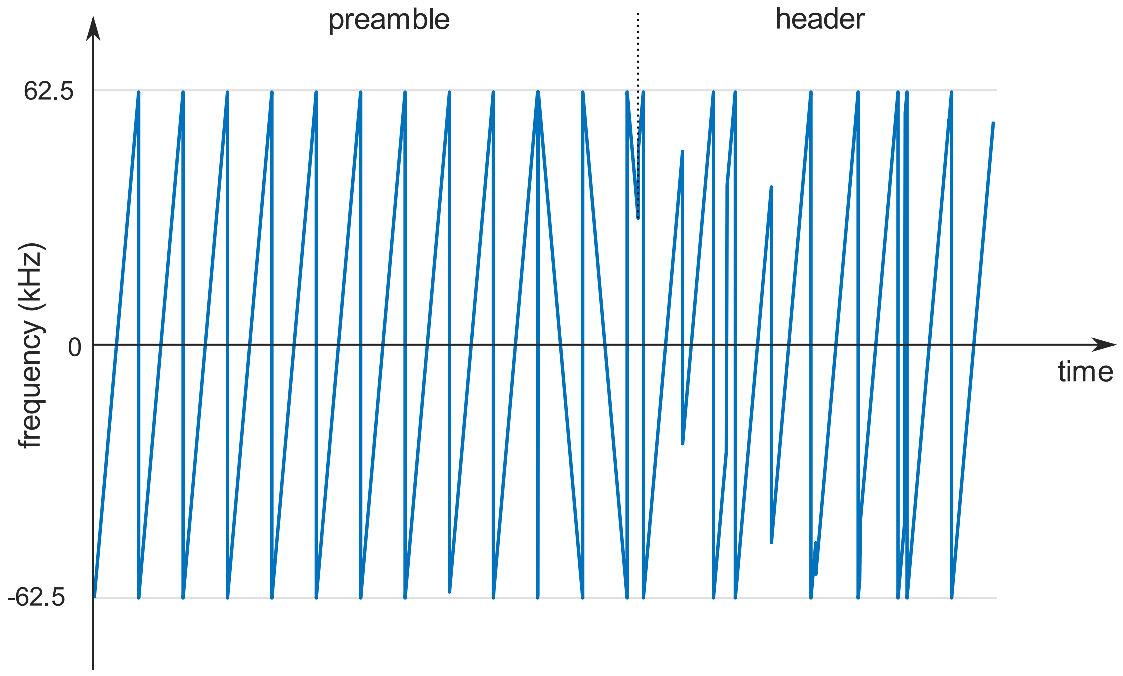
payload $\underline{p} = \begin{bmatrix} 0\text{x } 00 \\ 0\text{x } 00 \end{bmatrix}$

$$\underline{a_i} = \begin{bmatrix} 0\text{x } 471 \\ 0\text{x } 3F0 \\ 0\text{x } C70 \\ 0\text{x } C30 \\ 0\text{x } 650 \\ 0\text{x } F61 \\ 0\text{x } 591 \\ 0\text{x } 291 \end{bmatrix} \longrightarrow \underline{a_i} = \begin{bmatrix} 0\text{x } 7A1 \\ 0\text{x } 2A0 \\ 0\text{x } 750 \\ 0\text{x } 820 \\ 0\text{x } 460 \\ 0\text{x } 5B1 \\ 0\text{x } 911 \\ 0\text{x } C11 \end{bmatrix} = \begin{bmatrix} 1692 \\ 648 \\ 348 \\ 160 \\ 400 \\ 1748 \\ 1124 \\ 1136 \end{bmatrix}$$

Same Example with $SF = 11$*:*

payload $\underline{p} = \begin{bmatrix} 0\text{x } 00 \\ 0\text{x } 00 \end{bmatrix}$

- **Resulting signal:**



Contents

Airtime

- Payload Length

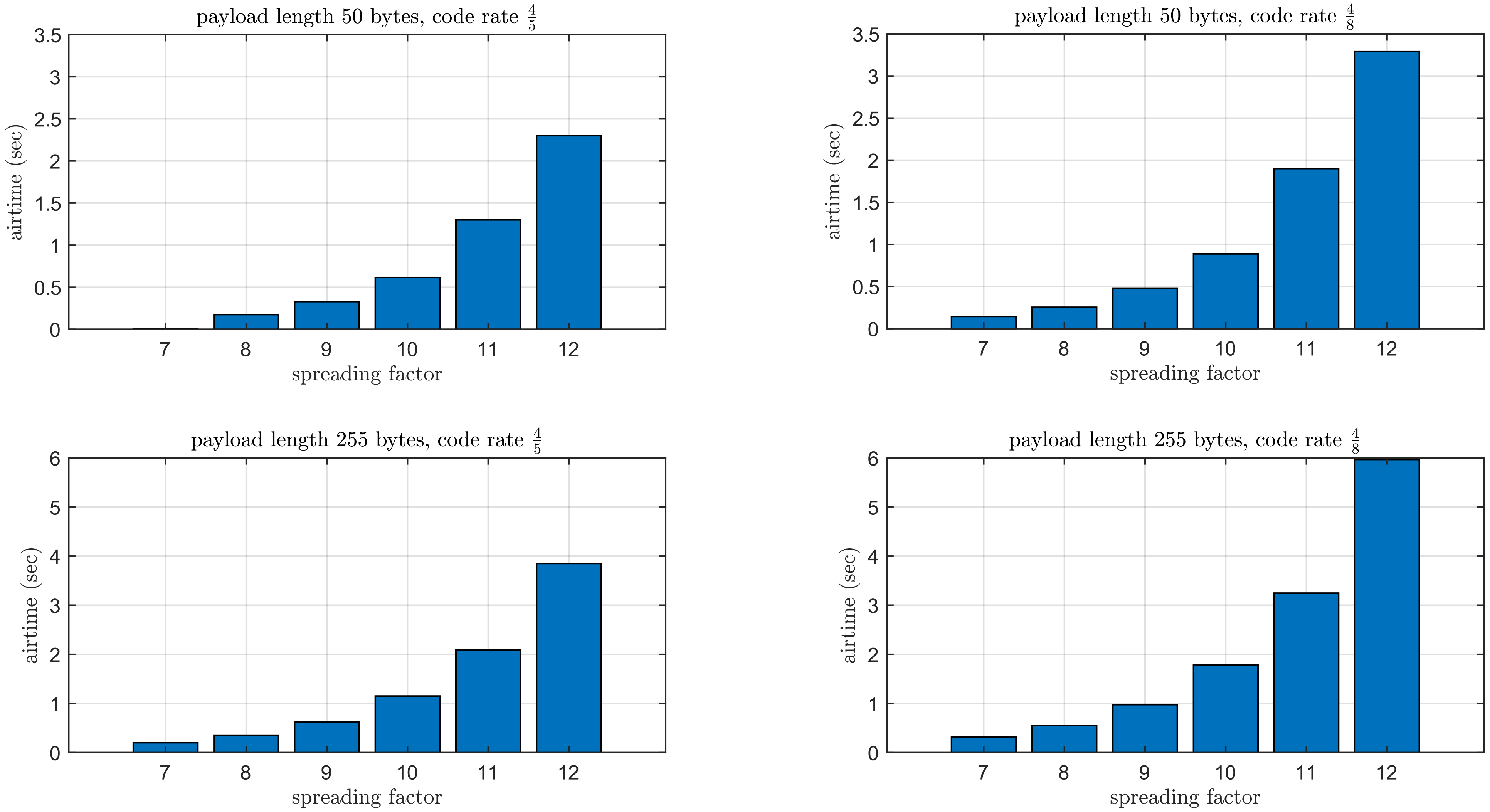
$$N_{payload} = 8 + max\left(\left\lceil \frac{(8\mathrm{PL}-4\mathrm{SF}+28+16\mathrm{CRC}-20\mathrm{H})}{4\mathrm{N}} \right\rceil \frac{4}{\mathrm{CR}+4}, 0\right)$$

  with:
  - PL Payload length
  - SF Spreading factor
  - CRC CRC-presence
  - H implicit mode $H = 1$, explicit mode $H = 0$
  - N Symbol alphabet size
  - CR Code rate ($CR = 1 \ldots 4$ representing code rates $4/5$, $4/6$, $4/7$, and $4/8$)

- Airtime of a LoRa signal:

$$T_{LoRa} = \left[N_{preamble} + N_{payload}\right] T_s$$

Airtime



payload length 50 bytes, code rate $\frac{4}{5}$
airtime (sec)
spreading factor
payload length 50 bytes, code rate $\frac{4}{8}$
airtime (sec)
spreading factor
payload length 255 bytes, code rate $\frac{4}{5}$
airtime (sec)
spreading factor
payload length 255 bytes, code rate $\frac{4}{8}$
airtime (sec)
spreading factor