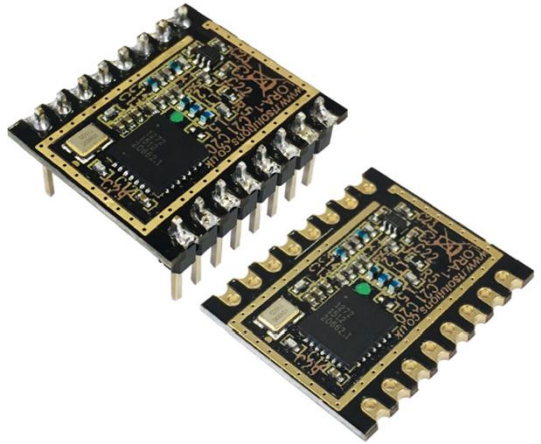
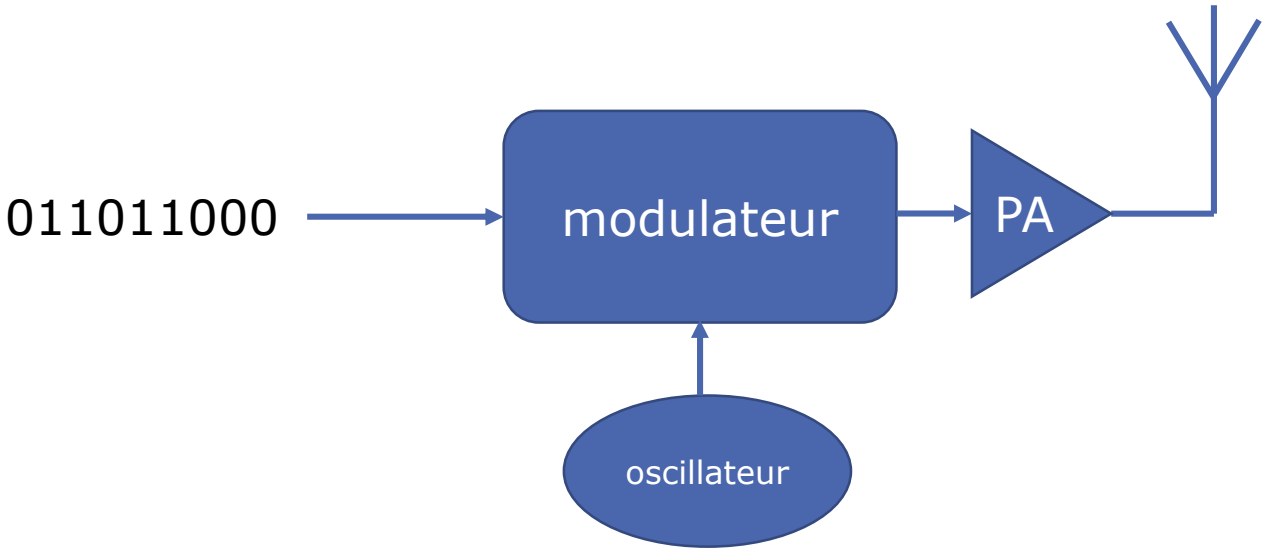
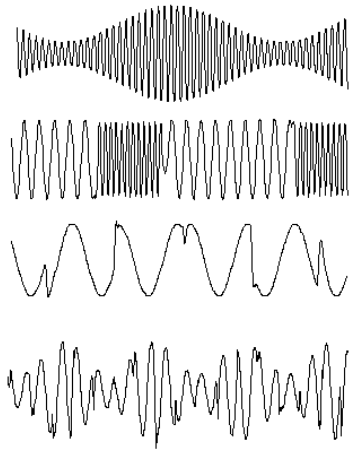
LoRa

Utiliser le module RF-LORA-868-SO et le circuit SX1272



Les transmissions numériques en deux mots…

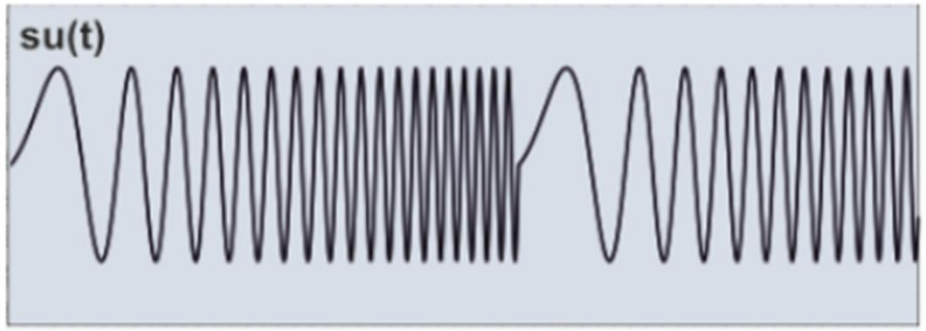




A chaque groupe de bits (symbole) correspond une combinaison particulière de phase et/ou fréquence et/ou amplitude du signal émis

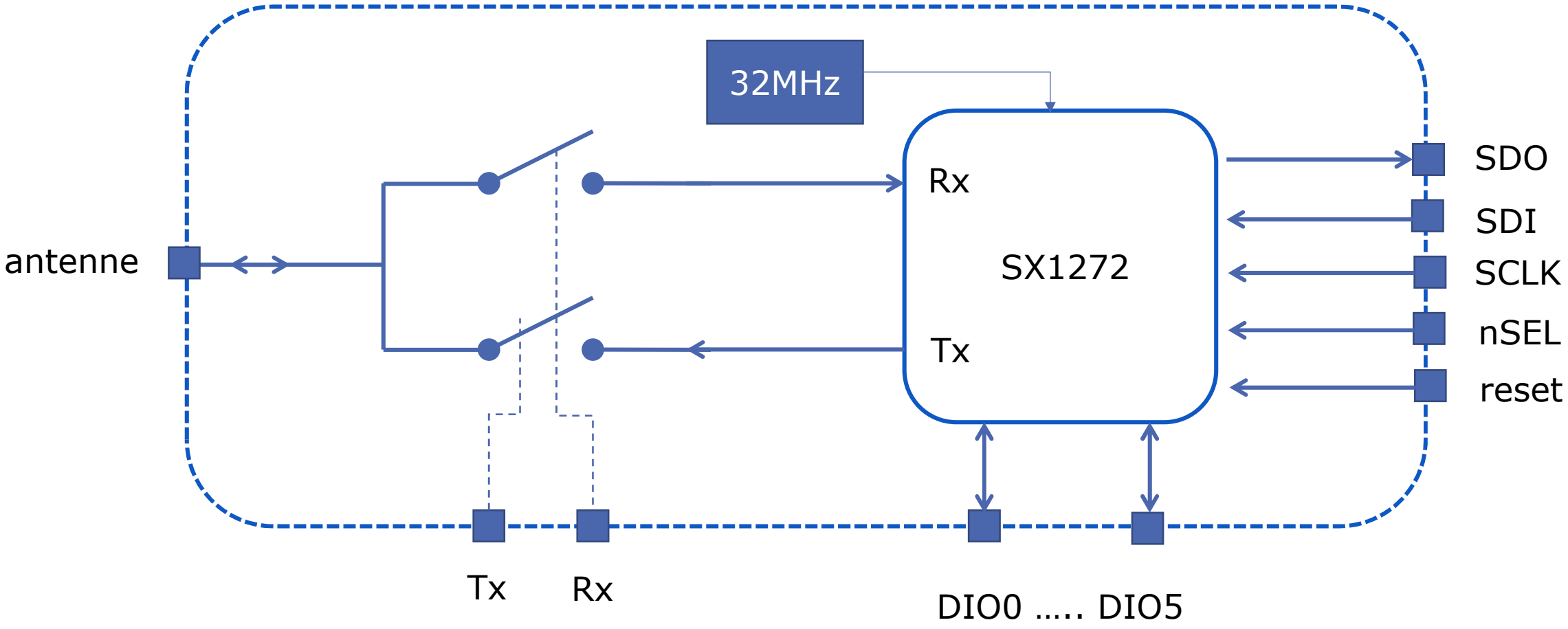
LoRa: modulation de fréquence par CHIRPs (Compressed High Intensity Radar Pulse), signal dont la fréquence varie linéairement en fonction du temps

Exemple (upchirp)

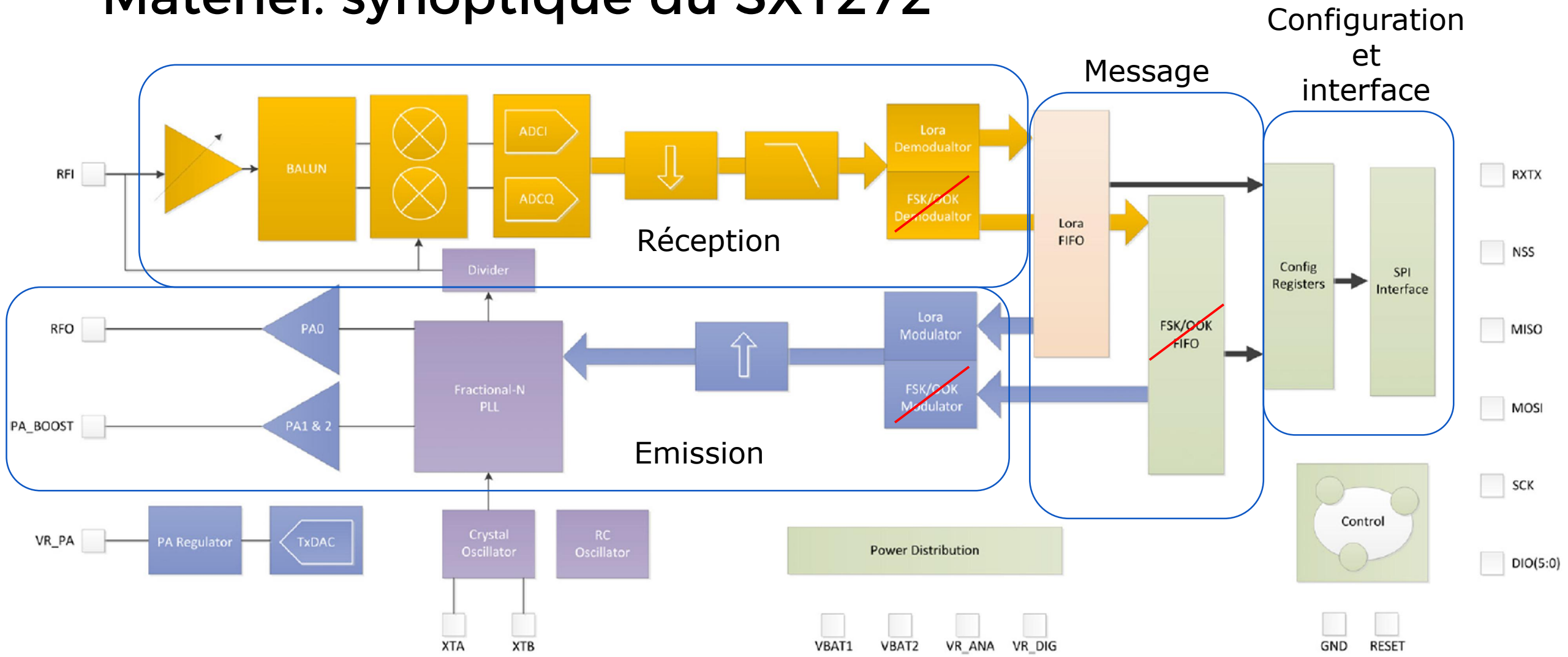


source: https://dautrylimoges.scenari-community.org/Sciences%20de%20l%27ingenieur/Cours_ModulationsNumeriques_diaporama/ModulationLoRa.html?mode=html

Matériel: synoptique du module RF-LORA-868-SO



Matériel: synoptique du SX1272



Matériel: configuration de la liaison (1)

- Bande de fréquence: en Europe, c'est la bande ISM autour de 868MHz
- Largeur de bande (**B**and**W**idth): définit l'encombrement spectral du signal.
  3 valeurs possibles: 125, 250 et 500kHz
- Facteur d'étalement (**S**preading **F**actor): nombre de bits par symbole.
  Valeurs possibles: 6, 7, 8, 9, 10, 11, 12
- Taux de codage (**C**ode **R**ate): données redondantes ajoutées au message original pour renforcer la robustesse de la transmission.
  Valeurs possibles: 4/5, 4/6, 4/7, 4/8

Débit (bit rate, R_b) : $R_b = SF \frac{BW}{2^{SF}} CR$ en bit/s

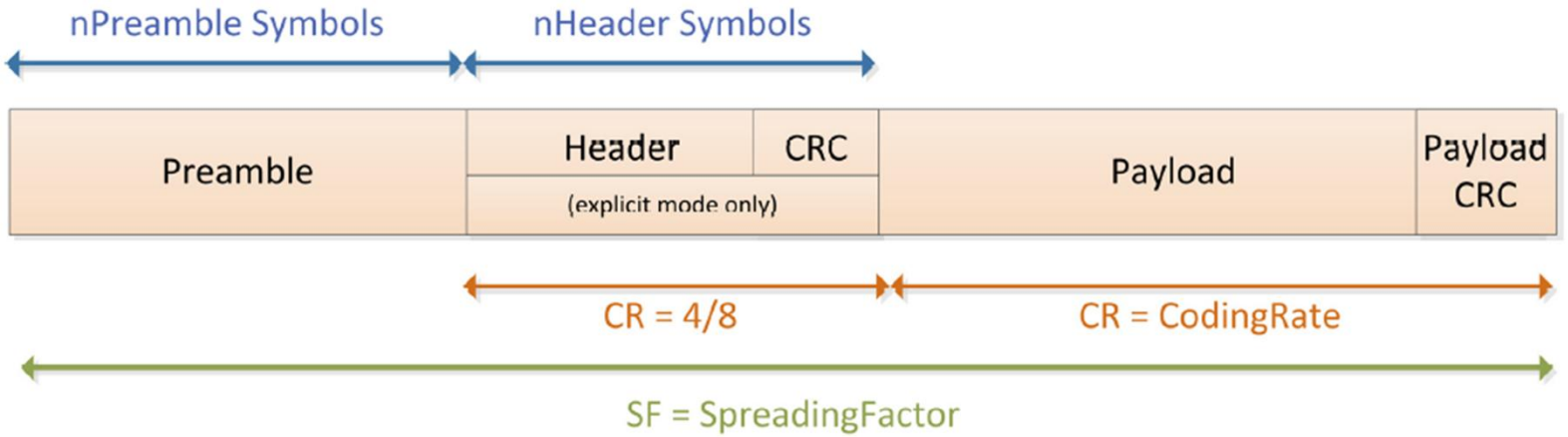
BW (kHz) = 125

SF	CR			
	4/5	4/6	4/7	4/8
6	9375	7813	6696	5859
7	5469	4557	3906	3418
8	3125	2604	2232	1953
9	1758	1465	1256	1099
10	977	814	698	610
11	537	448	384	336
12	293	244	209	183

JUNIA ISEN

Matériel: configuration de la liaison (2)

- Les données sont formatées de la façon suivante:



  - Préambule (Preamble): utilisé pour synchroniser le récepteur avec le flux de données reçu. Sa longueur est programmable (défaut = 12 symboles)
  - Entête (Header): contient certaines caractéristiques du paquet (nombre d'octets du message, valeur de CR, présence du CRC)
  - Message (Payload)

Matériel: configuration de la liaison (3)

- Deux modes d'entête possibles: implicite et explicite

Mode explicite: l'entête est transmise. Si le « payload CRC » a été transmis, on peut vérifier l'intégrité du message

Explicit Header	Transmitter	Receiver	CRC Status
Value of the bit RxPayloadCrcOn	0	0	CRC is not checked
	0	1	CRC is not checked
	1	0	CRC is checked
	1	1	CRC is chec

Valeur disponible dans le bit CrcOnPayload du côté récepteur

Mode implicite: l'entête n'est pas transmise, donc le nombre d'octets du message, la valeur de CR, la présence du CRC doivent être fixés d'avance et cohérents du côté transmetteur et récepteur

Implicit Header	Transmitter	Receiver	CRC Status
Value of the bit RxPayloadCrcOn	0	0	CRC is not checked
	0	1	CRC is always wrong
	1	0	CRC is not checked
	1	1	CRC is checked

JUNIA ISEN

Matériel: configuration de la liaison (4)

- Choix du gain pour l'amplificateur faible bruit d'entrée (LNA): 6 valeurs possibles

  gain ↑ alors SNR ↑ mais risque de saturation de l'étage d'entrée

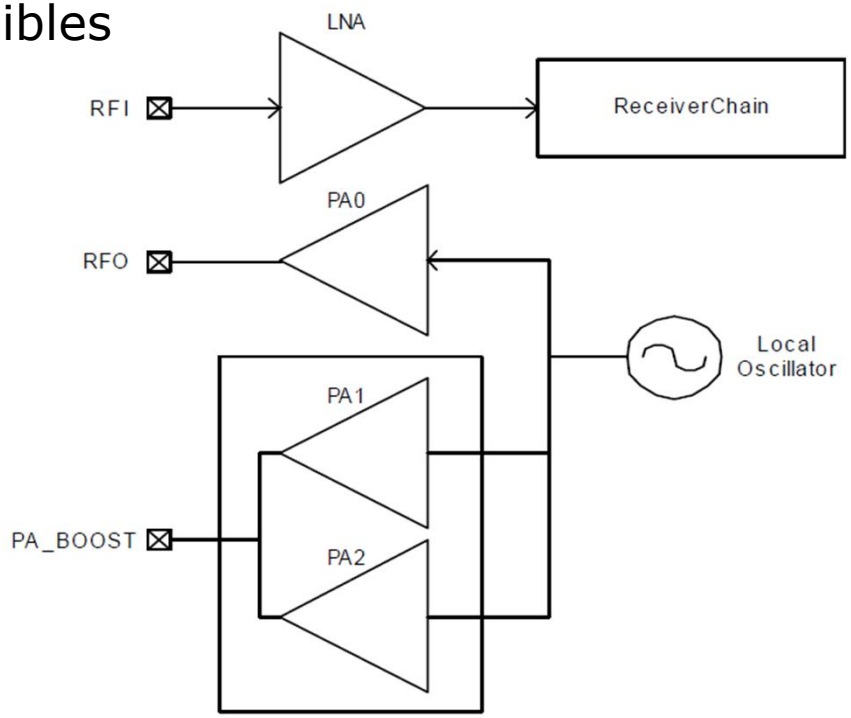
- Choix du courant consommé par le LNA: 2 valeurs possibles

  courant ↑ alors bruit ↓

- Choix de la sortie active: RFO ou PA_BOOST
- Réglage de la puissance d'émission par pas de 1 dB:

  Sur RFO entre -1 dBm et +13 dBm

  Sur PA_BOOST entre +2 dBm to +17 dBm



$$dBm = 10 \log \frac{Puissance}{1mW}$$

0 dBm = 1mW

Matériel: configuration de la liaison (5)

- Compromis en terme de débit binaire (Rb)

  SF élevé → Rb faible
  BW faible → Rb faible
  CR faible (par exemple 4/8 plutôt que 4/5) → Rb faible

- Compromis en terme de robustesse de la transmission

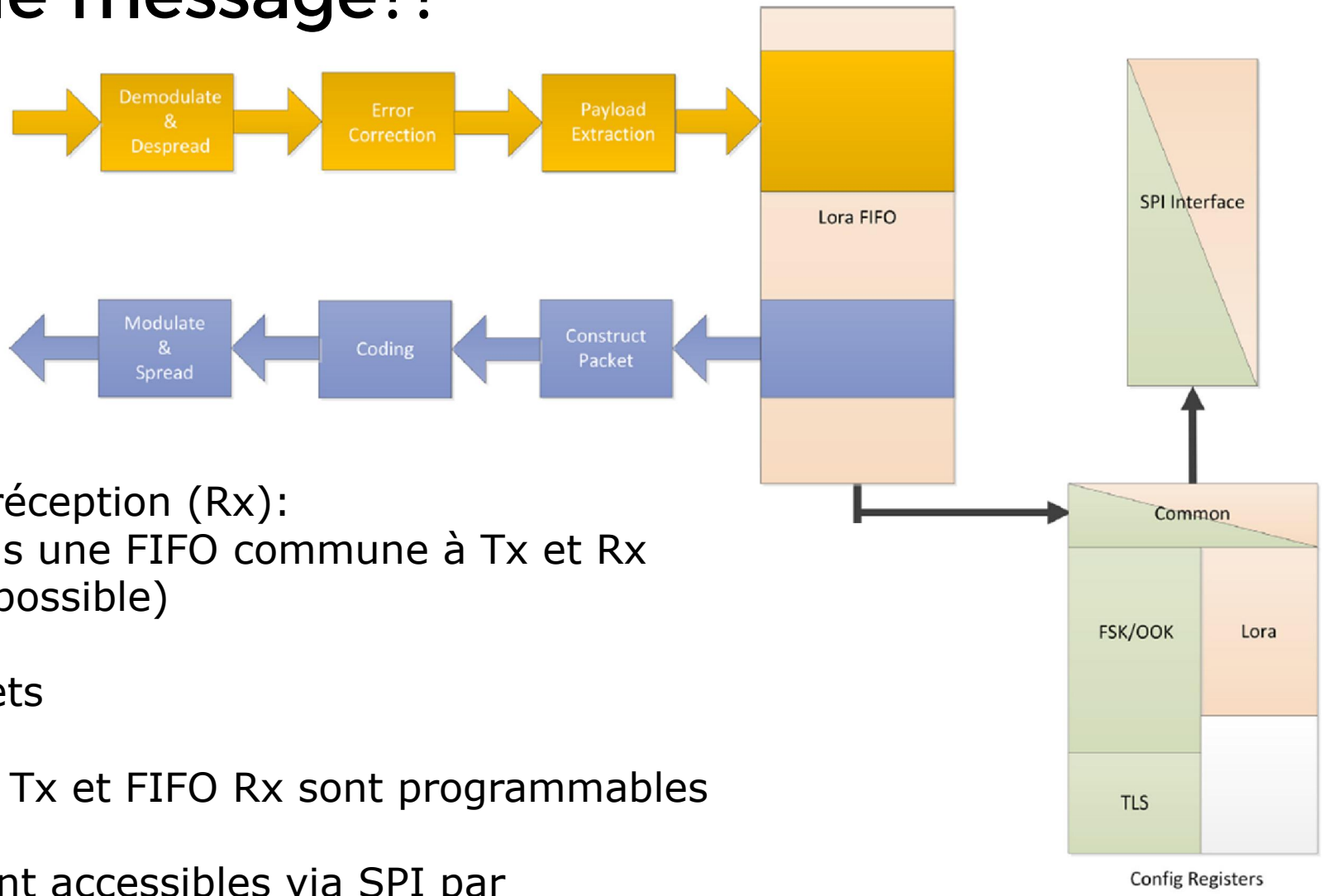
  CR faible → robustesse élevée

- Compromis en terme de distance de transmission

  Rb faible (= SF élevé et CR faible et BW faible) → distance élevée
  Gain du LNA élevé → distance élevée
  Puissance d'émission élevée → distance élevée

Favoriser distance et robustesse se fait au détriment du débit binaire

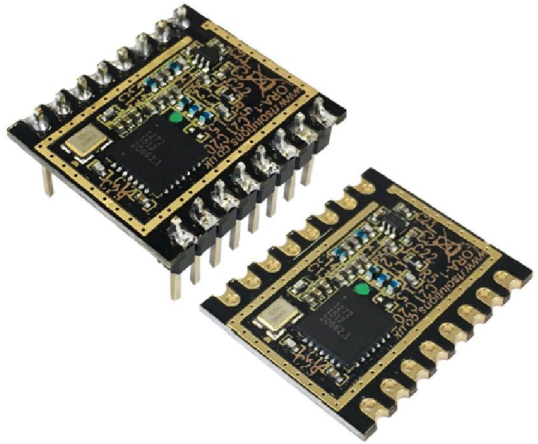
Matériel: où est le message??



En émission (Tx) comme en réception (Rx):

- Le message est stocké dans une FIFO commune à Tx et Rx (écrasement des données possible)
- La FIFO comporte 256 octets
- Les adresses de base FIFO Tx et FIFO Rx sont programmables
- Les données (Tx ou Rx) sont accessibles via SPI par lecture/écriture du registre RegFifo

Matériel: considérations pratiques (2)



- Le module fonctionne sous **3.3V**
  → idéalement, tout alimenter en 3.3V
  → dans le cas contraire, ne pas oublier les translateurs de tension

- Les broches Tx et Rx commandent le commutateur d'antenne et n'ont rien à voir avec le Tx-Rx du module UART du microcontrôleur

- Les broches Tx et Rx ne doivent **jamais** être au niveau 1 simultanément (on prendra ses précautions au niveau logiciel également)

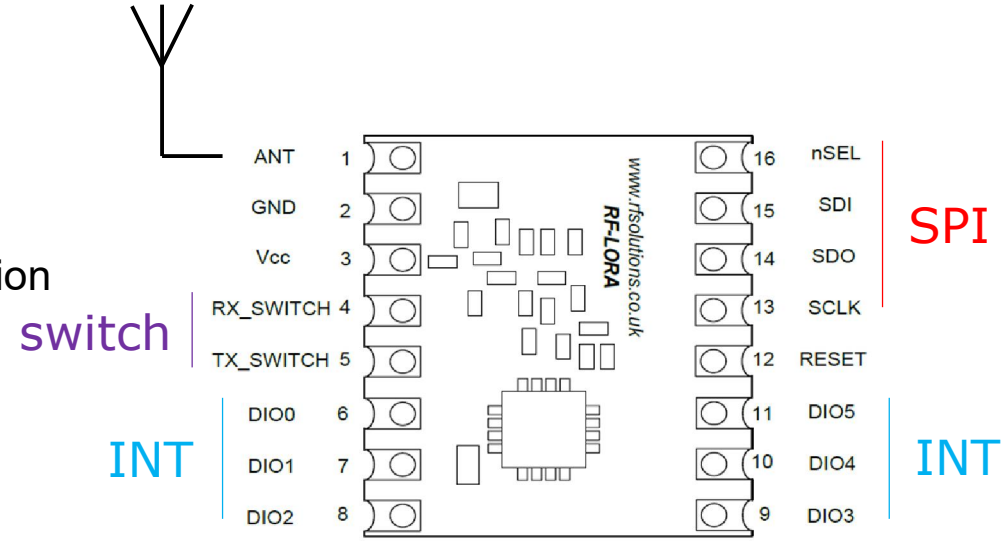
- Prévoir une antenne quart d'onde (i.e. de longueur $\lambda/4$)

- Les fréquences autorisées sont dans la bande 868MHz

- Le module peut être amené à consommer jusqu'à 240mA en émission

- Eviter d'émettre à forte puissance sans antenne

- Broches DIOx → interruptions sur évènements (reconfigurables)



Logiciel: lire/écrire dans un registre

Règles générales:

- Accès via l'interface SPI: CPOL=0, CPHA=0, MSB first, SCLK@10MHz max

- Type d'accès défini par le MSB de adresse du registre: MSB=1 → écriture, MSB=0 → lecture
  Exemple: RegFifo a pour adresse de base 0x00
  En lecture, l'adresse reste 0x00
  En écriture, l'adresse devient 0x80

Registres de configuration:

- Mode SINGLE: transmission de deux octets
  Premier octet = adresse
  Octet suivant = donnée

- Mode BURST: transmission de plusieurs octets
  Premier octet = adresse de départ
  Octets suivants = données
  Incrémentation automatique de l'adresse, donc les adresses sont **forcément** consécutives

Logiciel: lire/écrire dans un registre

FIFO:

- Semblable au mode BURST: un octet d'adresse, un ou plusieurs octets de données
- il faut mettre la bonne valeur dans le pointer RegFifoAddrPtr avant d'accéder à la FIFO
- L'accès se fait uniquement via RegFifo que ce soit en lecture ou en écriture

Ecriture (Tx):

RegFifoTxBaseAddr → RegFifoAddrPtr

Payload_length → RegPayloadLength

```
for (i = 0; i < Payload_length ; i++) {
    TxBuffer(i) → RegFifo
  }
```

Lecture (Rx):

RegFifoRxCurrentAddr → RegFifoAddrPtr

RegRxNbBytes → Payload_length

```
for (i = 0; i < Payload_length ; i++) {
 RegFifo → RxBuffer(i)
  }
```