PE1HVH

Van Tekst naar Chirp

Hoe LoRa data codeert

Een stap-voor-stap uitleg van bits naar radiosignaal

Versie 1.0 - Januari 2026

Door PE1HVH / ChatGPT / Claude.ai

Inhoudsopgave

1. Je wilt "Test" versturen

Laten we beginnen met iets concreets: je wilt het woord "Test" versturen via LoRa. Hoe wordt dat een radiosignaal?

1.1 Letters worden bits

Elke letter heeft een ASCII-code, en die code is een getal dat we als bits kunnen schrijven:

Letter	ASCII	Binair
T	84	01010100
e	101	01100101
s	115	01110011
t	116	01110100

Samen is "Test" dus 32 bits.

2. Bits worden symbolen

Nu komt de Spreading Factor (SF) in beeld. Bij SF12 groeperen we bits in groepjes van 12.

2.1 Waarom 12 bits?

SF12 betekent: elk symbool draagt 12 bits informatie. Die 12 bits vormen samen een getal van 0 tot 4095 (want $2^{12} = 4096$ mogelijke waarden).

2.2 "Test" opsplitsen in symbolen

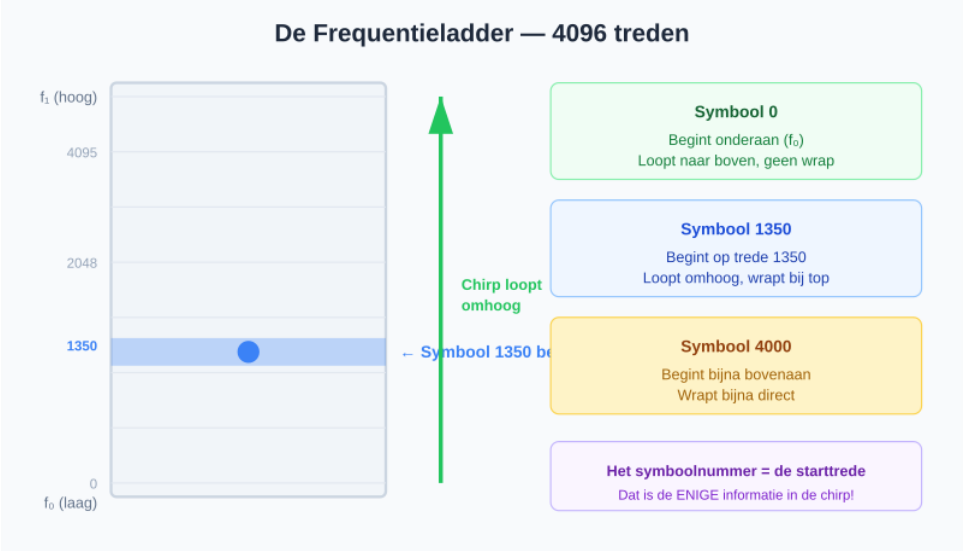
We pakken onze 32 bits en verdelen ze in groepjes van 12. Het woord "Test" wordt zo drie symbolen: 1350, 1395 en 1860.

3. Symbolen worden chirps

Nu de cruciale stap: hoe wordt een getal (bijvoorbeeld 1350) een radiosignaal?

3.1 De frequentieladder

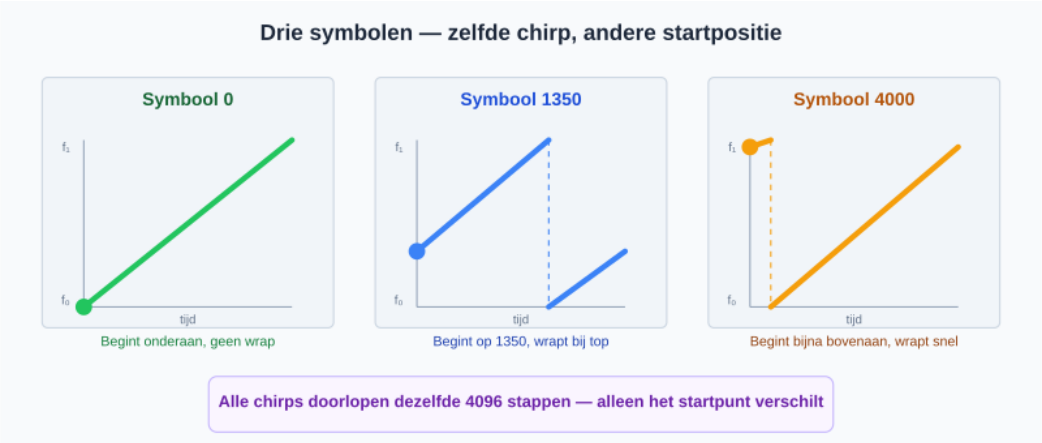
Stel je de 125 kHz bandbreedte voor als een ladder met 4096 treden. Elk symboolnummer correspondeert met een trede op die ladder:



Figuur 1: De frequentieladder — het symboolnummer bepaalt de starttrede

3.2 De chirp loopt de ladder op

Een chirp begint op zijn startpositie en loopt dan alle treden af, omhoog. Bij de top wrapt hij naar beneden en gaat verder tot hij weer bij zijn startpunt is. Elke chirp doorloopt altijd ALLE 4096 treden.



Figuur 2: Drie symbolen — zelfde chirp-vorm, andere startpositie

4. De complete keten

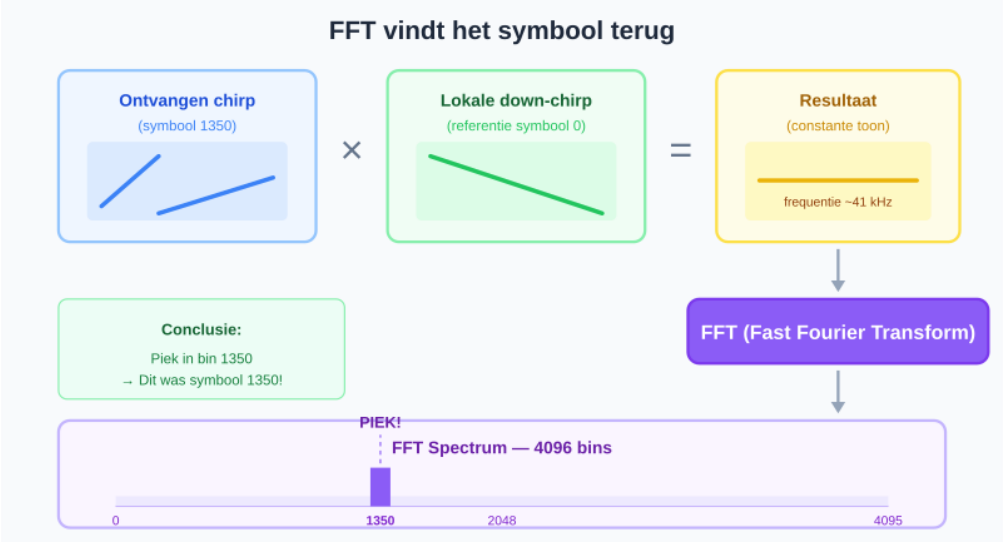
Laten we alles samenvoegen in één overzicht:

Figuur 3: Van "Test" naar chirps — de complete encoding keten

5. Hoe weet de ontvanger welk symbool het was?

De ontvanger doet een slimme wiskundige truc: hij vermenigvuldigt de ontvangen chirp met een lokaal gegenereerde omgekeerde chirp (down-chirp).

Stijgende frequentie × dalende frequentie = constante toon. De hoogte van die toon hangt af van waar de originele chirp begon!



Figuur 4: FFT vindt het symbool terug — de piek-positie IS het symboolnummer

De FFT (Fast Fourier Transform) analyseert de toon en geeft een spectrum met 4096 bins. De bin waar de piek zit, is direct het symboolnummer.

5.1 Waarom een constante toon?

De chirp zelf is een sweep (frequentie verandert continu), geen constante toon. Maar bij de vermenigvuldiging gebeurt iets bijzonders:

Signaal	Karakter
Ontvangen chirp	Sweep omhoog (laag → hoog)
Lokale down-chirp	Sweep omlaag (hoog → laag)
Resultaat	**Constante toon**

Wanneer je een stijgende frequentie vermenigvuldigt met een dalende frequentie, heffen de veranderingen elkaar op. Het resultaat is een constante toon — en de frequentie van die toon hangt af van het faseverschil tussen de twee chirps.

Wiskundig:

- Ontvangen: frequentie stijgt met +X Hz per sample
- Lokaal: frequentie daalt met -X Hz per sample
- Som: +X + (-X) = 0 verandering → constante frequentie

De hoogte van die constante toon hangt af van waar de ontvangen chirp begon (het symboolnummer). Symbool 1350 geeft een andere constante toon dan symbool 1351.

5.2 Fouttolerantie: Processing Gain

Wat als er een stukje van de chirp verloren gaat door interferentie of ruis? Stel: van de 4096 samples mis je er 100 aan het begin.

Omdat de FFT niet de chirp zelf meet, maar de constante toon die ontstaat ná de vermenigvuldiging:

- Je hebt nog steeds 3996 samples van dezelfde constante toon
- De FFT analyseert die toon en vindt dezelfde piek
- De piek is iets zwakker (minder energie), maar zit op exact dezelfde positie

Het is alsof je een liedje herkent, ook al heb je de eerste seconde gemist. De melodie is hetzelfde, je hebt alleen iets minder “bewijs” gehoord. De positie van de FFT-piek is dus inherent robuust tegen partieel signaalverlies — dit noemen we **Processing Gain** of **Spreading Gain**.

5.3 De rol van de preamble bij synchronisatie

Maar hoe weet de ontvanger eigenlijk waar die eerste 100 samples hadden moeten zijn? Dit is waar de preamble cruciaal wordt.

Het synchronisatieproces:

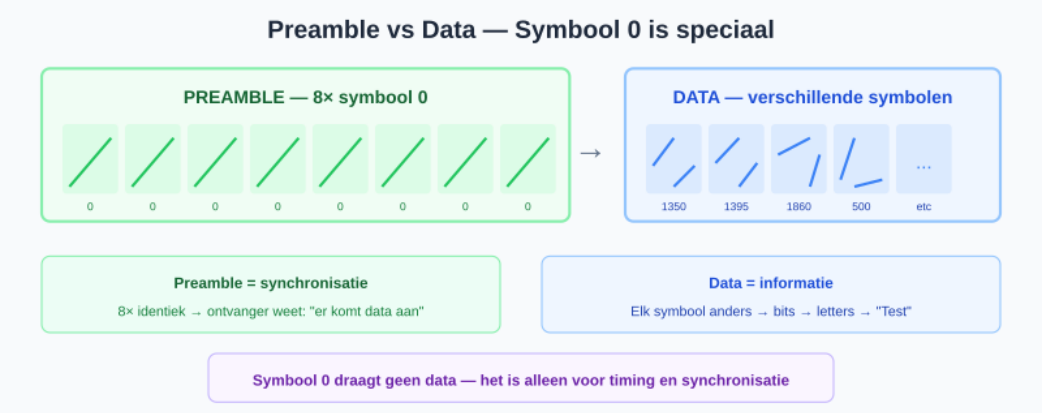
- **Preamble detectie** — De ontvanger ziet eerst 8 identieke symbool-0 chirps
- **Timing lock** — Door deze herhaalde, identieke chirps weet de ontvanger exact wanneer elke chirp-periode begint en eindigt
- **Verwachtingsvenster** — Voor elk volgend datasymbool weet de ontvanger: “de chirp loopt van sample X tot sample X+4096”

De preamble fungeert dus als een kloksignaal. Zonder die synchronisatie zou de ontvanger niet eens weten waar een chirp begint of eindigt. Met de synchronisatie weet hij precies welke samples bij welke chirp horen — ook als sommige samples verstoord zijn.

Als de eerste 100 samples van een chirp door ruis zijn verstoord, weet de ontvanger door de preamble-synchronisatie dat die 100 samples er hadden moeten zijn. Hij analyseert gewoon de 3996 samples die hij wél goed heeft ontvangen, en de FFT vindt nog steeds dezelfde constante toon op dezelfde positie. De verstoorde samples verschuiven het symboolnummer niet — ze verzwakken alleen de piek.

6. Waarom symbool 0 speciaal is

Symbool 0 is de referentie — een chirp die begint op de laagste frequentie en netjes eindigt op de hoogste, zonder wrap. Dit symbool wordt gebruikt in de preamble voor synchronisatie.



Figuur 5: Preamble (8× symbool 0) vs Data (verschillende symbolen)

De ontvanger ziet 8 identieke chirps en weet: "er komt data aan, ik ben gesynchroniseerd". Daarna komen de data-symbolen, elk met een andere startpositie.

7. Samenvatting

De kern van LoRa modulatie:

- Tekst wordt bits, bits worden gegroepeerd in symbolen (12 bits bij SF12)
- Elk symbool is een getal van 0-4095
- Dat getal bepaalt waar de chirp begint in de frequentiesweep
- De ontvanger vindt het symbool terug via FFT

Het maakt niet uit of het symbool 0, 1350, of 4000 is — de chirp doorloopt altijd alle 4096 stappen. Alleen het startpunt verschilt, en dat startpunt IS de data.

— Einde Document —