Progress report: Implementation & Integration

Boyan & Mihai

2026-02-08

LoRaWAN with RadioLib

- Decided to use **RadioLib** for MAC layer to not reinvent the wheel
- ABP activation over SX1278 on our custom PCB (and untested OTAA)
- Full uplink/downlink cycle through our HackRF SDR gateway
- Stack: Node → HackRF (SDR) → ChirpStack → MQTT

Channel access: a very primitive TDMA

- No hardware clock sync between nodes
- Simple slotted schedule: each node assigned a fixed offset

$$t_{\text{tx}} = t_0 + n \cdot T_{\text{interval}} + \text{slot} \cdot \frac{T_{\text{interval}}}{N_{\text{slots}}}$$

- Currently $T_{\text{interval}} = 60\text{s}$, $N_{\text{slots}} = 2$
- Rudimentary but works - nodes must be started roughly together

Proportional-Derivative controller

Per-node PD loop controlling TX power and spreading factor:

$$e_k = \mathrm{SNR}_{\mathrm{target(SF)}} - \mathrm{SNR}_{\mathrm{measured}}$$

$$\Delta P = K_p \cdot e_k + K_d \cdot (e_k - e_{k-1})$$

Parameters: $K_p = 0.5$, $K_d = 0.1$, $P \in [2, 17]$ dBm

- If link is stressed ($\mathrm{SNR} < \mathrm{target}$) $\rightarrow$ SF $\uparrow$, TX $\rightarrow P_{\max}$
- Else for ≥ 3 consecutive frames $\rightarrow$ SF $\downarrow$

PD controller: Examples

Example 1: SF7, target SNR = 7.5 dB, measured = 5 dB, prev error = 0

$$e = 7.5 - 5 = 2.5 \rightarrow \Delta P = 0.5 \cdot 2.5 + 0.1 \cdot (2.5 - 0) = 1.5 \text{ dBm}$$

Example 2: SF8, target SNR = 5 dB, measured = 0 dB

$$e = 5 - 0 = 5 \rightarrow \text{SNR} < \text{target (stress)} \rightarrow \text{SF} \uparrow \text{to SF9}, P \rightarrow P_{\max}$$

PD state reset[1]: $e_{k-1} := 0$, start optimising from $P = 17$ dBm

[1] Resets the error term e to 0

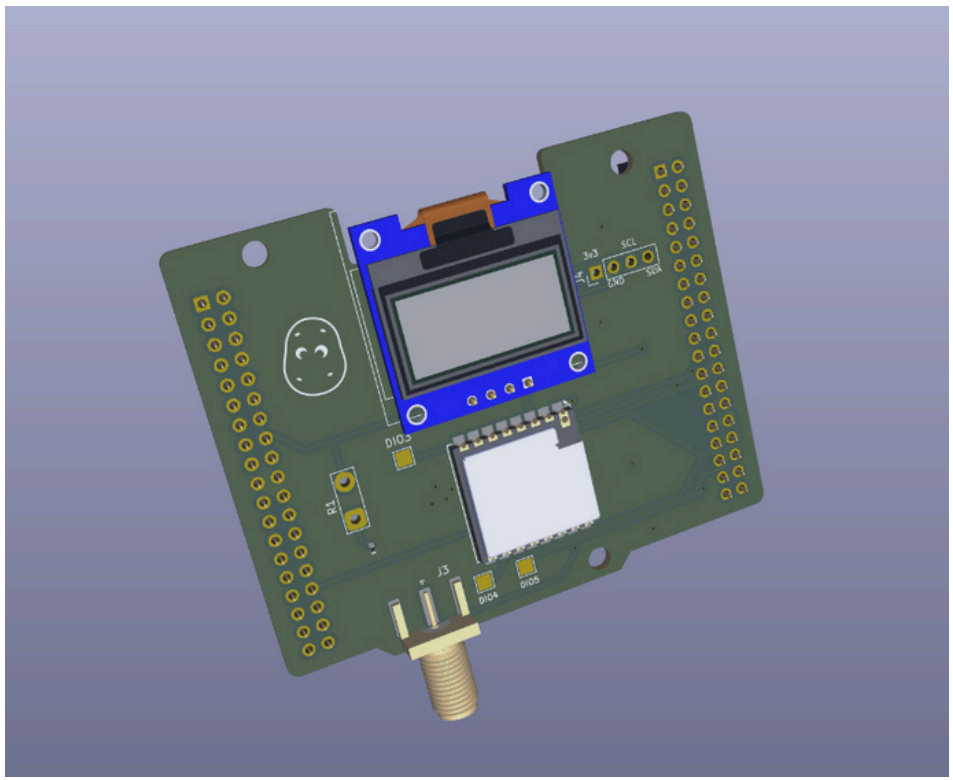
System architecture

$$\text{Node} \xrightarrow{\text{LoRa}} \text{HackRF GW} \xrightarrow{\text{UDP}} \text{ChirpStack} \xrightarrow{\text{MQTT}} \text{PD Controller} \xrightarrow{\text{gRPC}} \text{Downlink}$$

- Everything runs on a single HackRF - **half-duplex, one channel**
- This constraint shaped every design decision

Node (STM32 + SX1278)

- Custom PCB with BME280 sensor + SX1278 LoRa
- Single channel means collisions with >1 node
- Primitive TDMA – fixed time slots per node



HackRF SDR Gateway

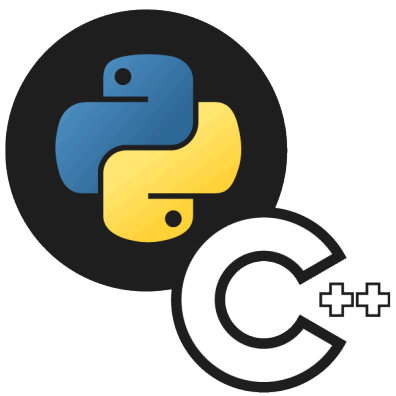
- GNU Radio flowgraph: demod LoRa → Semtech UDP
- Half-duplex – can’t RX and TX simultaneously
- Pre-computed TX timing offset
- No frequency hopping – one channel only
- ChirpStack configured for single-channel EU433
- Scans SF7-12 for every packet to receive everything

ChirpStack Network Server

- Standard LoRaWAN network server
- Handles MAC layer, device management, data routing
- ADR commands use standard algorithm - not ours
- Disabled built-in ADR, our PD controller sends MAC commands directly via gRPC API
- Publishes uplinks over MQTT for our controller to consume

PD Controller

- C library (libalgo) + Python wrapper
- Subscribes to MQTT uplinks, computes commands, pushes via gRPC
- Per-node state tracked with uthash
- Downlink only works 50% of the time (half-duplex)
- Controller is idempotent - repeats commands until acknowledged
- SF changes reset SNR baseline
- Reset PD state on SF transition, start from $P_{\max}$



Current status

- End-to-end LoRaWAN uplink + downlink working
- PD controller integrated and responding to uplinks
- Two physical nodes running concurrently
- OLED display on nodes showing live status
- No field/performance results yet – coming very soon

NS3 simulation

- Set up NS3 LoRaWAN simulation environment
- Uplink and downlink working - almost complete
- We need to validate ADR algorithm at scale alongside field tests

TODOs (done by Wednesday)

- Make TDMA reliable - try sync mechanisms other than “start at the same time”
- Collect results with the PD controller
- Run NS3 simulations and compare with physical measurements
- Update the final report

Note: development of the core system took priority over building baseline algorithms to compare against. Comparative results may be limited.

Demo!