Funkwetter (NRW) – VHF/UHF/HF “Radio Weather” Display

A small ESP8266 project that estimates **VHF/UHF ducting quality** (2 m, 70 cm) and **10 m band openness**, shows it on an **SSD1309 128×64 OLED**, and serves a **web UI** to tweak inputs and see raw/derived values.
It can run purely on **local sensors** *or* enrich results with **solar activity APIs** (NOAA/SWPC). A **second ESP8266** with **BMP280 + DHT22** can act as a remote sensor publisher.

Features

- **Bands**: 2 m, 70 cm (from tropospheric refractivity gradient), 10 m (on-air proxy or solar fallback)
- **Display**: 4 progress bars (2 m / 70 cm / 10 m / Solar), header icons (Wi-Fi + sun/moon), bottom **ticker**
- **Overlay**: periodic **RAW sensor overlay** (configurable: enable, interval, duration)
- **Web UI** (`/local`): enter/measured values, scaling (0–5 or 1–10), today override, solar/flare toggles, OLED contrast, remote host settings, overlay timing
- **API** (`/json`): machine-readable summary
- **Persistence**: all settings in EEPROM
- **Time**: NTP + local sunrise/sunset and day/night handling
- **mDNS**: reachable as `http://funkwetter.local/`

What you need

Main node (Display + Web)

- **ESP8266 NodeMCU (ESP-12E)**
- **SSD1309 128×64 OLED (SPI, 4-wire)**
  Pins used in the sketch:
    - CS → **D1 (GPIO5)**
    - DC → **D2 (GPIO4)**
    - RST → **D0 (GPIO16)**
    - SCK → **D5 (GPIO14)**
    - MOSI → **D7 (GPIO13)**
    - VCC → 3V3, GND → GND

- Optional local sensors (if you later want direct measurements):
    - Pressure/temperature (e.g., BMP280/BME280)
    - Humidity (e.g., DHT22 or from BME280)
    - A second height level (barometric/temperature/humidity) if available

Remote sensor node (optional)

- **ESP8266 NodeMCU**
- **BMP280 (I²C)**
    - VCC → 3V3, GND → GND
    - SCL → **D1 (GPIO5)**, SDA → **D2 (GPIO4)**
    - I²C Address: typically **0x76** (SDO→GND) / **0x77** (SDO→3V3)
- **DHT22**
    - VCC → 3V3, GND → GND, DATA → **D6 (GPIO12)**
- Publishes `http://<host>/sensors.json` with:
  `{"P_hPa":1018.3,"T_C":22.4,"RH_pct":50.0,"z_m":120.0}`

Software & Libraries

- **Arduino IDE** with **ESP8266 core** installed via Boards Manager
- **U8g2** (for SSD1309 OLED)
- (Remote node) **Adafruit BMP280** (or any BMP280 lib) + **DHT** library

On the main node we use the ESP8266 core's BearSSL for HTTPS calls (SWPC APIs).

How it works (math & logic)

Refractivity and tropospheric ducting (2 m / 70 cm)

1. From pressure P [hPa], temperature T [°C], and relative humidity RH [%], compute **radio refractivity** N:

$$e = RH/100 \cdot Es(T), Es(T) = 6.112 \cdot \exp(243.12 + T17.62T)\ N = 77.6TKP + 3.73 \cdot 105TK2e, TK = T + 273.15$$

2. With two heights z1,z2, compute **gradient**:

$$dzdN = z2 - z1N2 - N1 \cdot 1000[N/km]$$

3. Heuristics for ducting:

- **Super-refraction/ducting** likely if $dN/dz < -157$ N/km → "Tropo possible"
- **Enhanced range** if $-157 \le dN/dz < -79$

- Else **normal**

4. Map to **scores** (configurable 1–10 or 0–5):
    - Construct a normalized score from dN/dz and scale slightly differently for 2 m vs 70 cm (UHF a tad more sensitive).
    - If you only have one height, a fallback **heuristic** uses pressure anomaly, night factor, and RH proximity to ~60 %.

10 m band

- **Primary (on-air proxy)** if provided:
    - **Spots/15 min**, **max DX distance**, **median SNR** → weighted blend → score
- **Fallback** if not provided:
    - Use **F10.7 solar flux** and **Kp** (geomagnetic) + **time-of-day** + **season (equinox proximity)** to estimate MUF trends → score
- Optional **solar adjustment**: very high solar score nudges +2/+1; high Kp (>5) subtracts 1.

Solar activity & flares

- **Solar score (SOL)** from **F10.7** (↑ is better) and **Kp** (↓ is better).
- **Flares (GOES X-ray)**: if a C/M/X flare is current, apply a **penalty** primarily to 10 m (especially **daytime**). Values and class are shown.

Disturbance probability (%)

- Combines **Kp** and any active **flare penalty** (weighted); reported as **low / medium / high**.

Web UI and endpoints

- `http://funkwetter.local/` → redirects to `/local`
- **`/local`** (HTML):
    - Enter or edit: P/T/RH/z (two heights), 10 m on-air proxy (spots/dx/snr)
    - Toggles: Solar, Solar-adjust-10m, Flares (day-only option), VHF heuristic, Splash
    - Scale: 1–10 or 0–5
    - Today override: manually set VHF score (e.g., 8/10 for a great day)
    - OLED contrast slider
    - **Overlay settings**: enable, interval (s), duration (ms)
    - **Remote sensor**: enable + host (e.g. `funk-remote.local`)
- **`/json`** (machine readable): all inputs + scores + labels + flags

- `/reset`: factory default (and clears EEPROM state)

Display UX

- **Top row**: “Funkwetter”, Wi-Fi icon, sun/moon (day/night), clock (HH:MM)
- **Bars**: 2 m, 70 cm, 10 m, Solar (with a soft animated “shine”)
- **Ticker** (bottom line): readable ASCII text with **band scores + labels**, solar values, disturbance %, **H1/H2** raw readings, **N1/N2/dN/dz/k**, 10 m proxy/fallback info, and remote status.

Overlay screen (periodic, configurable): a compact **RAW/Sensor** view while the ticker continues to scroll.

Build & Flash

1. **Wire the OLED (SPI)** to the ESP8266 as shown above.
2. Install **ESP8266 core** in Arduino IDE; install **U8g2** library.
3. Open the main sketch and set:
    - `WIFI_SSID`, `WIFI_PASS`, `HOSTNAME`
    - Your coordinates: `LAT`, `LON` (used for sunrise/sunset)
    - Optionally change default overlay timing, score scale, etc.
4. **Board**: *NodeMCU 1.0 (ESP-12E Module)*, CPU 80 MHz (default fine)
5. **Flash** the sketch. On serial monitor you should see IP and mDNS info.
6. Open **`http://funkwetter.local/`** (or the IP) to configure values.

Remote sensor node

1. Wire **BMP280 (I²C)** and **DHT22** as above.
2. Flash a minimal sketch that:
    - Reads BMP280 (P, T) and DHT22 (RH), and a configured site height `z_m`
    - Serves JSON at `/sensors.json`
      Example payload: