🔧 Step 1: Initial Configuration of LoRa Modules

In the first step, I use my laptop along with the following Python script to configure the LoRa modules through a TTL-to-USB converter:

```
#!/usr/bin/python3
import json
import serial
import os

# Configuration paths
JSON_PATH = "/home/amer/Work/Elebel/15jun25/lora_nodes.json"
INDEX_FILE = "/home/amer/Work/Elebel/15jun25/.device_index"

# Serial port settings
SERIAL_PORT = "/dev/ttyUSB0"  # Update this to match your LoRa serial
port
BAUD_RATE = 115200

# Load device configurations from JSON
def load_devices(json_path):
    with open(json_path, "r") as f:
        return json.load(f)

# Get the index of the next device to configure
def get_next_index(index_path, total):
    if not os.path.exists(index_path):
        return 0
    with open(index_path, "r") as f:
        idx = int(f.read().strip())
    return (idx + 1) % total

# Save the current device index for next time
def save_index(index_path, index):
    with open(index_path, "w") as f:
        f.write(str(index))

# Send configuration commands to the LoRa module
def send_command(serial_port, network_id, address):
    with serial.Serial(serial_port, BAUD_RATE, timeout=1) as ser:
        cmd = f"AT+NETWORK={network_id}\r\n"
        ser.write(cmd.encode())
        print(f"Sent: {cmd.strip()}")

        cmd = f"AT+ADDRESS={address}\r\n"
        ser.write(cmd.encode())
        print(f"Sent: {cmd.strip()}")

# Main logic
def main():
```

```
    devices = load_devices(JSON_PATH)
    index = get_next_index(INDEX_FILE, len(devices))
    device = devices[index]

    print(f"Using device ID: {device['device_id']}, Role:
{device['role']}")
    send_command(SERIAL_PORT, device["network_id"], device["address"])
    save_index(INDEX_FILE, index)

if __name__ == "__main__":
    main()
```

📝 Description

- This script configures one LoRa module at a time using a USB-TTL converter.
- Device-specific settings are loaded from a JSON file (`lora_nodes.json`).
- The current index of the device being configured is stored in `.device_index` to ensure the next run configures the next device in the list.
- The configuration commands (like setting the `NETWORK_ID` and `ADDRESS`) are saved to the module's EEPROM.
- Other temporary parameters (such as frequency, group password, and AT parameters) can be set directly from the IDE and are **not** stored in EEPROM.

Output example

Here's a simple example of the **output** you might see when running the script:

```
Using device ID: 2, Role: receiver
Sent: AT+NETWORK=1
Sent: AT+ADDRESS=2
```

PROF

This output means:

- The script selected device with `device_id = 2` and `role = receiver` from the JSON file.
- It sent two configuration commands to the LoRa module over serial:
    - Set the network ID to `1`.
    - Set the device address to `2`.

Next time you run the script, it will move to the next device in the list.

If you **run the script again**, assuming the JSON file contains at least three devices, the output would move to the next device in the list. For example:

```

```

```
Using device ID: 3, Role: transmitter
Sent: AT+NETWORK=1
Sent: AT+ADDRESS=3
```

This means the script:

- Picked the **next device** (`device_id = 3`, `role = transmitter`).
- Sent the same configuration commands (with updated address):
    - `AT+NETWORK=1`
    - `AT+ADDRESS=3`

The script will continue cycling through all devices listed in the JSON file on each run.