

README

SX Frequency Calibration Repair Tool

This tool is intended to repair XBee modules that have a miscalibrated frequency alignment. The following products are supported:

- XBee PRO SX 900 MHz (Regions: USA, Australia, Brazil)
- XBee SX 900 MHz (Regions: USA, Australia, Brazil, New Zealand)
- XBee SX 866 MHz

Setup

There are some dependencies that need to be installed in order for this tool to run properly. To prepare the python environment:

- Make sure python version 3.9 or newer is installed and present on the system path.
- Optional: Create a virtual environment
 - Run `python -m venv .venv`
 - Run `source .venv/bin/activate` on Linux, or `.venv/Scripts/activate.bat` on Windows. You will need to run this command in each new terminal to activate the environment.
- Run `python -m pip install -r requirements.txt`

Usage

Repairing with UART access

To repair a module that can be accessed via the UART, connect the module to the computer using a USB-to-Serial adapter or XBee Interface Board and run the following:

```
python xbee_repair.py repair-local --port COM1 --baud 115200
```

where:

- `--port` is the path to the COM port with the XBee (e.g. `COM1`)
- `--baud` is the baud rate configured on the XBee (e.g. `115200`)

Repairing without UART access

Modules that do not have serial access available can be repaired over the air. To do so, a module must be connected to the device running this tool to act as a 'server' for the updates.

Repairing modules over the air consists of two steps: Discovering nearby modules, and repairing them.

Searching for modules

To discover modules, run the following:

```
python xbee_repair.py search --port com1 --baud 115200 --output radios.txt --count 10
```

where:

- `--port` is the path to the COM port with the XBee.
- `--baud` is the baud rate configured on the XBee.
- `--output` is the path to a file to save the list of all discovered radios and their frequency offsets.
- `--count` is the number of radios expected to be found. The search will finish early if this many radios are found. This parameter is optional, but can cut a decent amount of time out of the search.

This will search for radios across a range of frequency offsets, and will save the results in the file specified by the `--output` parameter.

Repairing discovered modules

To repair modules once they've been discovered, run the following:

```
python xbee_repair.py repair --port com1 --baud 115200 --input radios.txt
```

where:

- `--port` is the path to the COM port with the XBee.
- `--baud` is the baud rate configured on the XBee. Repairing over the air will be significantly faster if a baud rate of at least 115200 is used.
- `--input` is the path to the file created in the search step.

Restoring the host module

The module that is used to perform the OTA updates will need to be restored to its original state after all updates are complete. This can be done with the `repair-local` command, as described in "Repairing with UART access" above.

Configuration

Certain parameters of this tool can be changed to support different radios. These are controlled by a configuration file that can be overridden when running the program.

A default configuration is stored in `xbee_repair_config.zip` alongside the source file and will be used if no configuration is specified.

If you have received a different configuration file, it can be used with all of the usage instructions above by specifying the `--config / -C` argument before the subcommand, for example:

```
python xbee_repair.py -C other_config.zip repair --port com1 --baud 115200 --input radios.txt
```

Version history

This is version 1.0

- v1.0: Initial Release