LoRa/MQTT/Telnet Terminal/Gateway

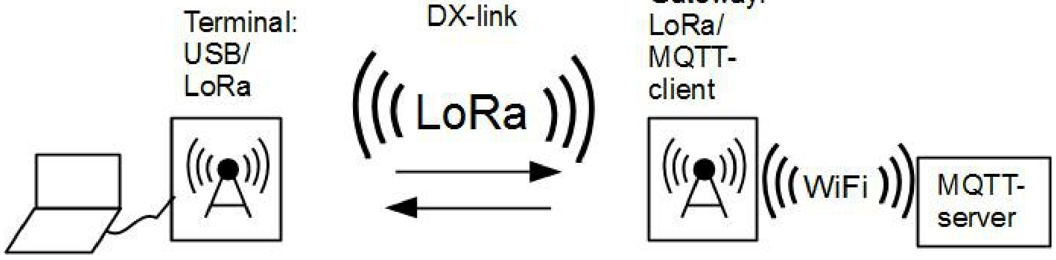
Manual for the Software of the LoRa/Serial/USB/WiFi Interface

From a distance

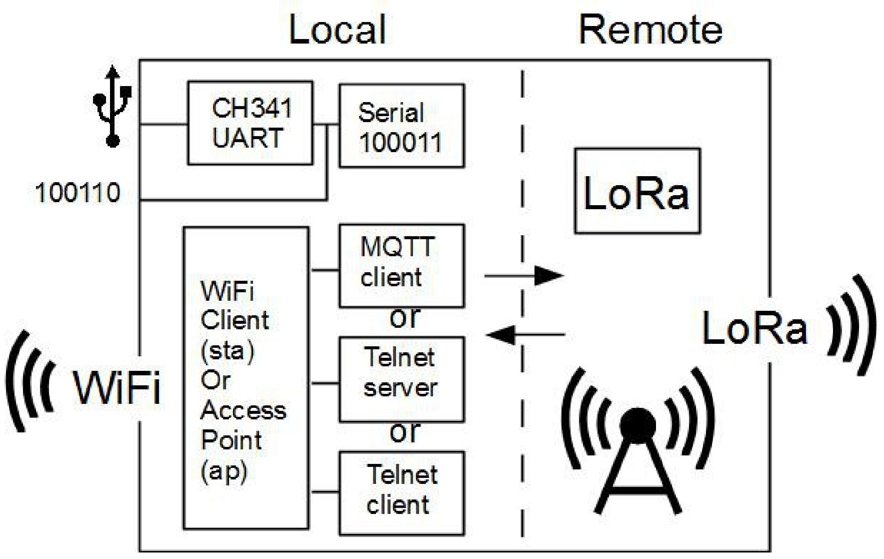
This software turns the above interface for converting messages between LoRa and MQTT, into a versatile and powerful, simultaneous gateway and terminal. With two identical devices, one can be used to generate LoRa messages from a terminal (PuTTY), the other one to receive them and publish onto the MQTT server in our smarthome. Or, two MQTT servers can be linked together over long distance to seemlessly extend smarthome range to remote sites (smart office/smart cities). Or, two devices can be used to exchange chat messages over long distances without using the cellular network, e.g. for emergeny communications (Emcom environment). Etc etc.

One unique device does it all

The usual setup for controlling a MQTT server over long distances using LoRa consists of a LoRa terminal and a LoRa gateway:



Here, the innovative approach is to use a unique and identical device on both sides of the long distance link. It integrates a series of protocols, grouped in two categories, “Local“ and “Remote“.



The philosophy is that a message entering on the local side goes out the remote side and vice versa. The serial interface is always enabled and one of the other local protocols has to be chosen by selecting the local mode, defaulting to MQTT. Therefore, the device can be used as is on both sides of the connection in the above scheme. On the former “terminal“ side, the MQTT or even the WiFi connection would just fail or time out, leaving the serial and LoRa interface operational. On the MQTT side, the WiFi and MQTT credentials have to be entered to establish communication with the server. Furthermore, the radio settings have to be identical on both sides. That's it.

Features:

- "Local" interfaces: Serial/USB /BT and Telnet server/Telnet client/MQTT client over WiFi
- LoRa Radio fully user configurable
- User configuration via terminal at runtime. settings stored on EEPROM
- Remote Configuration via LoRa possible (see section on Verbatim mode)
- Command acknowledgements can be enabled or disabled
- Power mamagement, auto-shutdown on low batt, and remote shutdown if enabled by hardware

First steps

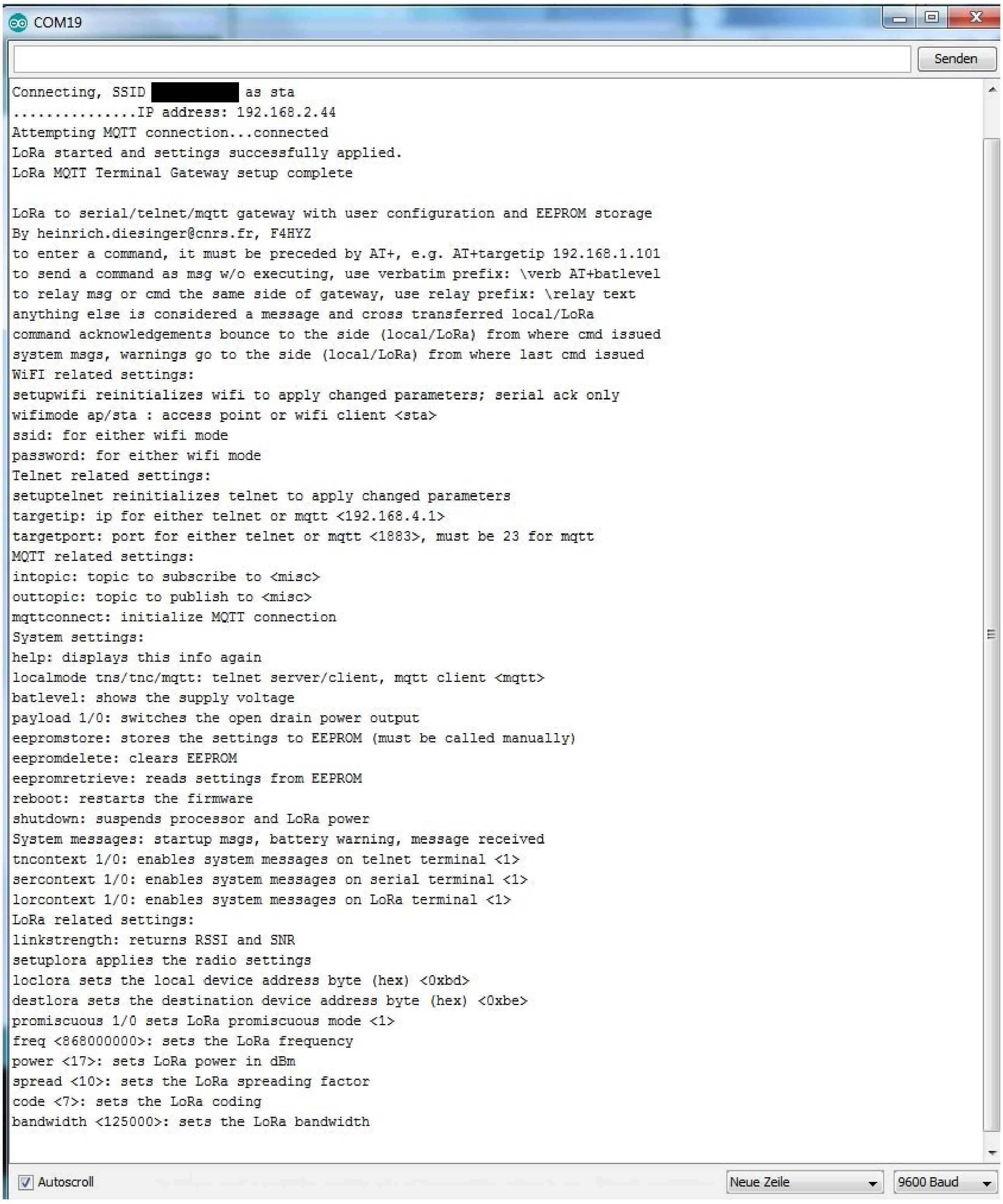
The software is compatible to the board presented earlier
http://heinerd.online.fr/elektronik/loraboard1.php3
or similar setups with identical connections between a WeMos D1 mini and a Ra-02 LoRa radio. It is envisageable to port it to other setups by editing the respective GPIO port assignments in the source file.

Flashing the firmware

To get the software onto the hardware, either it may be compiled locally in the Arduino IDE, after installing the necessary board and libraries. Alternatively, the binary can be flashed onto the board using the ESP8266 flasher.

Startup

At power on, some messages are immediately displayed if a terminal program is connected with the serial settings correctly set, default 9k6 baud 8N1. If missed, this step can be repeated by pressing the reset button on the D1 mini module, or just typing AT+help to repeat the help text.



```
Connecting, SSID [redacted] as sta
...............IP address: 192.168.2.44
Attempting MQTT connection...connected
LoRa started and settings successfully applied.
LoRa MQTT Terminal Gateway setup complete

LoRa to serial/telnet/mqtt gateway with user configuration and EEPROM storage
By heinrich.diesinger@cnrs.fr, F4HYZ
to enter a command, it must be preceded by AT+, e.g. AT+targetip 192.168.1.101
to send a command as msg w/o executing, use verbatim prefix: \verb AT+batlevel
to relay msg or cmd the same side of gateway, use relay prefix: \relay text
anything else is considered a message and cross transferred local/LoRa
command acknowledgements bounce to the side (local/LoRa) from where cmd issued
system msgs, warnings go to the side (local/LoRa) from where last cmd issued
WiFI related settings:
setupwifi reinitializes wifi to apply changed parameters; serial ack only
wifimode ap/sta : access point or wifi client <sta>
ssid: for either wifi mode
password: for either wifi mode
Telnet related settings:
setuptelnet reinitializes telnet to apply changed parameters
targetip: ip for either telnet or mqtt <192.168.4.1>
targetport: port for either telnet or mqtt <1883>, must be 23 for mqtt
MQTT related settings:
intopic: topic to subscribe to <misc>
outtopic: topic to publish to <misc>
mqttconnect: initialize MQTT connection
System settings:
help: displays this info again
localmode tns/tnc/mqtt: telnet server/client, mqtt client <mqtt>
batlevel: shows the supply voltage
payload 1/0: switches the open drain power output
eepromstore: stores the settings to EEPROM (must be called manually)
eepromdelete: clears EEPROM
eepromretrieve: reads settings from EEPROM
reboot: restarts the firmware
shutdown: suspends processor and LoRa power
System messages: startup msgs, battery warning, message received
tncontext 1/0: enables system messages on telnet terminal <1>
sercontext 1/0: enables system messages on serial terminal <1>
lorcontext 1/0: enables system messages on LoRa terminal <1>
LoRa related settings:
linkstrength: returns RSSI and SNR
setuplora applies the radio settings
loclora sets the local device address byte (hex) <0xbd>
destlora sets the destination device address byte (hex) <0xbe>
promiscuous 1/0 sets LoRa promiscuous mode <1>
freq <868000000>: sets the LoRa frequency
power <17>: sets LoRa power in dBm
spread <10>: sets the LoRa spreading factor
code <7>: sets the LoRa coding
bandwidth <125000>: sets the LoRa bandwidth
```

At the first startup, likely an error message shows up showing that the EEPROM settings cannot be retrieved because no such file exists. This is normal because it will be created only after setup using the command AT+eepromstore for the first time. Hence, the hardcoded factory defaults are used, and also the attempts of connecting WiFi and MQTT will fail and timeout.

Configuration using AT+ commands

We now start configuring as described by the help text. Commands are entered the oldfashioned way by the prefix AT+ as in a modem during the 90's (Hayes command set...) or still nowadays when configuring the bluetooth modules. This is to distinguish them from messages and causes them to be processed by our device rather than being sent over the ether. The syntax is AT+command [argument], and if the argument is absent, typically the present valued will be queried and displayed.

Entries

After successful configuration (don't forget to save by AT+eepromstore), it is possible to also send messages. For either entries it is normal that characters do not appear on the terminal windows as they are typed (echo off). To make an entry, our typing can be terminated by the new line command ('\n'), but a 2 second timeout is also implemented for devices where it is difficult to make a newline command. Usually, either this can be set in the terminal settings or there may be shortcuts such as Ctrl-J in PuTTY.

Verbatim mode - Sending a command to the remote device

As advertised , commands can also be entered remotely, our device also accepts them from the LoRa side. The \verb prefix prevents an AT+ command from being recognised as such and being executed on the local machine. Instead, the \verb prefix is stripped and the remainder of the entry is treated as a message, crossing the local/remote barrier and going over the ether. The purpose is to be able to send a command to the remote machine and have it being exected there.
Example: \verb AT+outtopic lightingcontrol

Relay mode

Inversely, the \relay prefix prevents an entry that would otherwise be treated as a message from crossing the local/remote barrier. This serves two purposes: either, the user can send something from the serial terminal to the mqtt interface, although both are local interfaces; or, an entry that arrives via LoRa is re-transmitted from the LoRa interface, enabling typical radio repeater operation. The \relay prefix is also stripped off before retransmission. It is possible to enter \relay several times for an increased number of hops.

IP addresses

If the WiFi interface is used as a access point (ap), the IP adress is always 192.168.4.1. In station (sta) mode, its IP adress is most likely be attributed by DHCP. The choosen IP is the destination address if the interface is set to telnet cient (tnc) or mqtt client (mqtt). If set to telnet server (tns), the IP adress is 192.168.4.1 for ap mode or the DHCP attributed one in station mode. It shows up on startup, or you can find it in your WiFi router.

Technical background

The processing of entries and the routing of messages is handled by a number of flags. When a LoRa entry arrives, the flag fromlora that contains the message direction is set, but reset in case of a local entry. Then, it is checked for a prefix and for \relay the direction is changed (bounce function). AT+ commands are withheld and fed to the command parser (unless preceded by the \verb prefix). The command parser stores the direction of the command in the flag lastcmdlocal, because when a system message is generated, the device has to remember from where the last AT+ command arrived to determine where the human user is sitting and to where the system message has to be sent. Also, when calling the command parser, the flag cmdflag is set, which informs the output routine that the following message is a command acknowledgement. Furthermore, for each interface, a boolean xxxcontext can be set in the configuration, that determines whether the respective interface will receive system messages. One exception is the serial interface, that will always inform of the arrival of a MQTT message. Because, if on the local side, a machine is connected to the MQTT interface, then there will most likely not be another machine listening on the serial port, and hence the system messages interesting for the human user watching the terminal (or not!), will not cause unexpected behavior. Also note that there is no boolean mqttcontext because the MQTT interface will never receive system messages from our device. Also note that the system messages (message transferred, message relayed, battery critically low...) can be turned off by the xxxcontext variables, but not so for the command acknowledgements. But this cannot perturb a machine, because only the human user is expected/supposed to send a AT+ command to our device. In summary, the user shall just make sure that xxxcontext boolean is disabled for interfaces where a machine is listening, which can not only be the MQTT interface but also be the telnet.

Miscellaneous

Warning: all messages are sent unencrypted, and anybody using the same radio settings and within LoRa range will receive them.

This device uses LoRa in a “raw“ mode by the driver of Sandeep Mistry: https://github.com/sandeepmistry/arduino-LoRa. It is not LoRaWAN nor Thethingsnetwork. These implement kind of a TCP/IP over LoRa on top of the same library, with encryption and receipt acknowledgement. To use this, you need a LoRaWAN gateway within range or install your own. Chance are that if you are not staying in Santa Barbara, your area is not covered. While the use of the network is free of charge, the installation of a gateway can be costly.

Here we just send plain and unsecured messages. Errors can occur but are more likely to be rejected /ignored by your smarthome, rather than freezing your home by the aircon.

Check this link for a comparison to LoraWAN and a gateway that uses LoRa in the same basic way as we do: https://appcodelabs.com/introduction-to-iot-building-a-budget-single-channel-lora-gateway

Maybe the ESP32 is now mature and the ESP32 core of the Arduino IDE stable, so it might be worth a portation, instead of homebrewing the hardware for marginal savings. The TTGO, Lillygo or Heltec ESP32 Lora boards and the like cost slightly more than our parts but have in addition an OLED display. At the beginning of development of this project they were not available.

Disclaimer: Radio transmitting device – comply with the frequency allocation of your country. Use at your own risk, no warranty of any kind. Not for use in medical/ life sustaining applications, passenger transport or anything else required to be failsafe.

Author: heinrich.diesinger@cnrs.fr, F4HYZ, http://heinerd.online.fr