iC880A-SPI QuickStart Guide

How to get started with the iC880A-SPI

Document ID: 4100/40140/0096

Category: confidential

IMST GmbH
Carl-Friedrich-Gauss-Str. 2-4
D-47475 Kamp-Lintfort

Document Information

File name	iC880A-SPI_QuickStartGuide.docx
Created	2015-03-15
Total pages	17

Revision History

Version	Description
0.1	Preliminary version.

Aim of this Document

Aim of this document is to give some quick start instructions how to start working with the WiMOD iC880A-SPI.

Confidentiality Note

This document has to be treated confidentially. Its content must not be published, duplicated or passed to third parties without our express permission.

Table of Contents

1 Overview

Purpose of this documentation is to give some useful information how to get started with the iC880A-SPI and the open source LoRa HAL published on the github platform.

The concentrator module iC880A-SPI is meant to be connected to a host system forming an integrated gateway.

1.1 Hardware PCB

Figure 1-1 depicts the printed circuit board of the iC880A-SPI. As main interface there are the pin headers for connecting the board to a host system that runs the driver software. A SMA connector or an uFL connector is supplied in order to connect any suitable antenna to this board. The board offers some LEDs to give a visual feedback of the current operation status of the board. The concrete meaning of the LEDs is software dependant. As default in Version 1.7 the setting of the LEDs is:

1) Backhaul packet
2) TX packet
3) RX Sensor packet
4) RX FSK packet
5) RX buffer not empty
6) Power

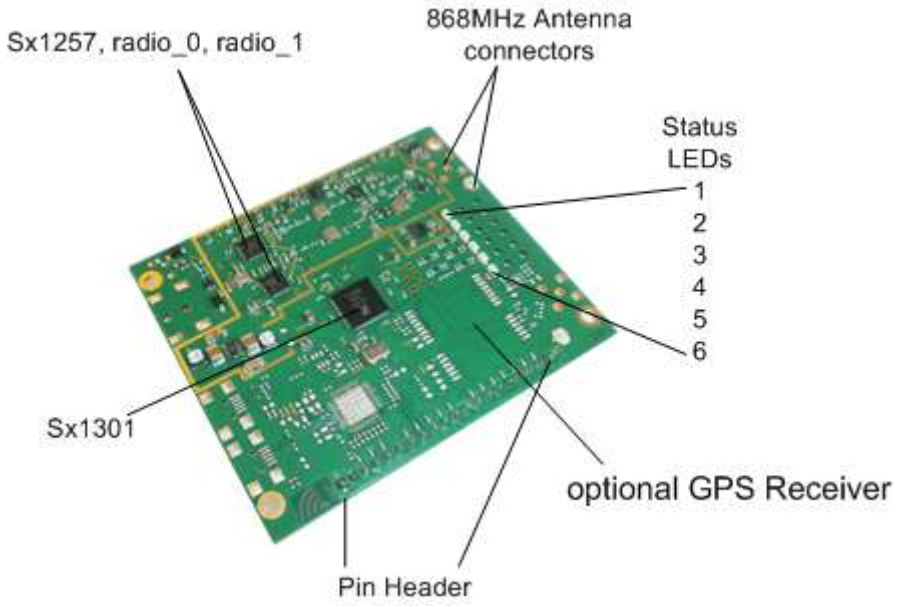

Figure 1-1Picture of iC880A-SPI

1.2 Basic System Concept

Figure 1-2 shows the basic system concept for the LoRA WAN system. The iC880A-SPI is the central hardware solution for all LoRa based radio communication. It receives and transmits radio messages. Processing of the radio messages as well as the protocol related tasks is done by (external) host system(s). The concrete segmentation of the protocol related tasks is outside the scope of this document.

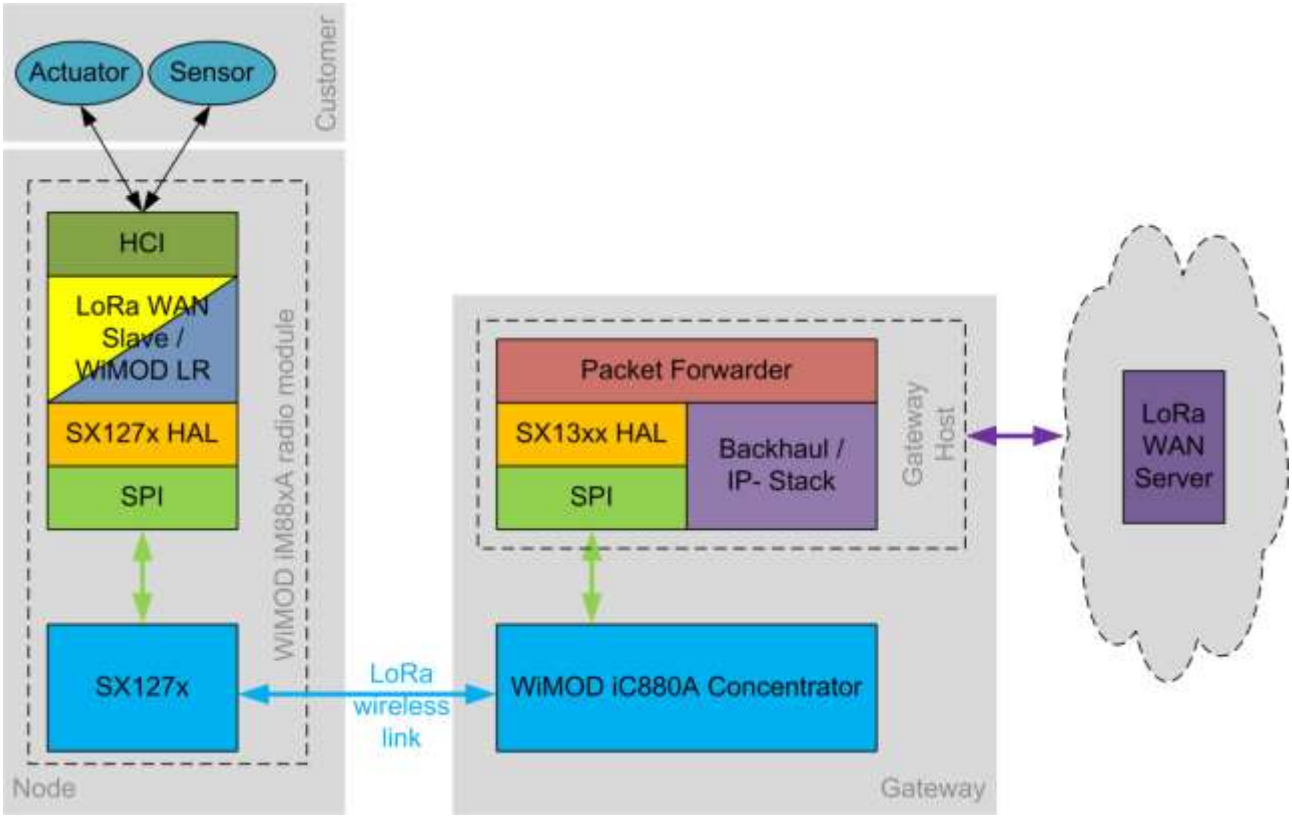

Figure 1-2: Basic System Concept

1.3 Linux Host System

As a host system an (embedded) Linux host system is planned to be used. This quick start guide assumes that the user has got an (embedded) Linux host system (e.g. Raspberry PI, or similar), is familiar with the basic concept of a Linux console and knows how to compile source code.

All commands shown in this document are tested with a Debian 8 "Jessie" Linux distribution on a Raspberry PI B+.

```
pi@raspberrypi ~ $ lsb_release -a
No LSB modules are available.
Distributor ID:     Raspbian
Description:   Raspbian GNU/Linux 8.0 (jessie)
Release:  8.0
```

```
Codename: Jessie
pi@raspberrypi ~ $ uname -a
Linux raspberrypi 3.18.7+ #755 PREEMPT Thu Feb 12 17:14:31 GMT
2015 armv6l GNU/Linux
```

It is possible to use any other host system and Linux distribution, but there may be some variations in the commands, that are not within the scope of this document.

Especially the Linux kernels that use the feature called "device-tree" seem to have a timing related problem with the SPI CE lines. This may cause communication disorders / problems on the SPI bus. (For the Raspberry Pi the kernel versions 4.x and newer seem to have that problem.) Fixing these kernel related SPI problems is however outside the scope of this document.

The basic setup of the (embedded) Linux host system (keyboard layout, time zone, network connectivity, accessibility, …) is outside of the scope of this document, too.

1.4 Hardware Connection

The iC880A-SPI provides SPI – pin headers as main interface to the Linux host system. Before connecting the iC880A-SPI and the host, please read this document carefully. In order to establish a logical connection the host needs to have some drivers installed.

1.4.1 Pin Headers

In order to establish a SPI connection between the iC880A-SPI and the host system, the following pins have to be used as minimum wiring set:

PIN	PIN Name	PIN Type	Description
21	VDD	Power	+5 V Supply Voltage (> 700mA)
22	GND	Power	
14	CLK	Input	Sx1301 SPI-Clock
15	MISO	Output	Sx1301 SPI-MISO
16	MOSI	Input	Sx1301 SPI-MOSI
17	NSS	Input	Sx1301 SPI-NSS
13	Reset	Reset	Sx1301 Reset, for a stable start-up Reset should be at high-level for 100 ns (min), once the supply voltage is stable
12	GND	Power	

Table 1-1: iC880A-SPI Minimum Pinout Table

The power pins (21, 22) have to be connected to a power source that is able to provide more

than 700 mA. Therefore it is recommended to choose a proper power supply that is able to supply the iC880A-SPI and the host system (e.g. Raspberry PI) at the same time. (See Figure 1-3)

WiringPi	Raspberry Pi (B / B+)	Pin	Pin	Raspberry Pi (B / B+)	WiringPi
-	+ 3,3 V	1	2	+ 5 V	-
8	(SDA1) GPIO 2	3	4	+ 5 V	-
9	(SCL1) GPIO 3	5	6	GND	-
7	(GPIO_GCLK) GPIO 4	7	8	GPIO 14 (TXD0)	15
-	GND	9	10	GPIO 15 (RXD0)	16
0	(GPIO_GEN0) GPIO 17	11	12	GPIO 18 (GPIO_GEN1)	1
2	(GPIO_GEN2) GPIO 27	13	14	GND	-
3	(GPIO_GEN3) GPIO 22	15	16	GPIO 23 (GPIO_GEN4)	4
-	+ 3,3 V	17	18	GPIO 24 (GPIO_GEN5)	5
12	(SPI_MOSI) GPIO 10	19	20	GND	-
13	(SPI_MISO) GPIO 9	21	22	GPIO 25 (GPIO_GEN6)	6
14	(SPI_SLCK) GPIO 11	23	24	GPIO 8 (SPI_CE0_N)	10
-	GND	25	26	GPIO 7 (SPI_CE1_N)	11
	special I2C	27	28	special I2C	
	GPIO 5	29	30	GND	
	GPIO 6	31	32	GPIO 12	
	GPIO 13	33	34	GND	
	GPIO 19	35	36	GPIO 16	
	GPIO 26	37	38	GPIO 20	
	GND	39	40	GPIO 21	

Table 1-2: Raspberry PI GPIO Pin Table

The SPI pins (14 to 17) have to be connected to the according pins of the host system. As an example Table 1-2 illustrates the GPIO ports of a Raspberry Pi. The SPI functionality is given by the pins 19, 21, 23 and 24.

Additionally the Reset pin of the iC880A-SPI has to be connected to the host system. On the host side virtually any normal GPIO pin can be used for that. (In the following pin 11 aka. GPIO17 is considered to be connected to the reset pin.)

Figure 1-3 depicts the necessary connections between the host and the iC880A-SPI.

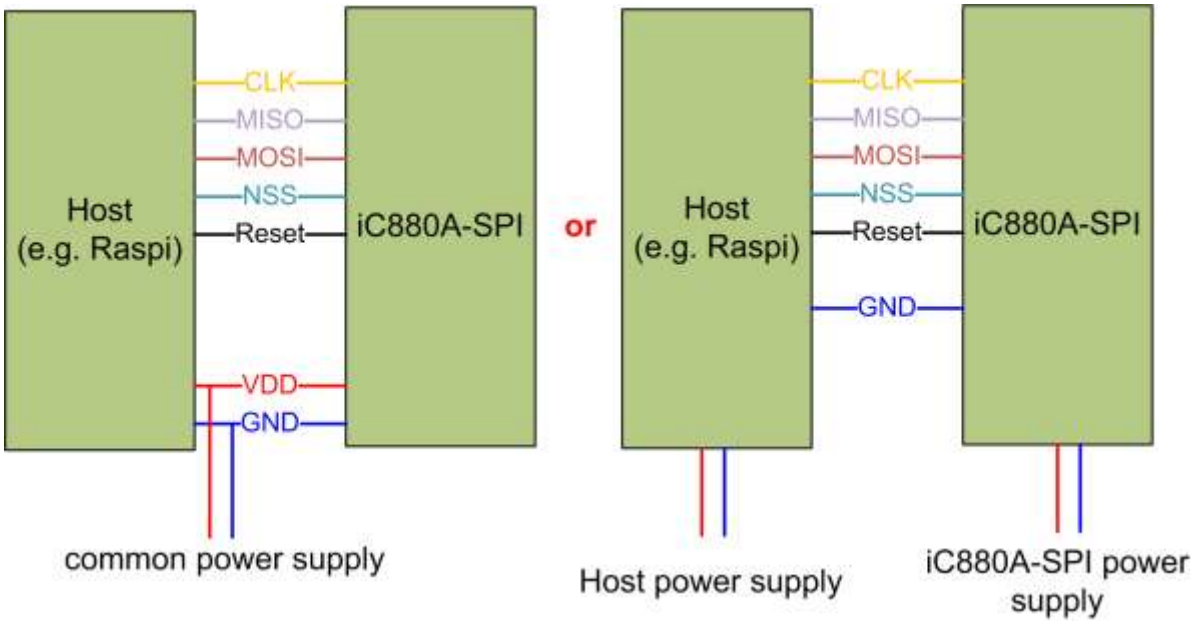

Figure 1-3: Connection variants between Host and iC880A-SPI

It is recommended to connect the necessary signals using wires that are as short possible in order to prevent communication disorders.

2 Open Source Driver on Github

Semtech provides an open source driver for SX1301 based LoRa solutions. The complete repository is hosted on the github platform (https://github.com/).

The main vanilla source repository for the LoRa related projects maintained by Semtech can be found at:

https://github.com/Lora-net

There you will find an implementation for the HAL (Hardware Abstraction Layer) of SX1301based designs. This implementation is called "lora_gateway" (https://github.com/Lora-net/lora_gateway). **This document references to version 3.2 of the lora_gateway repository. Newer or other versions of this software may behave differently or may require a different**

configuration. Changes / updates within all relevant software repositories are subject to change without notice.

2.1 Download of the Open Source Driver

Install a git client on your (embedded) host system: On Debian based systems the git client can be installed by calling the following command in a terminal emulator / console:

```
sudo apt-get install git
```

Create local folder that should contain the local copy of the repository:

```
mkdir -p ~/LoRa/lora_gateway
```

Get a copy of the repository:

```
cd ~/LoRa/lora_gateway
```

```
git clone https://github.com/Lora-net/lora_gateway.git
```

2.2 Preparing the host system (Raspberry Pi)

2.2.1 Preparing the SPI Interface

The github open source driver (in version 3.2 and newer) requires a "native" SPI interface of the host system to be present. This is by default called "/dev/spidev0.0". In order to use this interface, it must be enabled on the Raspberry Pi platform. To enable it the user must call the following command in a terminal emulator / console of the Raspberry Pi running the Debian based Rasbian distribution:

```
sudo raspi-config
```

A text based menu will appear and the user has to enter the "Advanced Options" sub menu. Next the entry "SPI" has to be chosen and the following two questions have to be answered with the "Yes" option. After a reboot of the system the SPI interface is ready to be used.

2.2.2 Preparing the Reset Pin Control

Now that the SPI interface is accessible, the control logic for the reset pin has to be set up. During the development history of the LoRa driver, the support for GPIO pin control has been removed from the sources. (Note: the repository is not maintained by IMST.) That means the user has to take care of it via an additional piece of software. In general there are two possibilities to control the GPIO pin that is connected to the Reset pin of the iC880A-SPI.

2.2.2.1.1 Possibility A: Control the GPIO pin via a shell script

A small shell script can be written to reset the iC880A-SPI before the LoRa driver can access the hardware. The content of the shell script can look like the following example (which assumes that the GPIO 17 (pin 11) of the Raspberry Pi is connected to the reset pin of the iC880A-SPI):

```
#!/bin/bash
echo "17"  > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio17/direction
echo "1"   > /sys/class/gpio/gpio17/value
sleep 5
echo "0"   > /sys/class/gpio/gpio17/value
sleep 1
echo "0"   > /sys/class/gpio/gpio17/value
```

These lines can be stored in a file called "iC880-SPI_reset.sh". The user must call this script once after every boot up in order to get the concentrator IC in a clean state.

2.2.2.1.2 Possibility B: Control the GPIO pin via a compiled C-Tool / library

If the host system is a Raspberry Pi the user can write a small C-Tool to reset set the iC880A-SPI. In order to access the GPIO pins of the Raspberry Pi there is a library called "wiringPi" that takes care of the low level details. The library can be downloaded from http://wiringpi.com. Please refer to this site to get installation and usage instructions. The content of the iC880-SPI_reset.c file can look like the following:

```
#include <wiringPi.h>
#include <unistd.h>

#define GPIO_RESET_PIN       0 // see wiringPi mapping!

int main() {
    wiringPiSetup();
    pinMode(GPIO_RESET_PIN, OUTPUT);
    digitalWrite(GPIO_RESET_PIN, HIGH);
    sleep(5);
    digitalWrite(GPIO_RESET_PIN, LOW);

    return;
}
```

The user must call this tool once after every boot up in order to get the concentrator IC in a clean state.

(As an alternative to an external program, the reset routine shown can also be copied into the "lgw_spi_open()" function that is located in the file called "loragw_spi.native.c".)

2.3 Configuration of the Driver

The driver can be configured in order to change the verbosity level of the internal processes within the library.

The filef ile lora_gateway/libloragw/library.cfg is used as general configuration file for the whole `libloragw` system library. In order to fit the needs of the iC880A-SPI, some parameters can be set to the following values prior (re-) compiling the library:

```
DEBUG_AUX= 0
DEBUG_SPI= 0
DEBUG_REG= 0
DEBUG_HAL= 1
DEBUG_GPS= 0
```

These flags are used by the internal modules of the library to enable / disable additional output for debugging purposes.

2.4 Compilation of the Library

In order to compile the `libloragw` library it is assumed that a gcc/g++ compiler and a make utility is already installed and runable.

Enter the lora_gateway folder and execute the make command:

```
cd lora_gateway

make
```

This will compile the library and some of the basic test utilities located in the subfolders of `lora_gateway`.

After a successful compilation a library file called "`libloragw.a`" has been created and is ready for usage now.

2.5 Example Utilities

In the folder `lora_gateway` there are several example utilities showing the usage of the `libloragw` library. This quick start guide will show the "`util_pkt_logger`" and the "`util_tx_test`".

2.5.1 util_tx_test

This is a small utility demonstrating the ability to send out some data via the iC880A-SPI concentrator device. To start this utility go into the folder and start it:

```
cd lora_gateway/util_tx_test

./util_tx_test -r 1257 -f 866.5
```

The program should start and print out some output like that:

```
Sending -1 packets on 866500000 Hz (BW 125 kHz, SF 10, CR 1,
16 bytes payload, 8 symbols preamble) at 14 dBm, with 1000 ms
between each

INFO: concentrator started, packet can be sent

Sending packet number 1 ...OK

Sending packet number 2 ...OK

Sending packet number 3 ...OK
```

Each time a new packet is being transmitted the TX-LED of the iC880A-SPI should flash. The tool can be stopped by pressing the key combination `Ctrl+C`. For more information about this tool, please consult the online help by calling:

```
./util_tx_test --help
```

2.5.2 util_pkt_logger

This utility tool is able to demonstrate the receiving ability of the iC880A-SPI device. It configures the iC880A-SPI device for RX operation and writes each received packet into a CSV file. That file can be opened and inspected afterwards.

This tool needs a configuration file that holds parameters relevant to the RX operation of the concentrator. Please take a look into this file called `global_conf.json` .

An example global_conf.json looks like that:

```
{
    "SX1301_conf": {
        "lorawan_public": true,
        "clksrc": 1, /* radio_1 provides clock to concentrator */
        "radio_0": {
            "enable": true,
            "type": "SX1257",
            "freq": 867500000,
            "rssi_offset": -166.0,
            "tx_enable": true
        },
        "radio_1": {
            "enable": true,
```

```
                "type": "SX1257",
                "freq": 868500000,
                "rssi_offset": -166.0,
                "tx_enable": false
            },
            "chan_multiSF_0": {
                /* Lora MAC channel, 125kHz, all SF, 868.1 MHz */
                "enable": true,
                "radio": 1,
                "if": -400000
            },
            "chan_multiSF_1": {
                /* Lora MAC channel, 125kHz, all SF, 868.3 MHz */
                "enable": true,
                "radio": 1,
                "if": -200000
            },
…
            "tx_lut_0": {
                /* TX gain table, index 0 */
                "pa_gain": 0,
                "mix_gain": 8,
                "rf_power": -6,
                "dig_gain": 0
            },
…
            },
            "tx_lut_15": {
                /* TX gain table, index 15 */
                "pa_gain": 3,
                "mix_gain": 14,
                "rf_power": 27,
                "dig_gain": 0
            }
        },
        "gateway_conf": {
            "gateway_ID": "AA555A0000000000"
        }
    }
```

In the head of the file there are some general settings concerning the private/public LoRa network setup. The setting for the internal clock must not be changed. Next there is one config block for each radio of the concentrator. Each radio can be turned on / off and operates on a certain base frequency. (See block "`radio_0`" and "`radio_1`"). The settings called "type" must be kept unchanged (SX1257).

The following blocks define the parameters used by the virtual multi SF channels. Each logical channel configuration consists of the radio to be used and the RF frequency. The RF frequency is defined as difference to the base frequency defined in the corresponding radio config block.

For further details please check the documentation.

After checking the configuration the tool can be started via:

```
cd lora_gateway/util_pkt_logger

./util_ptk_logger
```

After starting the packet logger, each LoRa packet that is being received by the concentrator will be record in a CSV file.

If the user wants to send out some demo data with another radio module some care must be taken about the set up type of network. In case the user uses the Windows Software called "WiMOD LR Studio"[1], the setting called "lorawan_public" has to be set to false in order to receive the datagrams of the transmitting WiMOD iM88xA radio module or the iU88xA USB-Stick.

If the setting is "lorawan_public: true" than the peer has to send packets that are compliant to public LoRaWAN networks. This can be archived by using the Software called "WiMOD LoRaWAN EndNode Studio" in combination with a WiMOD module running a LoRaWAN firmware.

The following screen shot shows an example of the TX and RX data send by the Software called "WiMOD LR Studio":

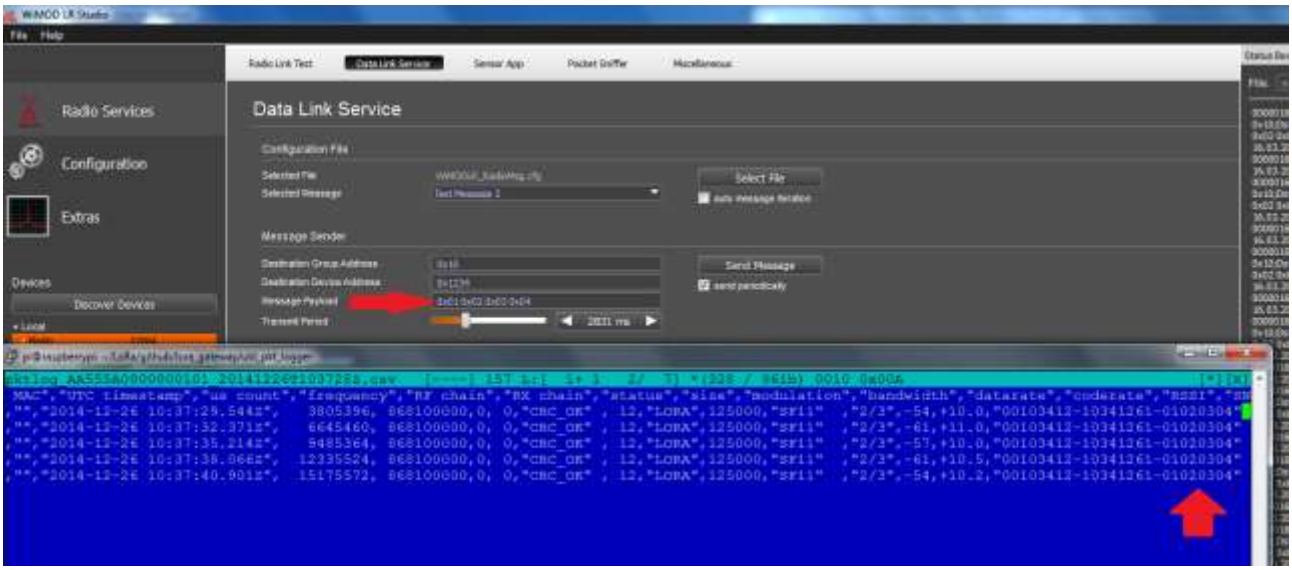

Figure 2-1: Screen shot of the recorded CSV file and the WiMOD LR studio

Note: The CSV file records all data as hexadecimal bytes with no respect to any fields defined in the LoRa WAN specification. Therefore you have to look at the end of the payload within the CSV file to see the payload data message itself.

[1] The WiMOD LR Studio is intended to be used with a WiMOD Module running the WiMOD LR-Base firmware. The other PC software is intended to be used in conjunction with a WiMOD Module running the WiMOD LoRaWAN firmware.

3 Next Steps

After stepping through this guide the user is able to start developing his/her own iC880A-SPI based solution.

As a final hint there are some more interesting open source projects within the gitbub repository

https://github.com/Lora-net/

For example there is a project called "packet_forwarder" that demonstrates how to receive LoRa packets and forward the data to a remote server which may run a LoRa WAN Server implementation or any other proprietary solution.

For more information please refer to the User Guides and specific data sheets of the corresponding products.

4 Regulatory Compliance Information

The use of radio frequencies is limited by national regulations. The radio module has been designed to comply with the European Union's R&TTE (Radio & Telecommunications Terminal Equipment) directive 1999/5/EC and can be used free of charge within the European Union. Nevertheless, restrictions in terms of maximum allowed RF power or duty cycle may apply.

The radio module has been designed to be embedded into other products (referred as "final products"). According to the R&TTE directive, the declaration of compliance with essential requirements of the R&TTE directive is within the responsibility of the manufacturer of the final product. A declaration of conformity for the radio module is available from IMST GmbH on request.

The applicable regulation requirements are subject to change. IMST GmbH does not take any responsibility for the correctness and accuracy of the aforementioned information. National laws and regulations, as well as their interpretation can vary with the country. In case of uncertainty, it is recommended to contact either IMST's accredited Test Center or to consult the local authorities of the relevant countries.

This Development Kit/Starter Kit does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and therefore may not meet the technical requirements of these directives or other related directives.

5 Important Notice

5.1 Disclaimer

IMST GmbH points out that all information in this document is given on an "as is" basis. No guarantee, neither explicit nor implicit is given for the correctness at the time of publication. IMST GmbH reserves all rights to make corrections, modifications, enhancements, and other changes to its products and services at any time and to discontinue any product or service without prior notice. It is recommended for customers to refer to the latest relevant information before placing orders and to verify that such information is current and complete. All products are sold and delivered subject to "General Terms and Conditions" of IMST GmbH, supplied at the time of order acknowledgment.

IMST GmbH assumes no liability for the use of its products and does not grant any licenses for its patent rights or for any other of its intellectual property rights or third-party rights. It is the customer's duty to bear responsibility for compliance of systems or units in which products from IMST GmbH are integrated with applicable legal regulations. Customers should provide adequate design and operating safeguards to minimize the risks associated with customer products and applications. The products are not approved for use in life supporting systems or other systems whose malfunction could result in personal injury to the user. Customers using the products within such applications do so at their own risk.

Any reproduction of information in datasheets of IMST GmbH is permissible only if reproduction is without alteration and is accompanied by all given associated warranties, conditions, limitations, and notices. Any resale of IMST GmbH products or services with statements different from or beyond the parameters stated by IMST GmbH for that product/solution or service is not allowed and voids all express and any implied warranties. The limitations on liability in favor of IMST GmbH shall also affect its employees, executive personnel and bodies in the same way. IMST GmbH is not responsible or liable for any such wrong statements.

5.2 Contact Information

IMST GmbH

Carl-Friedrich-Gauss-Str. 2-4
47475 Kamp-Lintfort
Germany

T +49 2842 981 0
F +49 2842 981 299
E wimod@imst.de
I www.wireless-solutions.de