Hardware: Raspberry Pi 2, IMST IC880A-USB, LoPy

I. Setting up the gateway
   1. Make sure that the Raspberry Pi is connected to the internet and that you connected the IMST iC880A over USB.
   2. Download of the Open Source Driver
      a. Install git client:

         sudo apt-get install git

      b. Create local folder:

         mkdir -p ~/LoRa/lora_gateway

      c. Clone the repository:

         cd ~/LoRa/lora_gateway

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

         Note: Use lora_gateway 2.0.0, not the latest version. (Run git checkout v2.0.0 in the lora_gateway folder).

         cd lora_gateway

         git checkout v2.0.0

   3. Downloading Additional Drivers
      a. Create a folder for the libmpsse files:

         mkdir –p ~/LoRa/libmpsse

      b. Download the newest version of the libmpsse files:

         cd ~/LoRa/libmpsse

         wget https://storage.googleapis.com/google-code-archivedownloads/v2/code.google.com/libmpsse/libmpsse-1.3.tar.gz

         tar –xzvf libmpsse-1.3.tar.gz

      c. Follow the installation instructions from the hosted website: (https://code.google.com/p/libmpsse/wiki/Installation)

         sudo apt-get install libftdi-dev
         cd ~/LoRa/libmpsse/libmpsse-1.3/src
         ./configure --disable-python && make && sudo make install

```
sudo ldconfig
```

Note: If the configure step fails, follow the instructions and try to install the missing dependencies.

4. Configuration of the Driver

The driver must be configured in order to work with the iC880A. The following chapter describes the necessary configuration options to make.

In the folder “LoRa/github/lora_gateway/libloragw” there is a file called “libraray.cfg”. This file contains the most important options for the HAL-library and must be modified to your needs before compiling the software.

a. File: lora_gateway/libloragw/library.cfg
   This file is used as general configuration file for the whole libloragw system library. In order to fit the needs of the iC880A, some parameters must be set to the following values prior (re-) compiling the library:

```
CFG_SPI= ftdi
CFG_CHIP= sx1301
CFG_RADIO= sx1257
CFG_BAND= eu868
```

   The parameter called “CFG_BRD” must be left empty or set to:

```
CFG_BRD= ref_1301_868
```

   Additionally, there are some optional debug settings that can be turned on (1) or off (0):

```
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.

b. File: lora_gateway/libloragw/src/loragw_spi.ftdi.c

   As second step a small modification must be done in one of the low-level connection files. The WiMOD iC880A LoRa Concentrator board uses a FTDI USB to SPI converter chip that is not supported in Semtech’s reference code. The only difference is another USB PID. Therefore, the USB PID settings in the libloragw library have to be adjusted prior compiling the library.

Open the file lora_gateway/libloragw/src/loragw_spi.ftdi.c with your favorite editor (e.g. nano)

```
nano lora_gateway/libloragw/src/loragw_spi.ftdi.c
```

Find the block with the "PRIVATE CONSTANTS" (around line 50 in the file) and change the line

```
#define PID 0x6010
```

To

```
#define PID 0x6014
```

Afterwards save the file and close the editor.

5. 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 runnable.

   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.

6. Setting up "udev" Rules

   The FTDI USB chip needs to be run in the SPI mode. Therefore an "udev" rule must be installed on the Linux host system. This rule will configure the right mode automatically when the iC880A device is connected to the host system. In order to install the udev rule the following steps must be done:

   In the folder lora_gateway/libloragw there is a template file present. Open the file in
   your favorite editor (e.g. nano).

   ```
   nano lora_gateway/libloragw/99-libftdi.rules
   ```

locate the line

```
SUBSYSTEM=="usb",ENV{DEVTYPE}=="usb_device",
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6010",
MODE="0664", GROUP="plugdev"
```

and change the entry

```
ATTRS{idProduct}=="6010”
```

to:

```
ATTRS{idProduct}=="6014”
```

Save the file and leave the editor. Next copy the file to the folder

```
/etc/udev/rules.d/
```

```
sudo cp 99-libftdi.rules /etc/udev/rules.d/
```

7. After following these steps:

   Restart the Pi.

   Run:

   ```
   ~/LoRa/lora_gateway/lora_gateway/util_pkt_logger/util_pkt_logger
   ```

   You see ‘INFO: concentrator started, packet can now be received’, which indicates that everything is functioning.

II. Setting up LoRa network on LORIOT

a. Installing the LORIOT software
   i. Sign up for an account.
   ii. You’re redirected to the dashboard page.
   iii. Click the link to register a new gateway.
   iv. You’re taken through a wizard. Choose the gateway you have, and follow the steps.
   v. You’re taken to the gateway page where you’ll find the LORIOT binary for your platform and a link to set up document.

      Tip: Use a tool like scp to copy the binary from your computer to the gateway. For example:

      ```
      scp ~/Downloads/loriot_pi_2_iC880A_USB_1.0.1.tar
      pi@192.168.2.7:~/
      ```

vi. Copy the libmpsse.so library from the installation package to the usr/lib directory.
vii. Run the loriot_pi_[B/2]_[CONCENTRATOR_TYPE]_[VERSION] binary
viii. Upon a successful startup, the process will daemonize to background. The last printed message should be INFO: Concentrator started, daemonizing
ix. The gateway shows as connected on the LORIOT gateway page. You're ready to work on the device.

III. Working with device

a. Create an application in LORIOT
b. Generate a device on LORIOT
    i. Go to your application page
    ii. Click on Devices
    iii. Click generate new device
    iv. Click on newly generated device
    v. Copy DevAddr, NwkSKey, AppSKey
    vi. Replace dev_addr, nwk_swkey, app_swkey in below example code and run the program

```
from network import LoRa
import socket
import binascii
import struct

# Initialize LoRa in LORAWAN mode.
lora = LoRa(mode=LoRa.LORAWAN)
```

vii. You should see message send from device on the device page.

c. Connect to LORIOT websocket for output data:
    i. In LORIOT: go to your dashboard, and click Applications > Sample App > Output.
    ii. Change the output type to WebSocket.
    iii. Copy the URL and the token under Current output setup, and paste them in the code sample below:

```
<!DOCTYPE html>
<html>
<head>
   <title></title>
</head>
<body>
  <p id="status">Connecting...</p>
  <p id="message"> </p>
   <script>
   var token = 'YOUR_AUTHENTICATION_TOKEN_HERE';
   var url = 'YOUR_TARGET_URL_HERE (incl {token} part)';
   var ws = new WebSocket(url.replace('{token}', token));
   ws.onopen = function() {
      document.querySelector('#status').textContent = 'Connected';
   };
   ws.onclose = function() {
      document.querySelector('#status').textContent = 'Disconnected';
   };
   ws.onmessage = function(e) {
      console.log('onmessage', e);
      var data = JSON.parse(e.data);
      document.querySelector('#message').textContent = data.data;
   };
   </script>
</body>
</html>
```

IV. References

Building your own private LoRa network

https://docs.mbed.com/docs/lora-with-mbed/en/latest/intro-to-lora/

iC880A Quick Start Guide

https://wireless-solutions.de/images/stories/downloads/Radio%20Modules/iC880A/iC880A_QuickStartGuide.pdf

LORIOT gateway installation guide

https://eu1.loriot.io/home/documentation.html#docu/gwguide/pi

LoRaWAN with ABP join method (LoPy)

https://docs.pycom.io/pycom_esp32/pycom_esp32/tutorial/includes/lora-abp.html