Development Instruction

DSI-0134 GATEWAY

CAO YANLIANG
## Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>Sect.</th>
<th>Update Description</th>
<th>By</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Approvals

<table>
<thead>
<tr>
<th>Organization</th>
<th>Name</th>
<th>Title</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Introduction.................................................................................................................................3
  1.1 Purpose & Description...............................................................................................................3
  1.2 Product Feature Summary.........................................................................................................3
  1.3 System block diagram...............................................................................................................3
  1.4 Development Environment and Tool.........................................................................................4
2. Reference Documentation..............................................................................................................4
  2.1 Standard Protocol Documentation.............................................................................................4
  2.2 Dusun standard documentation.................................................................................................4
3. Configuration.................................................................................................................................4
  3.1 Basic Configuration.....................................................................................................................4
    3.1.1 Deploy the environment of configuration.............................................................................4
    3.1.2 Login Password of Configuration Page..................................................................................5
    3.1.3 System Configuration...........................................................................................................6
    3.1.4 Administrator Configuration................................................................................................7
    3.1.5 Firmware Configuration....................................................................................................7
    3.1.6 Reboot..................................................................................................................................8
  3.2 Network Configuration...............................................................................................................8
    3.2.1 Network Interfaces Configuration........................................................................................8
    3.2.2 Interfaces-WAN Configuration............................................................................................8
    3.2.3 Interfaces – WAN6 Configuration.......................................................................................9
    3.2.4 Interfaces – WANPPP Configuration...................................................................................9
    3.2.5 Interfaces – LAN Configuration..........................................................................................10
    3.2.6 Network Diagnostics..........................................................................................................11
  3.3 MQTT server configuration.......................................................................................................11
  3.4 zigbee 3.0 manage.....................................................................................................................12
  3.5 Remote Assistance.....................................................................................................................12
  3.6 Z-WAVE Manage.......................................................................................................................13
  3.7 LET Configuration......................................................................................................................13
  3.8 Wi-Fi Configuration...................................................................................................................14
    3.8.1 AP mode..............................................................................................................................14
    3.8.2 STA mode............................................................................................................................14
4. Development Application on gateway.......................................................................................14
  4.1 UBUS in OPENWRT...................................................................................................................14
  4.2 Ubus Architecture.....................................................................................................................15
  4.3 Getting the zigbee sensors data from Ubus..............................................................................15
  4.4 Bluetooth Debug.......................................................................................................................16
    4.4.1 hcitool................................................................................................................................16
    4.4.2 gatttool...............................................................................................................................18
  4.5 Control all LED and buttons......................................................................................................18
  4.6 Building Tool.............................................................................................................................19
  4.7 Install the Application on gateway............................................................................................21
    4.7.1 Use the SCP command.......................................................................................................21
    4.7.2 Use opkg command............................................................................................................21
1. Introduction

1.1 Purpose & Description
The DSI-0134 is gateway product used in smart home, intelligent security industry, pension services. It’s a central device, support LTE-M1 (EMTC), Wi-Fi, zigbee3.0, Bluetooth and Z-WAVE wireless protocol. User can connect the network through LTE-M1, Wi-Fi and Ethernet. In addition, smart sensors can be connected through zigbee3.0, Bluetooth and z-wave.

1.2 Product Feature Summary
- **System:**
  - OS: Linux@ OpenWrt
  - Processor: MTK7620A (MIPS24KEc(580MHZ))
  - RAM: 128MB
  - Flash: 64MB
- **Wireless protocol:**
  - LTE-M1
  - Wi-Fi
  - Zigbee3.0
  - Bluetooth
  - Z-Wave
  - Ethernet
  - WLAN
  - LAN

1.3 System block diagram

![System Block Diagram](attachment:image.png)
## 1.4 Development Environment and Tool

<table>
<thead>
<tr>
<th>Development Environment</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>3.18.20</td>
</tr>
<tr>
<td>Development tool</td>
<td>Ubuntu + Crosstool</td>
</tr>
</tbody>
</table>

## 2. Reference Documentation

### 2.1 Standard Protocol Documentation

- Specifications for zigbee3.0
  - docs-05-3474-21-0csg-zigbee-specification.pdf
- Specifications for BT
  - Core_v4.2.pdf
- Specifications for MQTT
  - MQTT download: [http://mqtt.org/documentation/](http://mqtt.org/documentation/)

### 2.2 Dusun standard documentation

- Dusun_Communication_Protocol_between_Cloud_and_Gateway_V0.9606.pdf

## 3. Configuration

### 3.1 Basic Configuration

#### 3.1.1 Deploy the environment of configuration

![Diagram](image)  
Putting the PC connect to the LAN port of the gateway by netting twine.
3.1.2 Login Password of Configuration Page

Open the website, Input Gateway IP Address: 192.168.66.1

Enter the username and Password
Username: root
Password: root

In this page, you can check the status of system, memory, Network.
System Information: The host name, Model, Firmware Version, Kernel Version, Local Time.
Memory Information: Total Available, Free, Buffered.
Network: IPv4 WAN Status.
3.1.3 System Configuration

You can enter the “system” menu.

Here you can configure the basic aspects of your device like its hostname or time zone.
3.1.4 Administrator Configuration
You can enter the “administrator” menu.

You change the administrator password for accessing the device.

3.1.5 Firmware Configuration
You can enter the “Backup/flash” menu.

You can download the firmware from the current gateway to back up and restore. And you also can upload a new firmware image to replace the running firmware.
3.1.6 Reboot
Reboots the operating system of your device

3.2 Network Configuration
3.2.1 Network Interfaces Configuration
You can enter the "Interfaces" menu in <Network>.

3.2.2 Interfaces-WAN Configuration
On this page you can configure the network interfaces. You can bridge several interfaces by ticking the "bridge interfaces" field and enter the names of several network interfaces separated by spaces. You can also use VLAN notation INTERFACE.VLAN:NR (e.g.: eth0.1).
3.2.3 Interfaces – WAN6 Configuration
On this page you can configure the network interfaces. You can bridge several interfaces by ticking the "bridge interfaces" field and enter the names of several network interfaces separated by spaces. You can also use VLAN notation INTERFACE.VLANNR (e.g.: eth0.1).

3.2.4 Interfaces – WANPPP Configuration
On this page you can configure the network interfaces. You can bridge several interfaces by ticking the "bridge interfaces" field and enter the names of several network interfaces separated by spaces. You can also use VLAN notation INTERFACE.VLANNR (e.g.: eth0.1).
### Interfaces – LAN Configuration

On this page you can configure the network interfaces. You can bridge several interfaces by ticking the "bridge interfaces" field and enter the names of several network interfaces separated by spaces. You can also use VLAN notation INTERFACE.VLANNR (e.g.: eth0.1).

#### General Setup

<table>
<thead>
<tr>
<th>Status</th>
<th>Uptime: 1h 19m 60s</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC-Address: 00:1A:23:45:67:89</td>
<td>RX: 92.05 MB (5930 Pkts.)</td>
</tr>
<tr>
<td>TX: 1.17 MB (6297 Pkts.)</td>
<td></td>
</tr>
<tr>
<td>IPv4: 192.168.65.254</td>
<td>IPv6: fe80:2f9f:0:0:0:1:0:0</td>
</tr>
</tbody>
</table>

#### Protocol

- Static address
- IPv4 address: 192.168.65.1
- IPv4 netmask: 255.255.255.0
- IPv4 gateway
- IPv4 broadcast
- Use custom DNS servers
3.2.6 Network Diagnostics

![Network Diagnostics Image]

3.3 MQTT server configuration

You can enter the “MQTT server config” menu in <dusun>.

![MQTT Server Configuration Image]

You can configure the MQTT server: MQTT server Address, MQTT server Port, MQTT keep Alive Time, MQTT User Name, MQTT Password.
3.4 zigbee 3.0 manage
You can enter the “zigbee3.0 manage” menu in <dusun>.

You can press the <permit> button to pair with zigbee devices. If pair successful, you can see the sensor information.

3.5 Remote Assistance
You can enter the “remote assistance” menu in <dusun>.

It can Remote connect back so that doing some develop work and upgrade.
3.6 Z-WAVE Manage
You can enter the “Zwave Manage” menu in <dusun>.

- **Include button**: It can let gateway pair with sensors.
- **Exclude button**: It can let gateway unpair with sensors

3.7 LET Configuration
You can enter the “4G3G Arguments config” menu in <dusun>
You can set the APN in this page.

3.8 Wi-Fi Configuration
You can enter the “Wi-Fi manage” menu in <dusun>

<table>
<thead>
<tr>
<th>Dusun</th>
<th>Status</th>
<th>Socket</th>
<th>Network</th>
<th>Dusun</th>
<th>Login</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can set the mode of Wi-Fi, AP or STA.

3.8.1 AP mode
If you set AP mode, The gateway enter the AP mode. You can set the SSID ,Wi-Fi Encryption, and Wi-Fi key.

3.8.2 STA mode
If you set STA mode, The gateway enter the STA mode. You can set the SSID and WI-FI key that you want to connect the router.

4. Development Application on gateway
4.1 UBUS in OPENWRT
UBUS is a message bus in openwrt. Its main function is to realize information exchange between different applications. When UBUS starts, it runs the ubusd process in the background, which listens for a UNIX socket to communicate with other applications. Other applications can communicate with libubox based on the interface provided by libubox (or implemented by themselves).
4.2 Ubis Architecture

4.3 Getting the zigbee sensors data from Ubis

You can get the sensors data from ubus, and make your own application. The data format from UBUS can refer to Dusun_Communication_Protocol_between_Cloud_and_Gateway_V0.9606.pdf. Of course, you also can get a SDK. It’s very easy to operate sensor. The API as follow:

```c
int rbsdsk_init(void *unused, int (*msgcb)(char *src, char *dest, char *msg)); Function To Init Sdk.
int rbsdsk_vers(char *version); Function To Get SDK Version.
int rbsdsk_call(char *mac, char *attr, int setOrget, void *jvalue); A General Call Functions.
int rbsdsk_dev_msgcb_set(stDevMsgCb_t *dmc); Function to Set The Message Callback.
int rbsdsk_list_dev(); Function to Get Device list.
int rbsdsk_add_dev(char *mac, char *type); Function to Send Add Dev Request.
int rbsdsk_del_dev(char *mac); Function to Send Del Dev Request.
int rbsdsk_dev_lock_add_pass(char *mac, int type, int id, int passVal1, int passVal2, int startTime, int endTime, int suspend); Function to Send Add Lock PassWord Request.
int rbsdsk_dev_lock_del_pass(char *mac, int type, int id); Function to Send Delete Lock Password Request.
int rbsdsk_dev_lock_mod_pass(char *mac, int type, int id, int passVal1, int passVal2, int startTime, int endTime, int suspend); Function to Send Modify Lock Password Request.
int rbsdsk_dev_lock_clr_pass(char *mac, int type); Function to Send Clear Lock Password Request.
int rbsdsk_dev_lock_set_seed(char *mac, int id, int seed, int interval, int startTime, int endTime); Function to Send Set Dynamic Password Seed Request.
int rbsdsk_dev_light_onoff(char *mac, int onoff); Function To Send Light OnOff Request.
```
```c
int rbsdkg_dev_winctr_open(char *mac); // Function to Send Window Controller's Open Request.
int rbsdkg_dev_winctr_stop(char *mac);  // Function to Send Window Controller's Stop Request.
int rbsdkg_dev_winctr_close(char *mac);  // Function to Send Window Controller's Close Request.
int rbsdkg_dev_air_onoff(char *mac, int onoff); // Function to Send Air conditioner Onoff Request.
int rbsdkg_dev_air_mode(char *mac, int mode);  // Function to Send Air conditioner Mode Request.
int rbsdkg_dev_air_fan(char *mac, int fan);    // Function to Send Air conditioner Fan Request.
int rbsdkg_dev_air_temp(char *mac, int temp_mode, int dir, int temp_delt); // Function to Send Air conditioner temp Request.
```

4.4 Bluetooth Debug

There is one Bluetooth chip in our gateway that is CSR8510. It support HCI interface. And BlueZ 5.0 has been installed in the gateway. So you can use the hcitool and gatttool to debug the BLE function in openwrt system.

You can download the securecrt tool and use it to connect the gateway through ssh2.

4.4.1 hcitool

You can input the command to open Bluetooth device and scan the Bluetooth devices.

Command: `hciconfig`

```
$ hciconfig
hci0: Type: BR/EDR  Bus: USB
BD Address: 00:02:72:14:27:0E  ACL MTU: 1021:8  SCO MTU: 64:1
DOWN
RX bytes:2715 acl:1 sco:0 events:146 errors:0
TX bytes:2500 acl:0 sco:0 commands:133 errors:0
```

You can see the hci0, It’s the number from host. You can set up base on it.

Command: `hciconfig hci0 up`
Open the device.
Command: `hciconfig hci0 down`
Close the device.

After you open the device, you can use toolset to control the device. The tooling parameter as follow:
lewlsz  Read size of LE White List
lewclr  Clear LE White list
lecc    Create a LE Connection
ledc    Disconnect a LE Connection
lecup   LE Connection Update

For more information on the usage of each command use:
hcitool <command> --help

dev     Display local devices
inq     Inquire remote devices
scan    Scan for remote devices
name    Get name from remote device
info    Get information from remote device
spinq   Start periodic inquiry
epinq   Exit periodic inquiry
cmd     Submit arbitrary HCI commands
con     Display active connections
cc      Create connection to remote device
dc      Disconnect from remote device
sr      Switch master/slave role
cpt     Change connection packet type
rssi    Display connection RSSI
ljq     Display link quality
tpl     Display transmit power level
afh     Display AFH channel map
lp      Set/display link policy settings
lst     Set/display link supervision timeout
auth    Request authentication
4.4.2 gatttool

The purpose of using hcitool is to manage the connection of devices, so if we manage the BLE data finely, we need to use gatttool. The operation of sending instructions to Bluetooth devices using gatttool is much more complete than the CMD of hcitool. The use of gatttool can be divided into two kinds: one is to control Bluetooth devices directly by using parameters; the other is to use-I parameters to enter the inter of gatttool. Active mode controls Bluetooth devices.

The tooling parameter as follow:

```
1 # gatttool -h
2 Usage:
3 gatttool [OPTION...]
4 Help Options:
5 -h, --help
6 Show help options
7 --help-all
8 Show all help options
9 --help-gatt
10 Show all GATT commands
10 --help-params
11 Show all Primary Services/Characteristics args
12 --help-char-read-write
13 Show all Characteristics Value/Descriptor Res
14 Application Options:
15 -i, --adapter=hci0
16 Specify local adapter interface
17 -b, --device=MAC
18 Specify remote Bluetooth address
19 -t, --addr-type=[public | random]
20 Set LE address type. Default: public
21 -m, --mtu-MTU
22 Specify the MTU size
23 -p, --psm=PSM
24 Specify the PSM for GATT/ATT over BR/EDR
25 -l, --sec-level=[low | medium | high]
26 Set security level. Default: low
27 -I, --interactive
28 Use interactive mode
```

4.5 Control all LED and buttons

This is simple for openwrt, in the openwrt, we have export the led and buttons to filesystem, so you can operation the led or buttons the standard read and write, bellow is the buttons and led’s path:

```
root@dusun:~# ls /sys/class/leds/ -al
drwxr-xr-x 2 root root 0 Jan 1 1970 .
drwxr-xr-x 11 root root 0 Jan 1 1970 ..
```

For example, if you want to close the pwr led, you can only execute command:
```
echo none -> /sys/class/leds/pwrled/trigger; echo 0 > /sys/class/leds/pwrled/brightness
```

if you want to blink the pwrled, you can only execute command:
```
echo timer > /sys/class/leds/pwrled/trigger; echo 500 > /sys/class/leds/pwrled/delay_on; echo 500 > /sys/class/leds/pwrled/delay_off;
```

Tel: +86057185365259  Add: Room 801, Block A, Wantong Center, 189 Daguan Road, Gongshu District, Hangzhou, Zhejiang, China
If you want to edit the button functions, then you can edit the button response shell, below is the example for the BTN_0, it has two functions, one is the remote_shell and the other is greenpower pair, we can edit it to any functions we want.

So, if you want to operate the LEDs in app, you can execute shell command in the app or through standard linux io.

```bash
#!/bin/sh
./lib/functions.sh
./lib/functions/leds.sh

LED=yellow
if [ "$ACTION" = "pressed" ];
then
    touch /tmp/BTN_0_pressed
else
    rm -f /tmp/BTN_0_pressed
fi
if [ -f /tmp/dusun_upgrade ]; then
    exit 0
fi
if [ "$ACTION" = "released" -a "$SEEN" -ge 8 ];
then
    /usr/bin/remote_shell.sh &
    exit 0
fi
if [ "$ACTION" = "released" -a "$SEEN" -ge 1 ];
then
    echo now > /sys/class/leds/pcie伴ld/trigger
    echo 0 > /sys/class/leds/pcie伴ld/brightness
    echo 1 > /sys/class/leds/pcie伴ld/brightness
    led_send 65 GREENPOWER "cud" : "start_comision"
    sleep 1
    echo 1 > /sys/class/leds/pcie伴ld/brightness
fi
```

### 4.6 Building Tool

The crosstool for our app is the file that the name is <OpenWrt-SDK-ramps-mt7620_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-x86_64.tar.bz2>. You can get it from us.

The sample make file for cross compile C/C++ app as below:

```bash
{ARCH} := MT7620
ifeq ($(ARCH),MT7620)
  CROSS := mipsel-openwrt-linux-
  export STAGING_DIR := $(CROSSTOOLDIR)/staging_dir
  export PATH := $(PATH):$(STAGING_DIR)/toolchain-mipsel_24kec+dsps_gcc-4.8-linaro_uClibc-0.9.33.2/bin
  CROSS_CFLAGS := -I$(CROSSTOOLDIR)/staging_dir/toolchain-mipsel_24kec+dsps_gcc-4.8-linaro_uClibc-0.9.33.2/usr/include
  CROSS_CFLAGS := -I$(CROSSTOOLDIR)/staging_dir/toolchain-mipsel_24kec+dsps_gcc-4.8-linaro_uClibc-0.9.33.2/usr/include
  CROSS_LDFLAGS := -I$(CROSSTOOLDIR)/staging_dir/target-mipsel_24kec+dsps_gcc-4.8-linaro_uClibc-0.9.33.2/usr/include
  CROSS_LDFLAGS := -I$(CROSSTOOLDIR)/staging_dir/toolchain-mipsel_24kec+dsps_gcc-4.8-linaro_uClibc-0.9.33.2/usr/lib
```
CROSS_LDFLAGHS
-L$(CROSSSTOOLDIR)/staging_dir/target-mipsel_24kec+dsp_uClibc-0.9.33.2/usr/lib/
endif

CC := $(CROSS)gcc
LDFLAGS := $(CROSS_LDFLAGHS)
#LDFLAGS += -ldusun -ljson-c -lmosquitto -lpthread -lubox -lubus -lblobmsg_json
-lcyassl -lcares -lmosquitto -lssl -lcrypto
LDFLAGS += -ldusun -ljson-c -lpthread -lubox -lubus -lblobmsg_json -lcyassl

CFLAGS := -Wall -Wno-unused-variable -Wno-unused-function -g
-DOPENWRT
CFLAGS += $(CROSS_CFLAGS)
LOCAL_CFLAGS := $(CFLAGS) -DIHD_TEST
LOCAL_CFLAGS += -I./ -I./serial -I./new

all: nxp

SRCS = main.c cmd.c timer.c

#NXP Jennet related files
SRCS += serial/Serial.c serial/SerialLink.c nxp.c curve25519-donna.c util.c
SRCS += security.c sha1.c ota.c lock.c meter.c fp_temp_put.c infrared.c
SRCS += ./new/protox.c ./new/platform.c ./new/securityx.c ./new/ramx.c ./new/cmd_ver2.c ./new/util_ver2.c

#files related to server protocol
#SRCS+= cloud/cloud.c

nxp: FORCE
  @echo $(LDFLAGS)
  $(CC) -std=gnu99 -o $@ $(LOCAL_CFLAGS) $(SRCS) $(LDFLAGS)

clean:
  rm -rf ./nxp
  rm -rf ./tags

scp:
  #scp -P 22 ./nxp root@192.168.10.50:/tmp
  scp -P 22 ./nxp root@192.168.10.233:/tmp

cp:
  Tel: +86057185365259
  Add: Room 801, Block A, Wantong Center, 189 Daguan Road, Gongshu District, Hangzhou, Zhejiang, China
FORCE:
}

You need to edit the SRCS & CFLAGS & LDFLAGS for your app.

4.7 Install the Application on gateway
4.7.1 Use the SCP command
You can use the SCP command to transfer app file to the board and operation in the board on
Linux.

$ scp -P22 ./app root@192.168.10.230:/tmp/

4.7.2 Use opkg command
if you are familiar with openwrt, you can packet the app to a opkg packet and then install it on the
board through the opkg command, this command is the standard ways to install app on openwrt
system, you can search the ways to packet the opkg on google.