

iSense Gateway Module User Guide

Document history

<i>Version</i>	<i>Date</i>	<i>Changes</i>
1.0	13.05.2009	Initial version

Contents

1.	About this User Guide	4
2.	General Description	5
2.1.	Handling and Security Instructions	6
2.2.	Using the potentiometer	7
2.3.	Preventing External Reset of the Gateway Module	7
2.4.	Using both USB and Serial Cable	7
3.	GatewayModule API Description	8
3.1.	Constructor	8
3.2.	led_on	9
3.3.	led_off	9
3.4.	allow_external_reset	10
3.5.	set_button_handler	10
4.	ButtonHandler API Description	11
4.1.	button_down	11
4.2.	button_up	11
5.	Gateway Module Demo Application	13
5.1.	Obtaining the Gateway Module Demo Application	13
5.2.	Compiling the Gateway Module Demo Application	13
5.2.1.	Using Eclipse	13
5.2.2.	Using the Command Line	17
5.3.	Flashing the Gateway Module Demo Application	17
5.4.	Gateway Module Demo Application functionality	17
6.	References	23

1. About this User Guide

In this user guide,

- files and folders are represented in the `Arial` typeface,
- code fragments, function names etc. are represented in the `Courier New` typeface,
- GUI elements such as button descriptions etc. are represented in “quotation marks”,
- titles of other documents are presented in *Italic* type.

This manual assumes that the reader has successfully installed the iSense development environment, and obtained the iSense standard firmware. For further information on these steps, consult the *Development Environment Setup User Guide* [1].

In addition, it is assumed that the user is familiar with the use of iShell. For further information on iShell, consult the *iShell User Guide* [2].

For further information on iSense firmware programming concepts and on application development, it is recommended to read the *Writing iSense Applications User Guide* [3].

2. General Description

The iSense Gateway Module provides interconnection of iSense sensor nodes with other systems such as personal computers using a serial connection via USB and/or RS-232. It enables data exchange as well as serial programming of connected iSense Gateway Modules.

The USB connector can also be used to power other attached iSense modules, including the Lithium-Ion Rechargeable Battery Module and the iSense Solar Power Management Module.

In addition, the Gateway Module provides 3 LEDs, 2 buttons and a potentiometer.

Internally, the Gateway Module uses an I2C port expander for reading buttons and switching LEDs. The port expander generates interrupts on the DIO4/CTS0/I2C_IO_EXP_INT pin of X2 if the buttons are pressed or released.

The Gateway Module can also be used to program a connected Core Module. To do so, it can

- pull down the SPI_MISO pin of X2 by pulling down the RTS line of the serial or USB connection and
- pull down the RESET pin of X2 by pulling down the DTR line of the serial or USB connection.

This external reset of the controller via the Gateway Module can be disabled via a software switch.

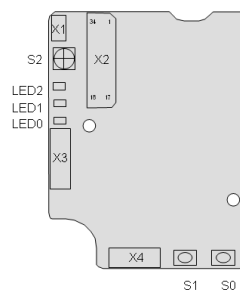


Figure 1 shows the top side of the iSense Gateway Module, the below table lists its features.

X1	If closed, the potentiometer can be used to vary the voltage level of the ADC input 3 (pin ADC3 of the system connector X2)
X2	System connector for plugging an iSense Core Module or other iSense modules
X3	Serial cable connector
X4	USB cable connector
LED0-2	Software switchable LEDs
S0, S1	User buttons
S2	Potentiometer

2.1. Handling and Security Instructions

Note that all iSense Modules are sensitive to electro-static discharge. Appropriate protection measures have to be taken.

If not protected by an appropriate housing, all iSense components must be protected from humidity, mechanical impact, and short circuiting by contact with conductive materials.

Note that all cable plugs and headers are designed to be extremely compact, and are not intended for frequent plugging and unplugging. Never pull the cables to remove the plugs from the corresponding modules, always pull the plug itself. If required, use an appropriate tool (or the pulling cord if available). Even though all plugs/headers are coded, be sure to pay attention to the proper orientation. Never apply force. Never use cables if their insulation is damaged.

Note that iSense modules are not intended for hot-plugging, (dis-)connecting modules from or to other modules in operation can result in a reset of the Core Module and other undesired effects.

All delivered components are intended for use in research application. For using these components within other application domains, consult coalesenses prior to use.

Coalesenses products are not intended for use in life support systems, appliances or systems where malfunction of these products can reasonably be expected to result in personal injury, death or severe property or environmental damage. Coalesenses customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify coalesenses for any damages resulting from such use.

Do not connect components or devices to any iSense component that are not manufactured or distributed by coalesenses without prior consultation of coalesenses. Note that doing so regardless of consultation will void the warranty and the declaration of CE conformity. Coalesenses customers connecting third party devices do so at their own risk and agree to fully indemnify coalesenses for any damages or injury resulting from such use.

If a wall mount adapter is connected to the Core Module, or a voltage is applied to pins 18 and/or 19 (V+_USB) of the system connector X1/X2 (e.g. because an iSense Gateway Module that is connected to an active USB port is attached to the Core Module), currents may flow back into devices or components connected to X3, X4 or the programming pads.

2.2. Using the potentiometer

To use the potentiometer, the jumper X1 has to be closed. Then, voltage applied to pin ADC3 of a connected Core Module is determined by the potentiometer on the Gateway Module. To use that ADC input of the Core Module differently when the Gateway Module is connected, the jumper must be removed.

If X1 is closed, turning the potentiometer carefully makes the voltage applied to the ADC3 pin of the controller vary between approximately 0V and 2.4V,

2.3. Preventing External Reset of the Gateway Module

As stated before, the Gateway Module can be used to pull down the RESET pin of X2 by pulling down the DTR line of the serial or USB connection, i.e. to externally reset the sensor node. However, some USB and RX-232 drivers pull this line upon plugging or unplugging the cable to the PC. To avoid undesired reset of a battery driven iSense sensor node that comprise a Gateway Module and that is connected to or disconnected from a PC while running, the external reset feature of the Gateway Module can be disabled via an API call (see Section 3.4)

Warning!

Note that while the external reset feature of the Gateway Module is disabled, the connected sensor node cannot be programmed via the Gateway Module using the iShell Flash Programmer.

By default, the external reset feature of the Gateway Module is enabled.

It is strongly recommended that applications do not disable the external reset feature automatically directly after boot, because you will not be able to reprogram the sensor node in case your program hangs up. Instead, you should manually disable the external reset (e.g. by pressing a button, or sending a message to the sensor node using the iShell Messenger plugin).

2.4. Using both USB and Serial Cable

It is possible to connect the Gateway Module to a PC using both the USB cable and the RS-232 cable in parallel. However, in this case, the RS-232 connection dominates the USB connection. In other words, the serial communication via USB is suppressed, and communication is possible only via RS-232. Nevertheless, power supply of the sensor node is still possible via USB.

3. GatewayModule API Description

The `GatewayModule` class in the `isense` namespace contains all software functionality for operating the Gateway Module-specific features, including

- switching the LEDs,
- allowing or disabling the external reset of the sensor node
- observing the buttons.

The `GatewayModule` class is defined in `src/isense/modules/gateway_module/gateway_module.h` in the `iSense` directory.

The `iSense` firmware is structured into a large number of software modules that can be (de-)activated separately. The below API description for each function specifies the required modules that must be activated to use the functions, and names both the web compilation module name as well as the according code define. An asterisk (*) indicates that the respective module is activated in the `iSense` standard firmware.

The API description also indicates the make targets for which the different functions are available.

3.1. Constructor

```
GatewayModule::GatewayModule(Os &os);
```

Description:

Initializes the `GatewayModule` class. By default, the external reset is allowed, and the LEDs are switched off.

Parameters:

`os` Reference to the operating system class `Os`

Required modules:

Gateway Module* `#define ISENSE_ENABLE_GATEWAY_MODULE`

Available for the targets:

JN5139R, JN5139R1

3.2.led_on

```
void GatewayModule::led_on(uint8 led);
```

Description:

Switches on one of the Gateway Module LEDs (regardless of prior state).

Parameters:

led number of the LED to be switched on. Supported values are 0, 1 or 2. For all other values, led is computed to $\text{led} \% 3$.

Required modules:

Gateway Module* `#define ISENSE_ENABLE_GATEWAY_MODULE`

Available for the targets:

JN5139R, JN5139R1

3.3.led_off

```
void GatewayModule::led_off(uint8 led);
```

Description:

Switches off one of the Gateway Module LEDs (regardless of prior state).

Parameters:

led number of the LED to be switched off. Supported values are 0, 1 or 2. For all other values, led is computed to $\text{led} \% 3$.

Required modules:

Gateway Module* `#define ISENSE_ENABLE_GATEWAY_MODULE`

Available for the targets:

JN5139R, JN5139R1

3.4.allow_external_reset

```
void GatewayModule::allow_external_reset(bool allow);
```

Description:

Enables or disables the external reset feature of the Gateway Module.

Warning:

While the external reset is disabled, the connected sensor node cannot be reprogrammed via the Gateway Module with the iShell “Flash Loader” plugin. Please refer to Section 2.3 before using the feature.

Parameters:

allow	true	enables the external reset
	false	disables the external reset

Required modules:

Gateway Module*	#define ISENSE_ENABLE_GATEWAY_MODULE
-----------------	--------------------------------------

Available for the targets:

JN5139R, JN5139R1

3.5.set_button_handler

```
void GatewayModule::set_button_handler( ButtonHandler* bh);
```

Description:

Sets the object that bh points to as the button handler for the GatewayModule. Note that the class of bh must inherit from isense::ButtonHandler (see Section 4). If a different ButtonHandler was set before, it will be unset, and the new handler will be set. When a button is pressed, bh->button_down(...) will be called, when the button is released, bh->button_up(...) will be called. For unsetting a ButtonHandler without setting a new handler, simply pass NULL.

Parameters:

bh	Pointer to the ButtonHandler that should be called upon button press or release. To unset the currently set handler, pass NULL
----	--

Required modules:

Gateway Module*	#define ISENSE_ENABLE_GATEWAY_MODULE
-----------------	--------------------------------------

Available for the targets:

JN5139R, JN5139R1

4. ButtonHandler API Description

The `ButtonHandler` class is defined in `src/isense/button_handler.h` in the `iSense` directory.

It must be inherited by classes that are to be set at the `GatewayModule` class as a `ButtonHandler` using the method `GatewayModule::set_button_handler(ButtonHandler* bh)`.

```
namespace isense
{
    class ButtonHandler :
        public iSenseObject
    {
        public:

            virtual void button_down (uint8 button_no) {}
            virtual void button_up (uint8 button_no) {}

    };
}
```

4.1.button_down

```
void ButtonHandler::button_down( uint8 button_no );
```

Description:

Overwrite this method if your class inheriting from `ButtonHandler` should be called if a button is pressed. If your class was set as the Gateway Module button handler before (see Section 3.5), it will be called when a button is pressed.

Parameters:

`button_no` number of the button that was pressed, i.e. 0 for S0 and 1 for S1

Required modules:

none

Available for the targets:

all

4.2.button_up

```
void ButtonHandler::button_up( uint8 button_no );
```

Description:

Overwrite this method if your class inheriting from `ButtonHandler` should be called if a button is released. If your class was set as the Gateway Module button handler before (see Section 3.5), it will be called when a button is released.

Parameters:

`button_no` number of the button that was released, i.e. 0 for S0 and 1 for S1

Required modules:

none

Available for the targets:

all

5. Gateway Module Demo Application

The Gateway Module demo application exemplifies how an application can make use of the GatewayModule API. It shows how to

- switch the LEDs,
- allow or disable the external reset of the sensor node
- observe the buttons.

5.1. Obtaining the Gateway Module Demo Application

Go to coalesenses web site, click on “Downloads”, and then on “iSense Hardware”. In the “Gateway Module” section, click on the icon at the right in the “Gateway Module Demo Application” row, and download GatewayModuleDemoApplication.zip.

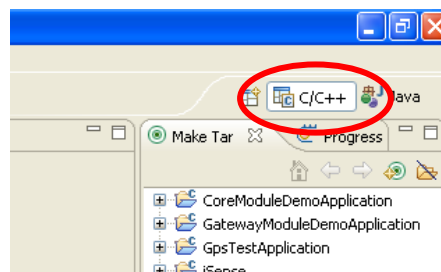
Extract the content of GatewayModuleDemoApplication.zip to the iApps directory.

5.2. Compiling the Gateway Module Demo Application

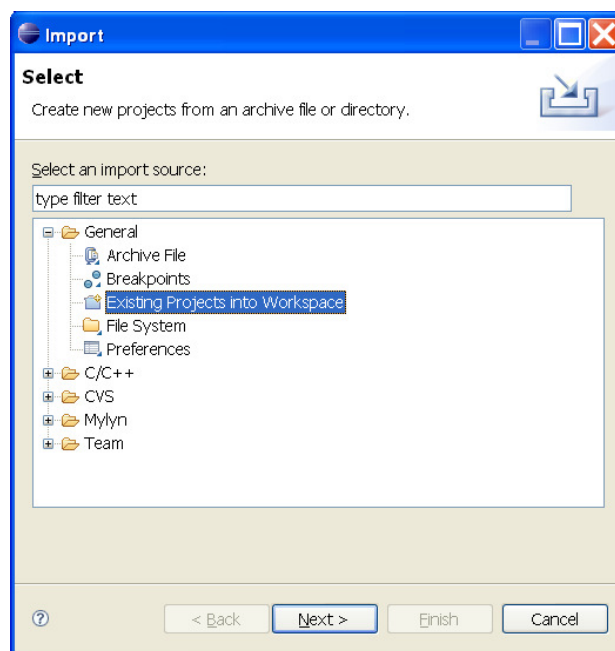
The next step is compiling the Gateway Module Demo Application.

5.2.1. Using Eclipse

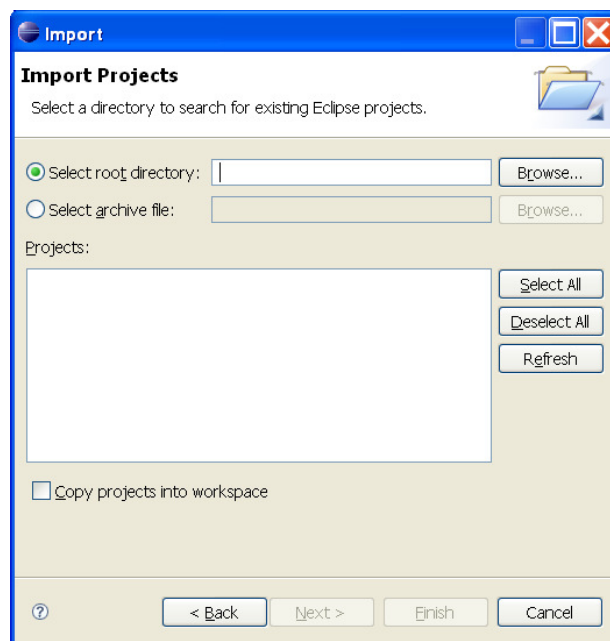
If you want to use Eclipse for compiling the Gateway Module Demo Application, open Eclipse, and change to the “C++” perspective.



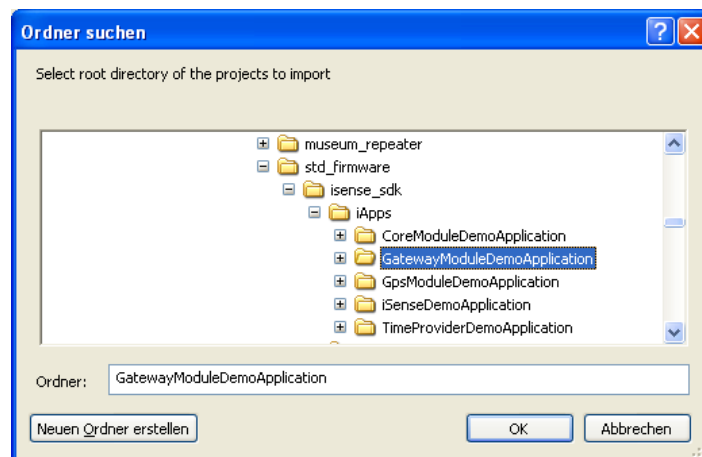
You can now import the application into Eclipse. Choose “File” → “Import” from the menu bar to open the “Import” dialog.



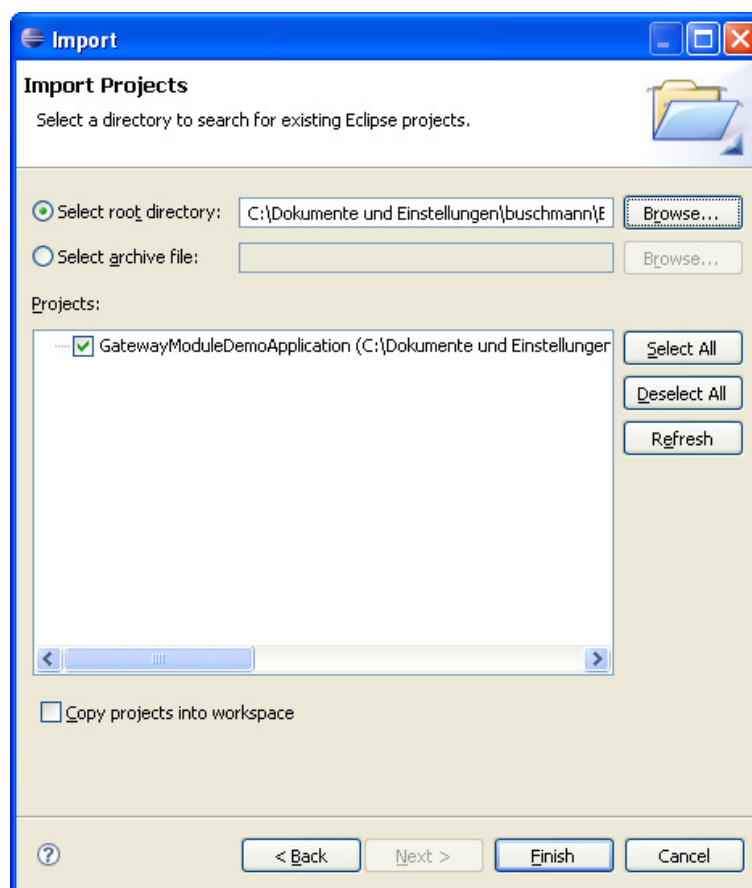
Choose “Existing projects into workspace”, and click on “Next”.



Click on the “Browse...”, and select the directory of the Gateway Module demo application, i.e. GatewayModuleDemoApplication in the iApps directory.



After doing so, the “GatewayModuleDemoApplication” project should appear in the projects list of the “Import” dialog.

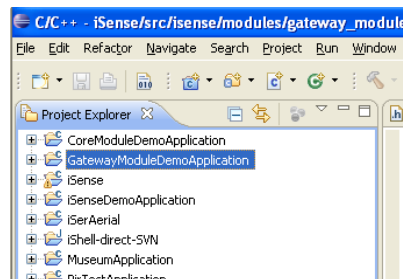


If it doesn't, the typical reason is that there is already a project called “GatewayModuleDemoApplication” in Eclipse, i.e.

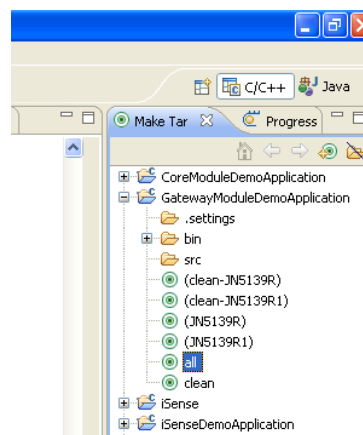
- you imported the GatewayModuleDemoApplication before or
- you created or imported a project with the same name before.

Be sure not to check “Copy projects into workspace” in the above dialog. Finish the import by clicking on the “Finish” button.

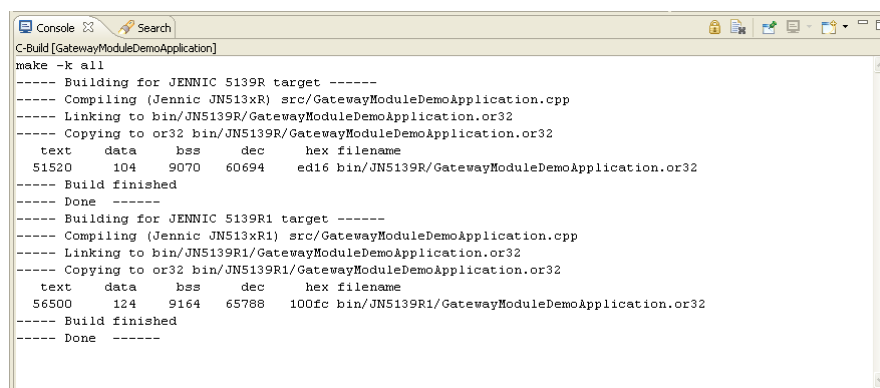
As a result, you will find the “GatewayModuleDemoApplication” project in the “Project Explorer”.



In addition, the “GatewayModuleDemoApplication” will appear in the “Make Targets” view. Double click on “all” to build the demo application for all targets.

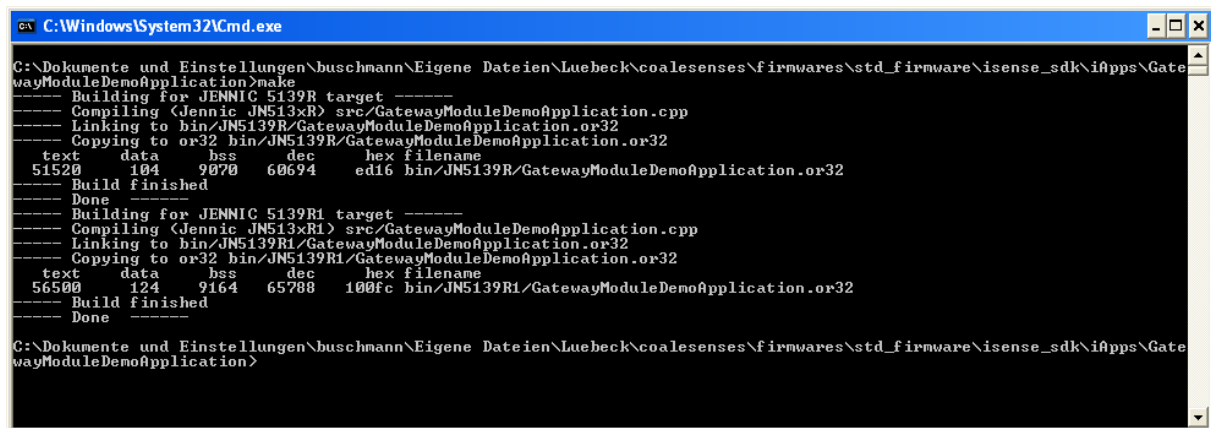


As a result, the “Console” view displays the compiler output during the build process. After that, it should look similar to the output depicted below.



5.2.2. Using the Command Line

If you want to use the command line tools for building the Gateway Module Demo Application, open a console window, change to the GatewayModuleDemoApplication directory in your iApps directory, and type “make”.



```
C:\Windows\System32\cmd.exe
C:\Dokumente und Einstellungen\buschmann\Eigene Dateien\Luebeck\coalesenses\firmwares\std_firmware\isense_sdk\iApps\GatewayModuleDemoApplication>make
-----
Building for JENNIC 5139R target -----
Compiling (Jennic JN513xR) src/GatewayModuleDemoApplication.cpp
Linking to bin/JN5139R/GatewayModuleDemoApplication.or32
Copying to or32 bin/JN5139R/GatewayModuleDemoApplication.or32
text    data    bss    dec    hex filename
51520    104     9070    60694    ed16 bin/JN5139R/GatewayModuleDemoApplication.or32
-----
Build finished
Done
-----
Building for JENNIC 5139R1 target -----
Compiling (Jennic JN513xR1) src/GatewayModuleDemoApplication.cpp
Linking to bin/JN5139R1/GatewayModuleDemoApplication.or32
Copying to or32 bin/JN5139R1/GatewayModuleDemoApplication.or32
text    data    bss    dec    hex filename
56500    124     9164    65788    100fc bin/JN5139R1/GatewayModuleDemoApplication.or32
-----
Build finished
Done
-----
C:\Dokumente und Einstellungen\buschmann\Eigene Dateien\Luebeck\coalesenses\firmwares\std_firmware\isense_sdk\iApps\GatewayModuleDemoApplication>
```

After the build finished, the console output should look similar as shown above.

5.3. Flashing the Gateway Module Demo Application

Connect either

- an iSense Core Module and
- an iSense Gateway Module with a USB cable

or

- an iSense Core Module,
- an iSense Battery Module and
- an iSense Gateway Module with a serial cable

and plug the USB/serial cable to the PC. Be sure that the Core Module is switched on.

Start iShell, configure the correct communication port, and change to the “Flash Loader” view. Select the appropriate binary according to the table below.

Chip	Binary file
JN5139R	iApps/GatewayModuleDemoApplication/bin/JN5139R/GatewayModuleDemoApplication.bin
JN5139R1	iApps/GatewayModuleDemoApplication/bin/JN5139R1/GatewayModuleDemoApplication.bin

For details regarding the identification of the chip version, refer to Section 2.4 of the *iSense Core Module User Guide* [4]. Then flash the binary to the connected iSense sensor node.

5.4. Gateway Module Demo Application functionality

The source file of the Gateway Module demo application is located at GatewayModuleDemoApplication/src/GatewayModuleDemoApplication.cpp. The class

GatewayModuleApplication is derived from `isense::Application`, and additionally inherits from `isense::ButtonHandler`.

```
class GatewayModuleDemoApplication :
    public Application,
    public ButtonHandler
{
public:
    GatewayModuleDemoApplication(Os &os);
    // inherited from application, called upon device boot
    void boot();
    //inherited from ButtonHandler, called when a button is pressed
    void button_down(uint8 button_no);
private:
    GatewayModule* gm_;
    //currently active Gateway Module LED
    uint8 led_;
    //current reset allowance state
    bool reset_allowed_;
};
```

GatewayModuleDemoApplication has three member variables that are initialized in the constructor. `gm_` holds the pointer to the GatewayModule instance allocated in `boot`, `led_` keeps track of which Gateway Module LED is currently selected, and `reset_allowed_` keeps track of whether the external reset feature of the Gateway Module is currently enabled or not.

```
GatewayModuleDemoApplication::
GatewayModuleDemoApplication(Os &_os) :
    Application(_os),
    gm_(NULL),
    led_(0),
    reset_allowed_(true)
{
}
```

GatewayModuleDemoApplication overwrites `isense::Application::boot()`, which is called upon the start up of the iSense sensor node.

```
void
GatewayModuleDemoApplication::
boot ()
{
    os().debug("Booting Gateway Module Demo Application, id=%x",
                                                       os().id());

    //create GatewayModule instance
    gm_ = new GatewayModule(os());
}
```

```
if (gm_ != NULL)
{
    os().debug("Press button 0 for LED testing"
               "and button 1 for reset prevention.");
    //set application as button handler -->
    //button_down called upon button press
    gm_>set_button_handler(this);
} else
    os().fatal("Could not allocate memory for GatewayModule");
}
```

Within the `boot()` method, at first a boot notification message is output.

You can send character output to the outside world using the methods `Os::debug` and `Os::fatal`:

```
//-----
/** Logs the given string depending on the log mode to a uart or radio.
 */
void debug( const char *format, ...);

//-----
/** Logs the given string depending on the log mode to a uart or radio.
 */
void fatal( const char *format, ...);
```

They work similar to the well-known `sprintf` functions in regular C, i.e. integer values etc. can be printed using the `%` notation. For example

```
uint16 i = 128;
os().debug("value of i is %d, hex=%x", i, i);
```

will output „value of i is 128, hex=0x80“. The destination of the debug output can be set using `Os::set_log_mode`, the corresponding constants can be found in `iSense/src/isense/os.h`.

```
//-----
/** Sets the log mode to the given value, e.g.
 * (ISENSE_LOG_MODE_UART0 | ISENSE_LOG_MODE_RADIO)
 */
void set_log_mode( uint8 mode ) {log_mode_ = mode; }
```

By default, the output destination is set to UART0, i.e. the output is sent to a PC via a connected iSense Gateway Module.

Then, in `GatewayModuleDemoApplication::boot()`, a `GatewayModule` object is created. If the allocation was successful, the `GatewayModuleDemoApplication` is set as the `GatewayModule`'s `ButtonHandler`.

```
//set application as button handler -->
//button_down called upon button press
gm_>set_button_handler(this);
```

This is possible because the application inherited the `ButtonHandler` interface. It overwrites its `button_down` method, that is called whenever a Gateway Module button is pressed.

```
void
GatewayModuleDemoApplication::
button_down( uint8 button_no )
{
    os().debug("Button %d pressed.", button_no);
    if (button_no == 0)
    {
        switch (led_)
        {
            case 0: // currently LED 0 on
                //switch of LED 0 and switch on LED 1
                gm_>led_off(0);
                gm_>led_on(1);
                led_=1;
                break;
            case 1: // currently LED 1 on
                //switch of LED 1 and switch on LED 2
                gm_>led_off(1);
                gm_>led_on(2);
                led_=2;
                break;
            case 2: // currently LED 2 on
                //switch of LED 2 and switch on LED 0
                gm_>led_off(2);
                gm_>led_on(0);
                led_=0;
                break;
        }
        os().debug("Switched on LED %d", led_);
    }
    if (button_no == 1)
    {
        if (reset_allowed_) //currently external reset allowed
        {
            //prevent external reset
            gm_>allow_external_reset(false);
            os().debug("External reset forbidden.");
            reset_allowed_=false;
        } else { //currently external reset prevented
            // allow external reset
            gm_>allow_external_reset(true);
            os().debug("External reset allowed.");
            reset_allowed_=true;
        }
    }
}
```

```
    }  
  
    }  
}
```

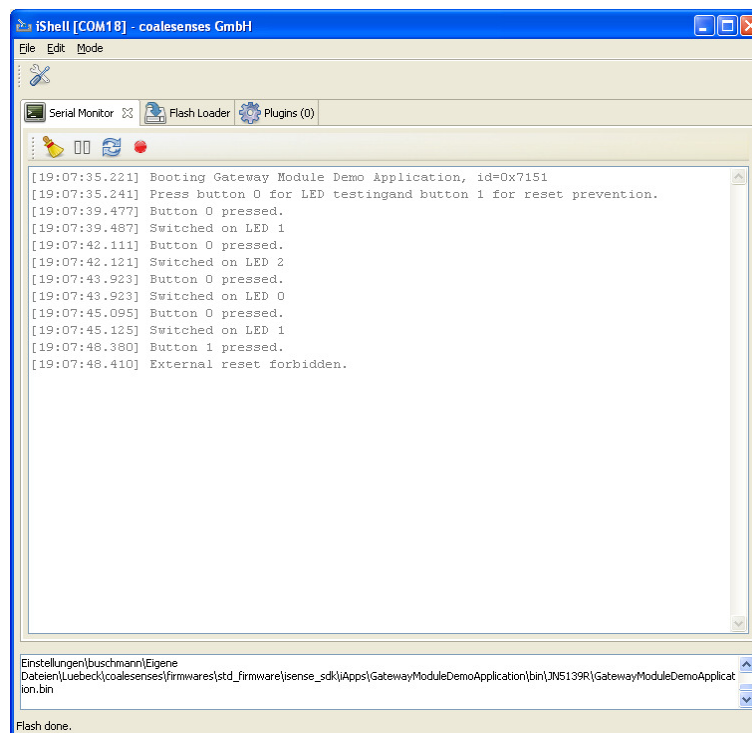
At first, a debug message indicating the number of pressed button is output.

In case it was button S0 (i.e. `button_no == 0`), the first `if` statement is `true`, i.e. the currently active LED (as tracked in the variable `led_`) is switched off, the next LED is switched on and an according message is output.

In case it was button S1 (i.e. `button_no == 1`), the second `if` statement is `true`, i.e. the allowance state of the external reset feature of the Gateway Module is toggled.

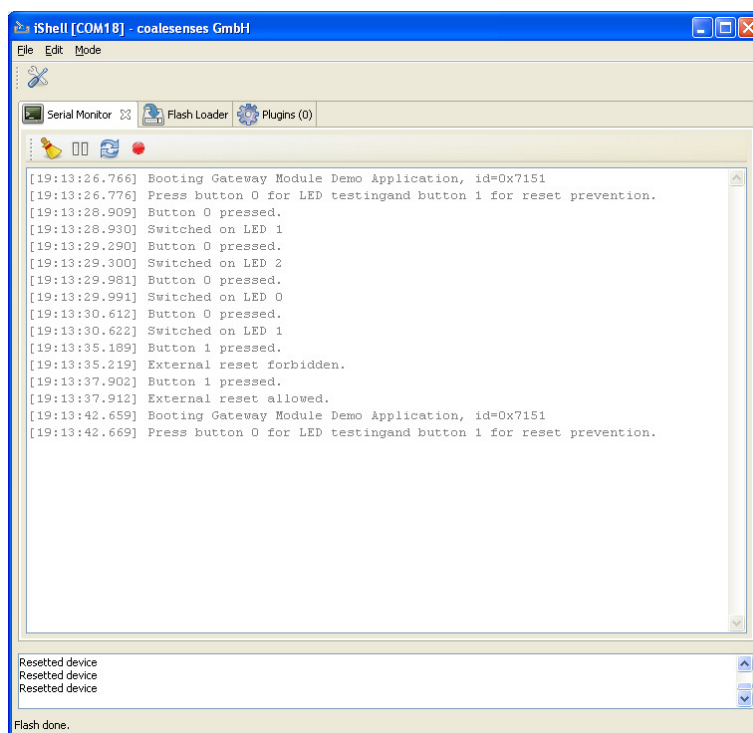
All in all, the `GatewayModuleDemoApplication` switches through the LEDs if button S0 of the Gateway Module is pressed, and toggles whether external device resets are allowed or prevented if button S1 is pressed.

Note that if the external reset is prevented, the sensor node cannot be programmed via the Gateway Module with `iShell`. However, as the external reset is allowed after boot, you can always switch the Core Module off and on again, and the external reset (i.e. programming with `iShell`) will be possible again.



Observing the output of the `GatewayModuleDemoApplication` with `iShell` yields the above result. Here, button S0 was pressed four times, and then button S1 once. The latter disabled the external reset feature of the Gateway Module. If you now click on the reset button (🔄) in the “Serial Monitor” plugin, nothing will happen.

If you press S1 a second time, the reset will be possible again. If you now click on the reset button, the device will restart and output the boot notification message (see below).



6. References

- [1] coalesenses Development Environment Setup User Guide, online available at http://www.coalesenses.com/download/UG_development_environment_setup_v1.9_web.pdf
- [2] coalesenses iShell User Guide, online available at http://www.coalesenses.com/download/UG_ishell_v1.3.pdf
- [3] coalesenses Writing iSense Applications User Guide, online available at http://www.coalesenses.com/download/UG_writing_isense_applications_v1.pdf
- [4] coalesenses iSense Core Module User Guide, online available at http://www.coalesenses.com/download/UG_CM10X_1v0.pdf

coalesenses GmbH
Maria-Goeppert-Str. 1
23562 Lübeck
Germany

www.coalesenses.com
sales@coalesenses.com