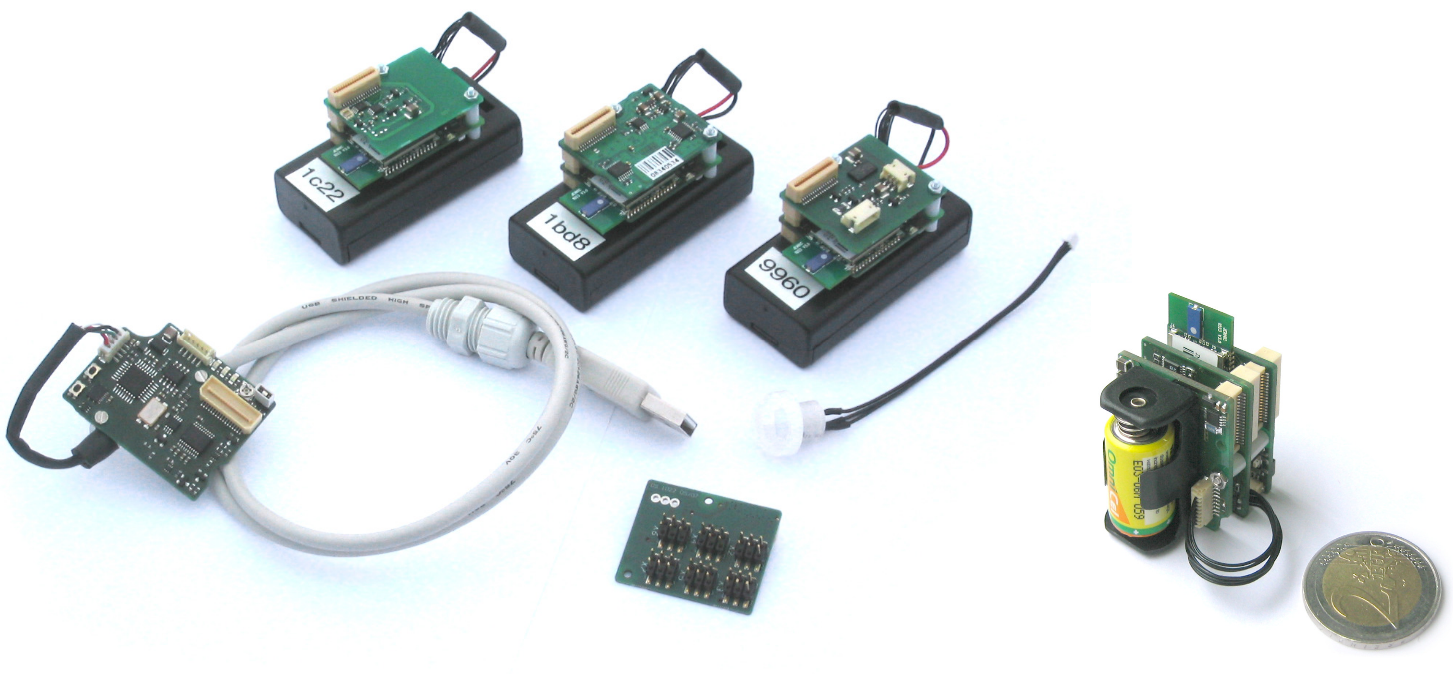


iShell User Guide

coalesenses
research to innovate

Document history

Version	*Date*	*Changes*
1.0	14.05.2007	Initial version
1.1	14.06.2007	iShell plugin API added
1.2	14.02.2008	Integrated plugins description added
1.3	01.05.2009	Reformatted, section on connections added, added more detailed information in nearly all parts of the document, added sections on time provider, JPEG viewer and data collector
1.4	31.07.2009	Added section on Hibernation plugin
1.5	04.06.2010	Updated section on Over-the-Air-Programming

Contents

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 theses steps, consult the *Development Environment Setup User Guide* 9.

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 of iShell*

The iShell programming, operating and analysis tool is the personal computer counterpart of the iSense operating and networking firmware. Besides a broad variety of functions for operating iSense wireless sensor networks that is already included, iShell can easily be extended through adding user plugins.

iShell offers its functionality to the user in a number of different views that group activities on separate tabulator windows.

First of all iShell can be used for both wired and wireless programming of the iSense devices. Besides, the serial monitor view provides a text window for displaying messages transmitted from a connected device via a serial or USB connection. Other views for example

- send messages to the network via an attached sensor node,
- plot graphs of measured data captured in the network e.g. by the vehicle detection sensor or the accelerometer
- provide data export to Microsoft Excel
- display camera pictures captured by wireless sensors

Besides use with real-world iSense hardware, iShell can also be connected to the wireless sensor network simulator Shawn via TCP/IP. Shawn is integrated into the iSense firmware, and can simulate iSense code directly. Applications just need to be compiled for the simulation target to be ready to run within the simulator.

3. Connecting iShell

iShell can be used to connect to iSense hardware as well as to the wireless sensor network simulator Shawn via TCP/IP. Shawn is integrated into the iSense firmware, and can simulate iSense code directly. Applications just need to be compiled for the simulation target to be ready to run within the simulator.

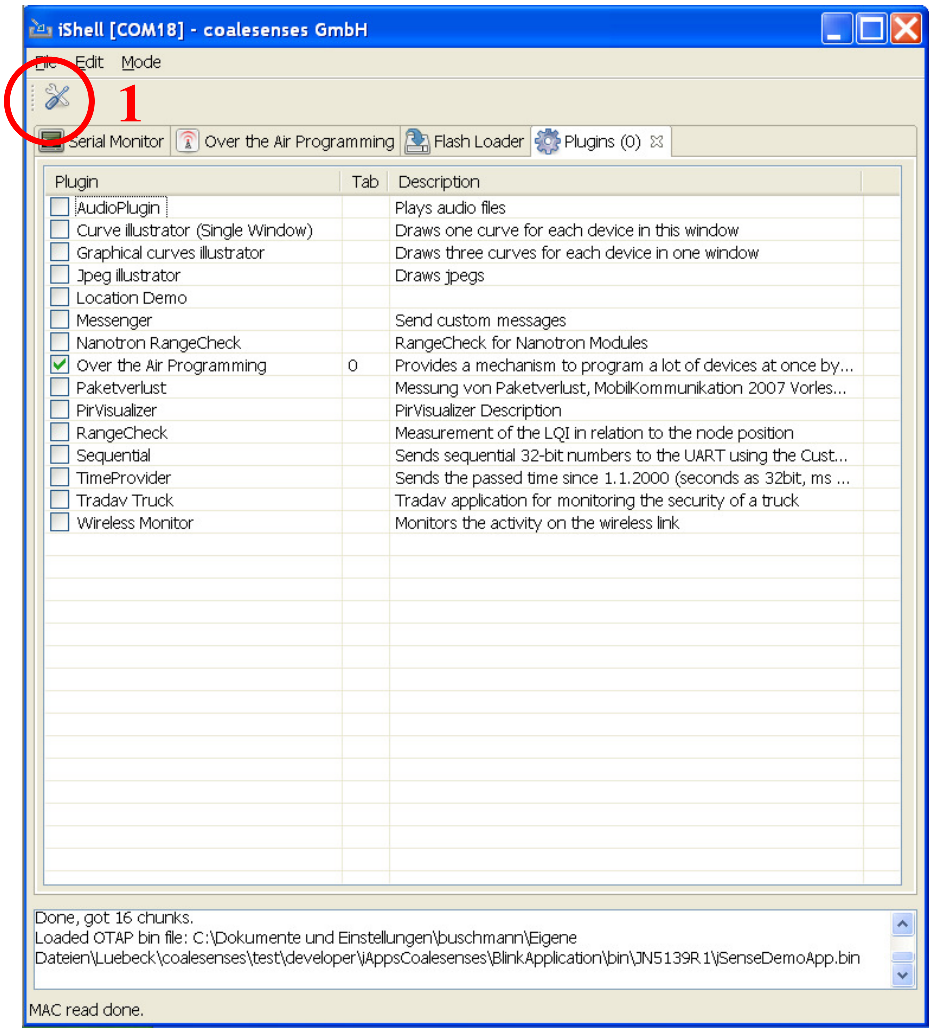
3.1. Connecting to an iSense device

Connect an iSense device to the PC. It must at least consist of

- an iSense Core Module and
- an iSense Gateway Module connected to the PC via a USB cable,

or

- an iSense Core Module,
- an iSense Gateway Module connected to the PC via a serial cable, and
- an iSense Power Module.

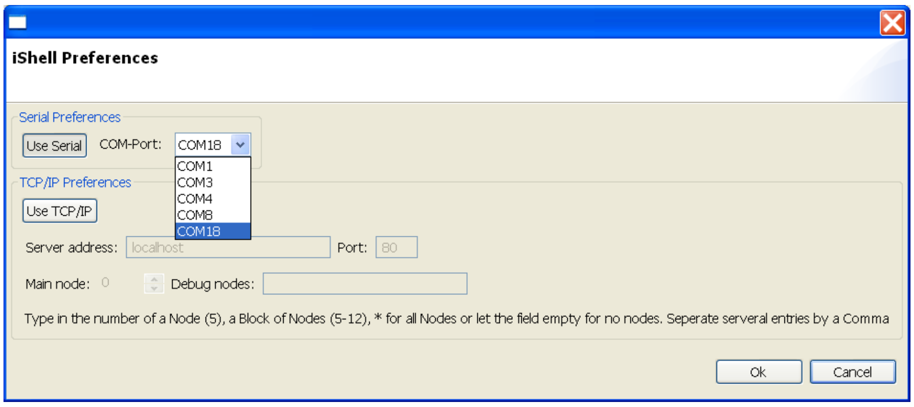


Plugin	Tab	Description
☐ AudioPlugin		Plays audio files
☐ Curve illustrator (Single Window)		Draws one curve for each device in this window
☐ Graphical curves illustrator		Draws three curves for each device in one window
☐ Jpeg illustrator		Draws jpegs
☐ Location Demo		
☐ Messenger		Send custom messages
☐ Nanotron RangeCheck		RangeCheck for Nanotron Modules
✓ Over the Air Programming	0	Provides a mechanism to program a lot of devices at once by...
☐ Paketverlust		Messung von Paketverlust, MobilKommunikation 2007 Vorles...
☐ PirVisualizer		PirVisualizer Description
☐ RangeCheck		Measurement of the LQI in relation to the node position
☐ Sequential		Sends sequential 32-bit numbers to the UART using the Cust...
☐ TimeProvider		Sends the passed time since 1.1.2000 (seconds as 32bit, ms ...
☐ Tradav Truck		Tradav application for monitoring the security of a truck
☐ Wireless Monitor		Monitors the activity on the wireless link

```
Done, got 16 chunks.
Loaded OTAP bin file: C:\Dokumente und Einstellungen\buschmann\Eigene
Dateien\Luebeck\coalesenses\test\developer\iAppsCoalesenses\BlinkApplication\bin\JN5139R1\iSenseDemoApp.bin
```

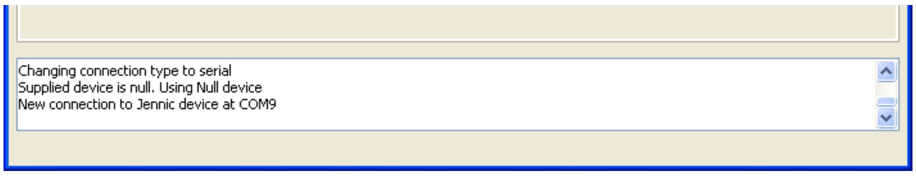
MAC read done.

Select the port the device is connected to in iShell. To do so, click on the preferences button (1).



Type in the number of a Node (5), a Block of Nodes (5-12), * for all Nodes or let the field empty for no nodes. Seperate serveral entries by a Comma

The "iShell Preferences" dialog will open. Choose "Use Serial", select the correct COM port, and click on "Ok".



```
Changing connection type to serial
Supplied device is null. Using Null device
New connection to Jennic device at COM9
```

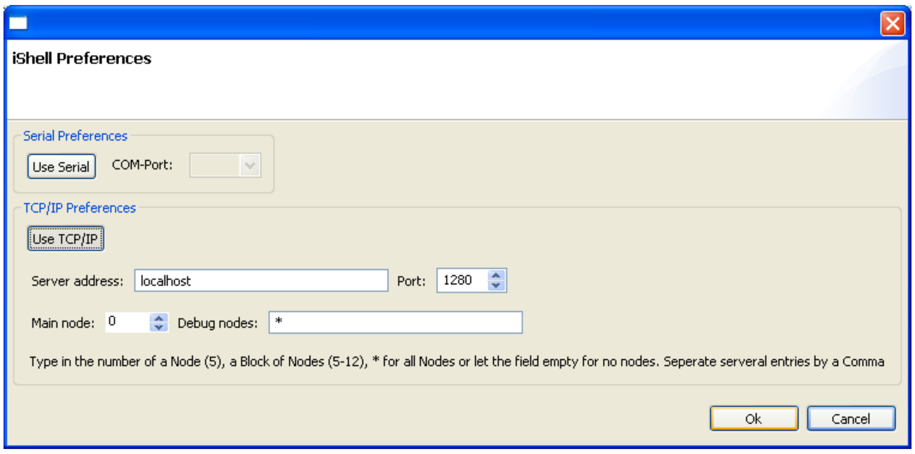
If the connection was successful, the logging pain at the bottom of the iShell window should say something like "New connection to Jennic device on COMX". However, be aware that this only indicates that a valid serial port was chosen.

3.2. Connecting to the Shawn simulator

Start the Shawn simulator by calling run_simulation.bat in the directory of your application. Be sure that the ShawnSocketTask is executed by adding the line

```
        shawn.runCommand("ShawnSocketTask",
                                 "socket_port=1280 socket_blocking=true");
```

to the configuration script configuration.jshawn directly before the simulation task. It will make Shawn wait for iShell to connect to it.



iShell Preferences

Serial Preferences

Use Serial COM-Port:

TCP/IP Preferences

Use TCP/IP

Server address: localhost Port: 1280

Main node: 0 Debug nodes: *

Type in the number of a Node (5), a Block of Nodes (5-12), * for all Nodes or let the field empty for no nodes. Seperate serveral entries by a Comma

Ok Cancel

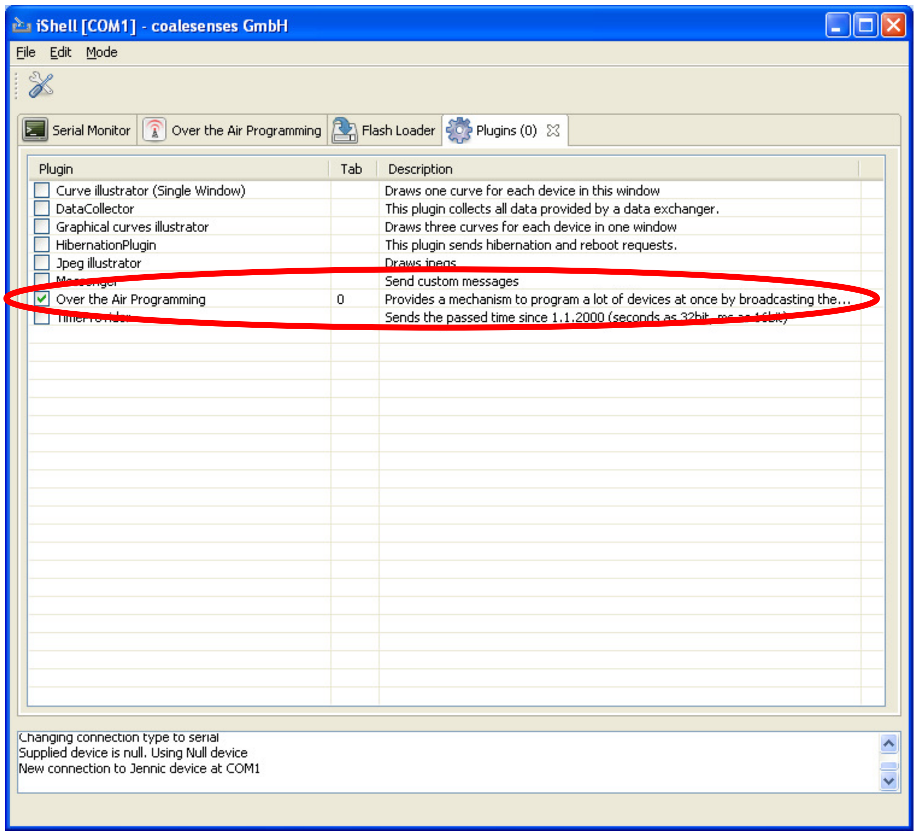
Now open the "iShell Preferences" dialog, choose "Use TCP/IP", enter the address of the computer where Shawn is waiting, and choose the same port as specified when starting the `ShawnSocketTask`. Then select the "Main node", that will appear to be "connected" to iShell.

You can specify additional "Debug nodes". If these nodes send messages through their `Os::debug(...)` or `Os::fatal(...)` methods, they will be forwarded to iShell as well, just as if they where also connected to iShell. You can specify the asterisk wild card (*) for all nodes, a node id (e.g. 3), intervals of nodes (e.g. 5-17) or a comma separated list of the latter two (e.g. 3, 5,7-12,27-29).

Then click on "Ok".

4. Activating Plugins

While the “Serial Monitor” plugin and the “Flash Loader” plugin are always active and cannot be closed, all other Plugins can be opened and closed on the “Plugins” tab.



Plugin	Tab	Description
☐ Curve illustrator (Single Window)		Draws one curve for each device in this window
☐ DataCollector		This plugin collects all data provided by a data exchanger.
☐ Graphical curves illustrator		Draws three curves for each device in one window
☐ HibernationPlugin		This plugin sends hibernation and reboot requests.
☐ Jpeg illustrator		Draws jpegs
☐ Messenger		Send custom messages
✓ Over the Air Programming	0	Provides a mechanism to program a lot of devices at once by broadcasting the…
☐ TimeProvider		Sends the passed time since 1.1.2000 (seconds as 32bit, ms as 16bit)

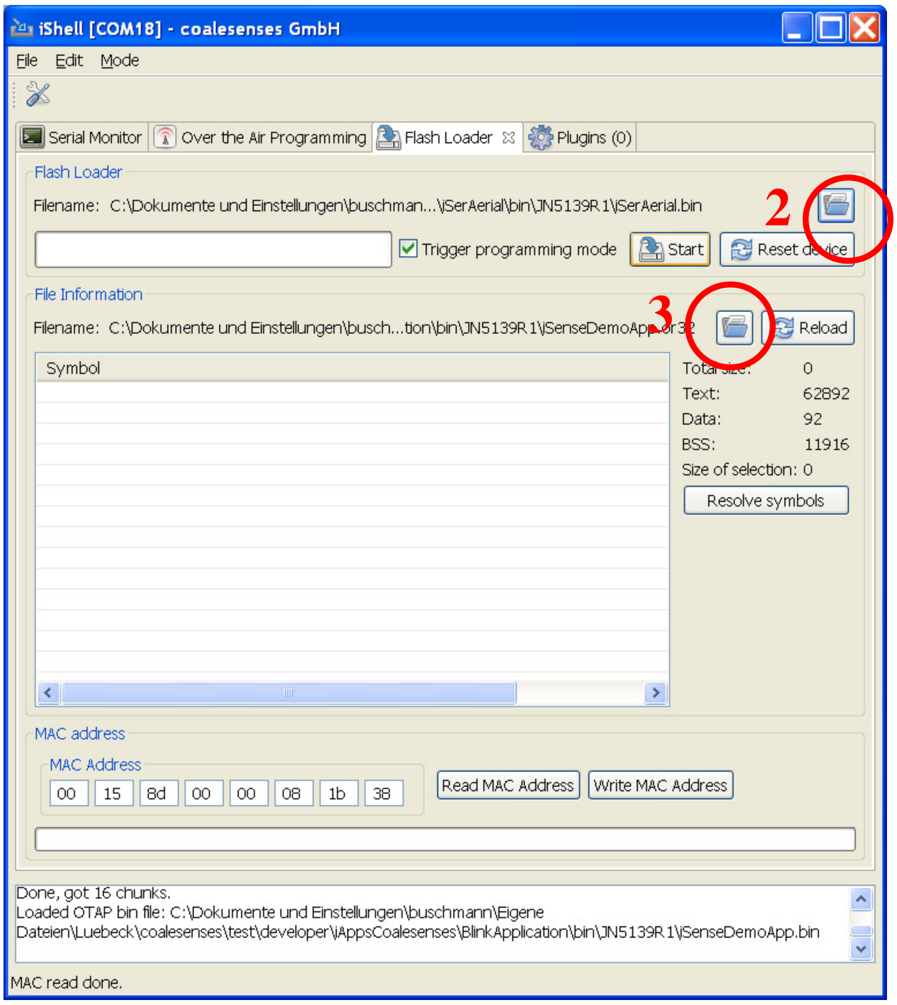
```
Changing connection type to serial
Supplied device is null. Using Null device
New connection to Jennic device at COM1
```

To activate a plugin, check the box on the very left, and the according tab will appear. You can close plugins by unchecking box again, or by clicking on the small red cross when the plugin is active.

5. Integrated plugins

5.1. Flash Loader

The „Flash Loader“ plugin provides functionality for flashing (i.e. programming) the connected device, analyzing the symbols in the program and reading and writing the device's MAC address.



Filename: C:\Dokumente und Einstellungen\buschman...\iSerAerial\bin\JN5139R1\iSerAerial.bin

Filename: C:\Dokumente und Einstellungen\busch...tion\bin\JN5139R1\iSenseDemoApp.or32

Total size:	0
Text:	62892
Data:	92
BSS:	11916
Size of selection:	0

00	15	8d	00	00	08	1b	38

```
Done, got 16 chunks.
Loaded OTAP bin file: C:\Dokumente und Einstellungen\buschmann\Eigene
Dateien\Luebeck\coalesenses\test\developer\iAppsCoalesenses\BlinkApplication\bin\JN5139R1\iSenseDemoApp.bin
```

MAC read done.

To flash a device, first of all choose the *.bin file to be flashed by clicking on the file select button (2) in the “Flash Loader” section. Make sure that the target chip version of the program and the chip version of the connected device match.

Then, flash the binary program to the iSense device connected to the PC. To do so, choose “Trigger programming mode”, click on “Start”, and wait for program download to complete. The chosen program is then by and by written to the connected device, the flash progress will be display in the status line of the iShell window.

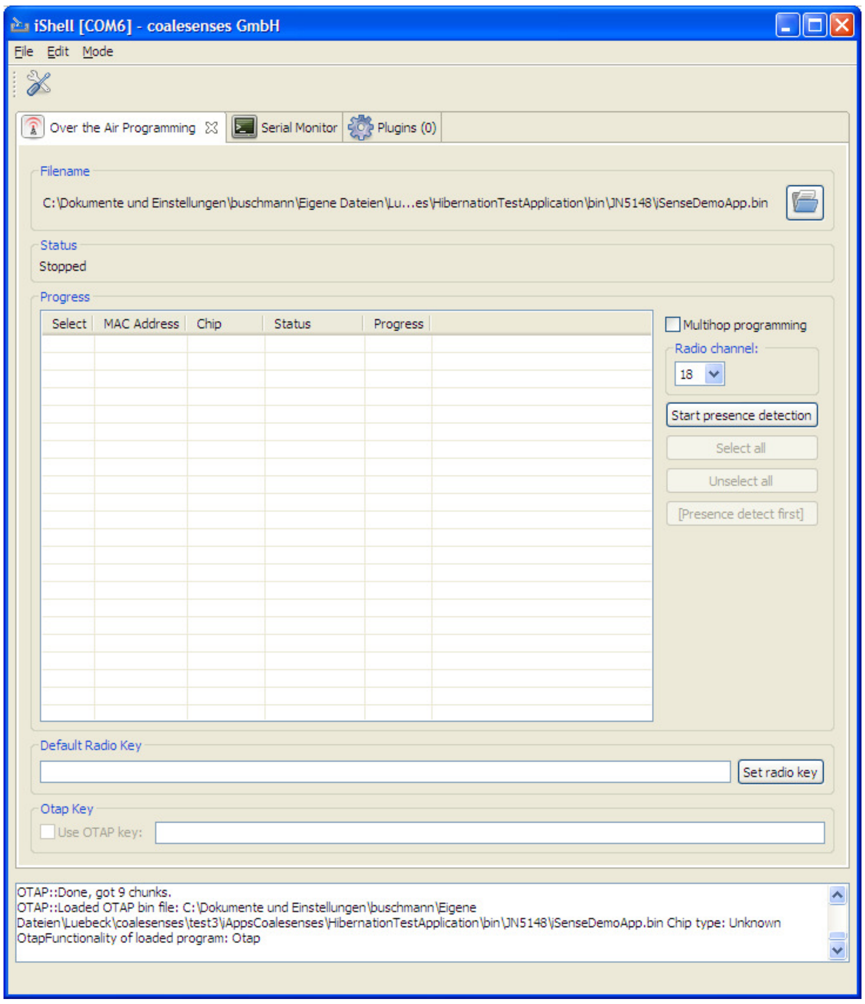
This process can be aborted before the whole program is transferred, but note that the MAC-address may be overwritten or set to zeros or ones, if the programming process is cancelled during the first second. After a successful programming the application is automatically started by a device reset.

In the “File Information”-section, *.or32-files can be loaded to have a look into the compiled program code. Chose a file by clicking on the file select button (3), and then click on “Resolve Symbols” The sizes and names of all existing symbols are shown. The symbols can be sorted by their sizes and names, as well as resolved for better reading.

In case the *.or32 file changed since last loading, you can update the information by clicking on “Reload”.

In the “MAC Address” section, the address fo the connected devices can be read or written. However, be aware that both actions involve restarting the device.

5.2. Over the air Programming



Filename: C:\Dokumente und Einstellungen\buschmann\Eigene Dateien\Lu...es\HibernationTestApplication\bin\JN5148\iSenseDemoApp.bin

Status: Stopped

```
OTAP::Done, got 9 chunks.
OTAP::Loaded OTAP bin file: C:\Dokumente und Einstellungen\buschmann\Eigene
Dateien\Luebeck\coalesenses\test3\iAppsCoalesenses\HibernationTestApplication\bin\JN5148\iSenseDemoApp.bin Chip type: Unknown
OtapFunctionality of loaded program: Otap
```

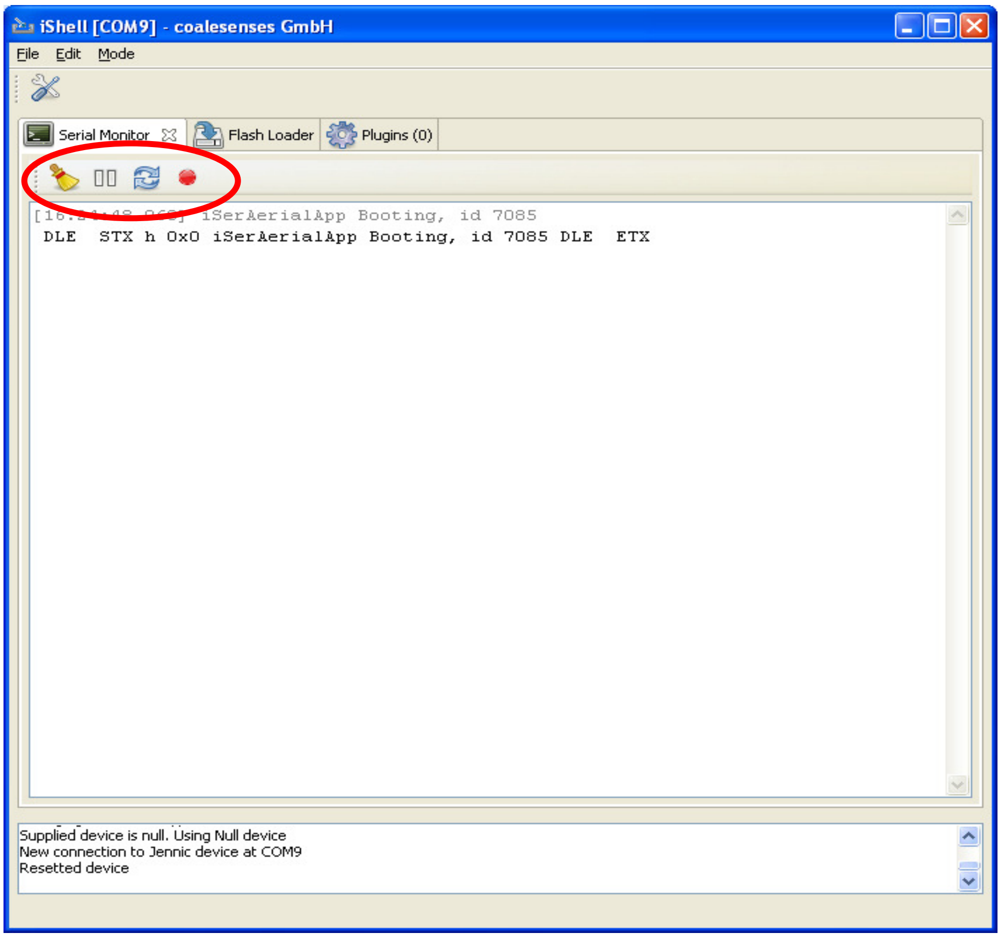
For details on using the over the air programming plugin, please refer to user guide "Flashing iSense Devices Wirelessly".

5.3. Serial Monitor

The serial monitor provides two modes, which can be set in the “Mode” menu.

The plain text mode displays all contents of text messages. Unknown characters are displayed as small boxes.

Operating the serial monitor in the packet mode means that only debug- and fatal-messages are displayed in grey and with red background respectively.



```
[16:21:48.968] iSerAerialApp Booting, id 7085
 DLE  STX h 0x0 iSerAerialApp Booting, id 7085 DLE  ETX
```

```
Supplied device is null. Using Null device
New connection to Jennic device at COM9
Resetted device
```

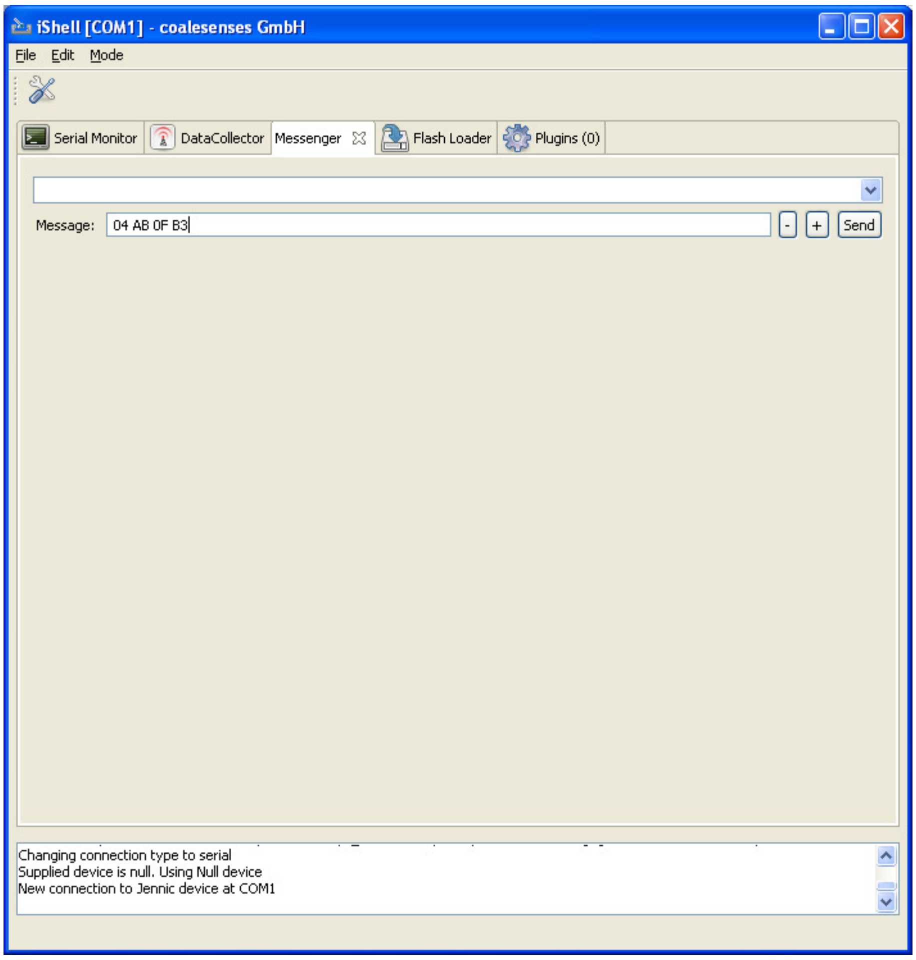
Four buttons permit managing the serial monitor as well as the iSense sensor node attached to iShell.

The leftmost button clears the window, the second pauses or restarts the output of messages to the text area, the third button resets the connected device, which means to restart the application on the device.

The rightmost starts/stops the recording of incoming messages to a file. On click, a file selection dialog opens. After a file name is specified, messages will be recorded to that file. Note the messages are recorded regardless of whether the text area output is paused or not. Clicking the button again stops the recording of messages.

5.4. Messenger

The “Messenger” plugin can send packets of arbitrary types and contents from iShell to the connected device.

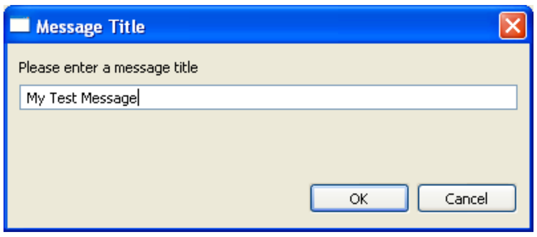


Message: 04 AB 0F B3

```
Changing connection type to serial
Supplied device is null. Using Null device
New connection to Jennic device at COM1
```

To send a message, type the message into the message field, and click on “Send”. All bytes of the message must be typed in hexadecimal encoding, separated by spaces. The first byte specifies the message type (i.e. the above example shows a message of type 4 that has a length of 3 bytes, the message byte values are 0xAB, 0x0F, and 0xB3). The plugin will check the message online for syntactical correctness, and the message field will turn red in case of errors.

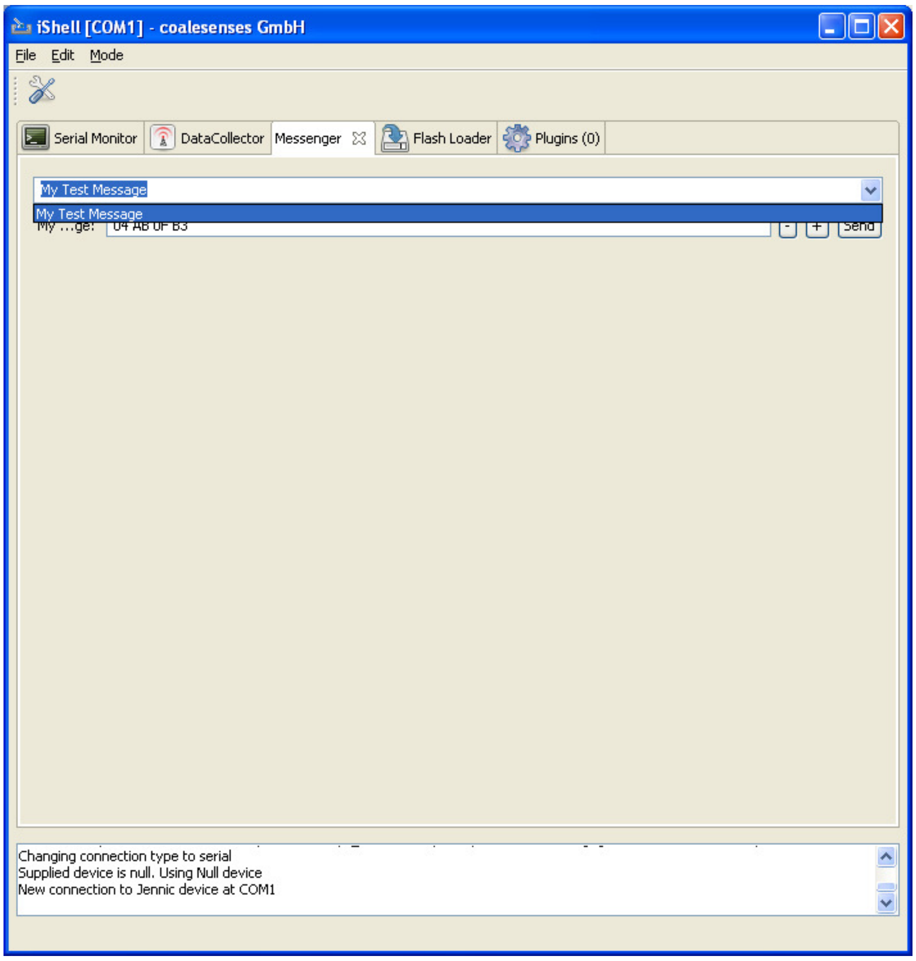
After a message was typed in, it can be saved to the history field at the top of the plugin by clicking on the “+” button.



Please enter a message title

My Test Message

In the opening message box, the message itself is offered as a default title. Specify a message title, and click on “Ok”. The message title will then appear in the history field.

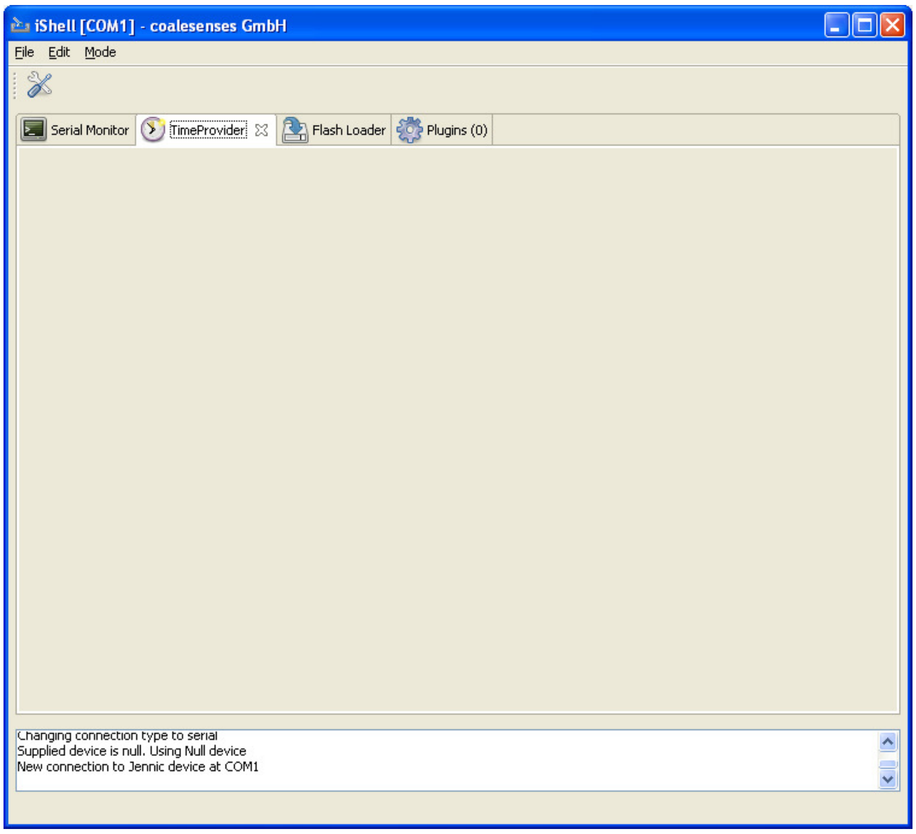


```
Changing connection type to serial
Supplied device is null. Using Null device
New connection to Jennic device at COM1
```

To delete a message from the history, select in from the history field, and then click on “-“.

5.5. Time provider

The “TimeProvider“ plugin can be used by an iSense sensor node that is connected to the PC to request the current time from iShell.



```
Changing connection type to serial
Supplied device is null. Using Null device
New connection to Jennic device at COM1
```

In order to do so, the calling iSense application instance must implement the `TimeHandler` Interface, and can then call `IShellInterpreter:: get_time_from_ishell(this)`. The `IShellInterpreter` then requests the time from iShell, and calls the `handle_time(Time t)` method of the `SensorHandler` that was passed to the `IShellInterpreter::get_time_from_ishell(SensorHandler* sh)` call.

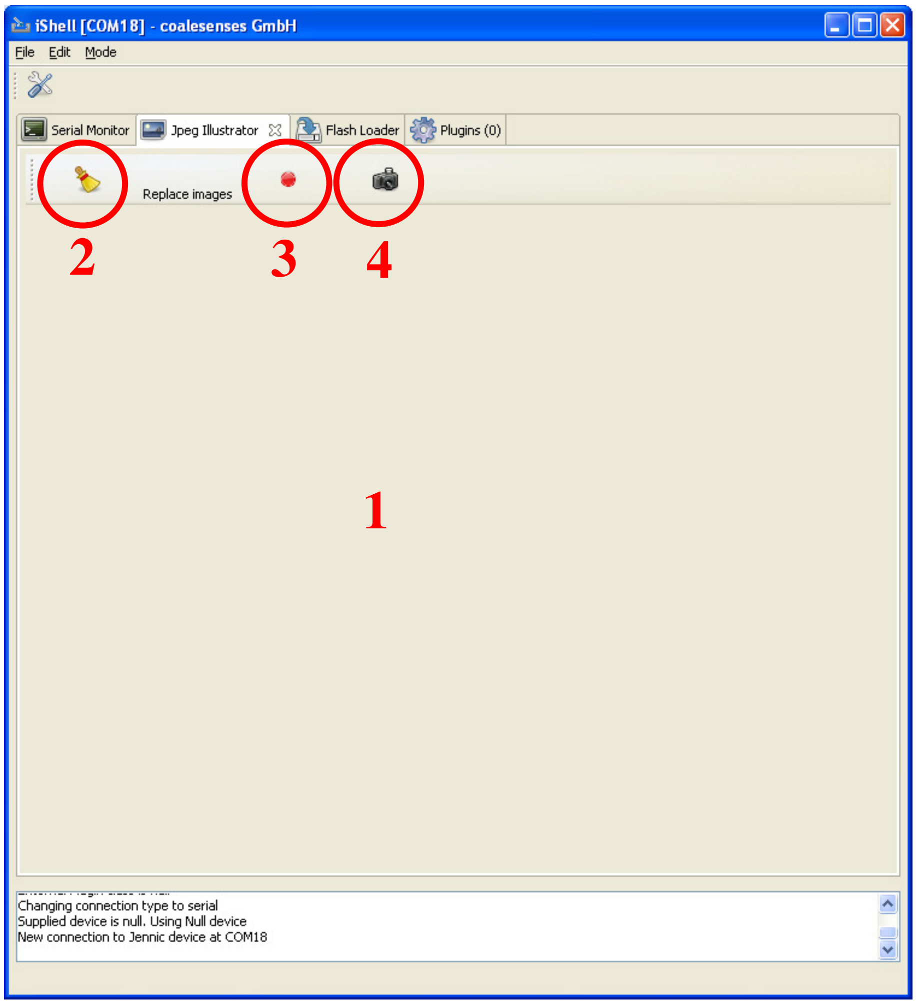
Note that the iShell time provisioning must be enabled to use this feature (true for the iSense standard firmware). Be sure that the requesting device does not sleep after the request, because it will not be able to receive the result then. For further information on using the time provider, consult the “iSense Time Provider” demo application.

Note that iShell returns the current time with regard to the UTC time zone, i.e. it ignores you local time zone as well as day light saving etc.

5.6. JPEG Viewer

The "JPEG Illustrator" plugin can display JPEG images sent to iShell from the connected iSense sensor node.

The connected node does so by calling `IShellInterpreter::send_jpeg(uint8* image, uint16 size)`, where `image` points the a JPEG image of `size` bytes. Note that the JPEG transmission feature must be enabled to activate this feature (not the case for the standard firmware)



```
Changing connection type to serial
Supplied device is null. Using Null device
New connection to Jennic device at COM18
```

Received images are depicted in the image area (1). If the user clicks on an image, a file selection dialog opens, and the image can be saved to the hard disk. By default, images are collected in the image area. However, by clicking on "Replace images", the image area replacement mode can be toggled between collecting and replacing images. In the latter case, each received image replaces the last one.

Clicking on the "Clear" button (2) clears the image area. Clicking the "Record" button (3) opens a directory selection dialog. All received images will then be saved to the selected directory until the "Record" button is clicked again to stop the recording. The file names of the saved images will be generated automatically from the current date and time data.

On click, the "Take image" button (4) sends a message to the connected iSense sensor node. The message is of type `Uart::MESSAGE_TYPE_CAMERA_APPLICATION`, and has a length of one, the only byte evaluates to 150. In case the attached iSense sensor node is flashed with the “Wireless Camera Application”, the associated camera will take a picture and transfer it to iShell.

5.7. Curve Illustrator (single window or multiple windows)

This plugin plots n curves for each device (in iShell or in a new window). The x-axis of the plot shows the time, the y-axis the values of the data series.

5.7.1. How to send data to this plugin

The Plugin listens to packets of type `isense::Uart::MESSAGE_TYPE_PLOT`, the following function encapsulates the interface in iSense and sends your data with the correct type.

```
IShellInterpreter::send_buffer_data(BufferData* bufferData, uint16
interval)
```

Sends a packet over the serial port to this iShell-plugin.

Parameters:

bufferData – The BufferData containing the time, dimension, count and the buffer with the data.

interval – The time interval in milliseconds between two values.

A BufferData has the following variables:

```
uint32 sec - Seconds of the time stamp of the first value
uint16 ms - Milliseconds of the time stamp of the first value
uint8 dim - Dimension: Number of curves
uint8 count - Number of entries per curve in the buffer
int16 buf[BUFFER_SIZE] - Values
```

Example:

```
BufferData buf_data;
buf_data.sec = 10;
buf_data.ms = 500;
buf_data.dim = 3;
buf_data.count = 20;
for (uint8 j = 0; j<buf_data.count; j++)
      buf_data.buf[j] = j;
IShellInterpreter isi = new IShellInterpreter(os_);
isi->send_buffer_data(&buf_data, 24);
```

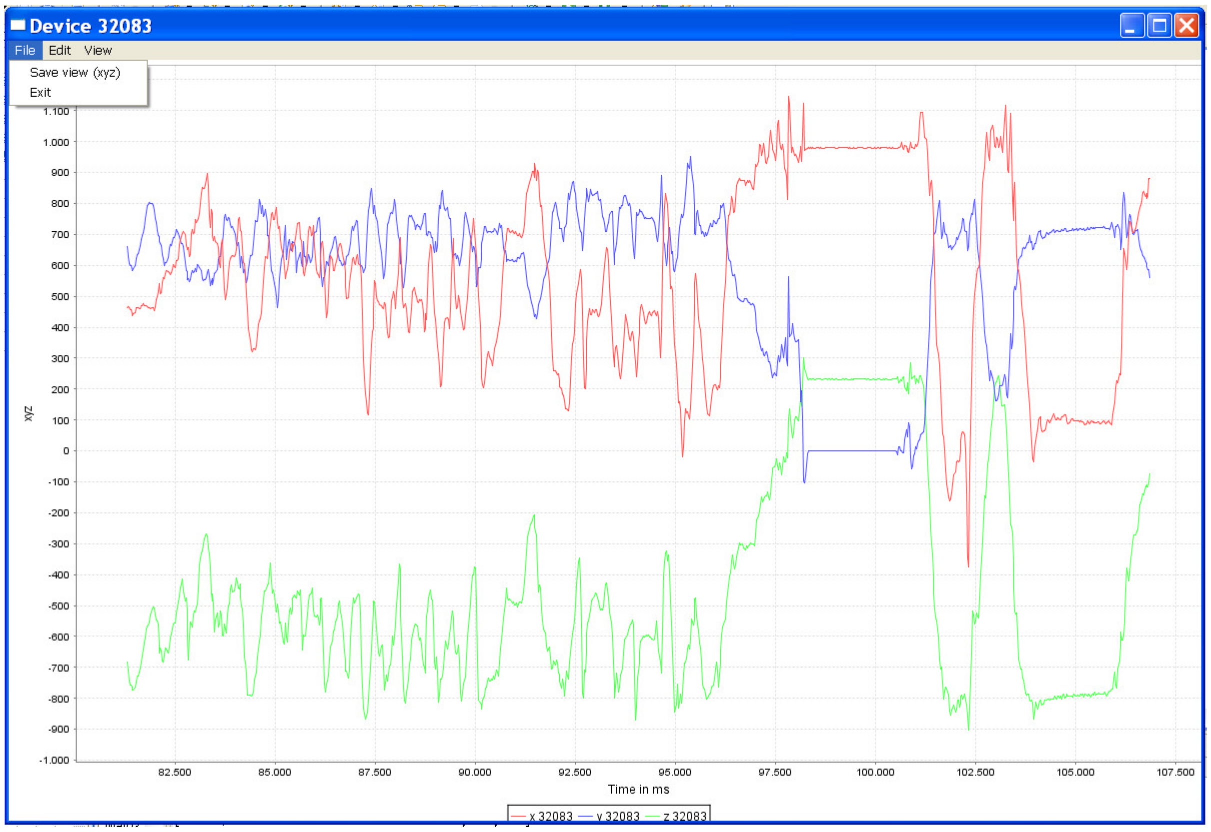
5.7.2. Data logging

When you open the plugin a random number R is generated to distinguish between log files of several iShell instances.

All data of incoming packets is written in a csv-file for each device with the name “iShelldata” + R + “_device” + DeviceID + “.csv”. In detail: for each value in the bufferData one entry (i.e. one line) is written in the csv-file containing

- the address of the device,
- the time in milliseconds and
- the value

separated by semicolons. This causes that the plugin logs (even if you do nothing) all data in a file, although not all data is plotted in the window anymore. When the plugin is closed all csv-files are deleted from the file system.



5.7.3. *Functionality of the buttons*

To control the data logging there are three buttons with the following functions:

- Clear: Deletes all data in view AND in csv-file.
- Save all data: Generates an excel-file (with the same name of the csv-file except the ending as default). All data which is actually in the csv-file is converted to excel data and written into the excel-file.
- Save data of view: Generates an excel-file with the name “iShelldata” + R + “_device” + DeviceID + “_view.xls”. and writes all data which can be seen in the window into the file.

5.8. Using the Data Collector

The “Data Collector” plugin is an iShell plugin which collects all data provided by the iSense devices and stores this data into files for further evaluation purposes. The Data Collector allows you to store all data in one file and to store the data individually for each node in a separate file.

Note that the device connected to iShell must run the “Data collector” application, and that the nodes that are to be queried for data must make use of the `DataExchanger` functionality.

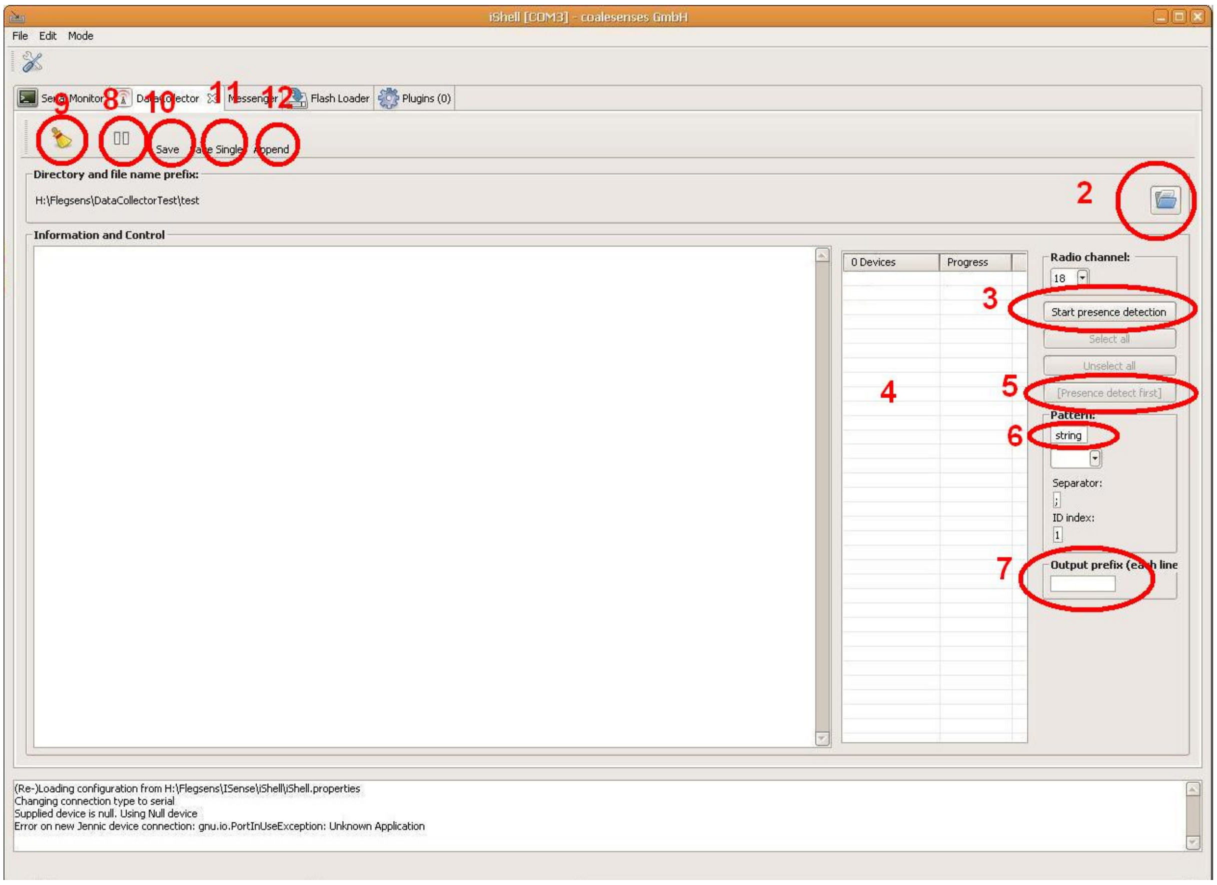
5.8.1. Selection of Devices

Click on the file selector button (2), and choose the filename prefix for all output files.

Now choose the channel on which the devices you want to collect data from communicate. By default devices use channel 18.

To detect present devices, click on “Start presence detection” (3). Nearby devices will then show up in the device list (4) if their program supports the DataExchanger feature.

Select the nodes of which you want to collect the data (by checking the corresponding checkbox left of the device address in the device list), or select all detected nodes with a click on “Select All”. Devices can be unselected by un-checking the corresponding checkbox or with a click on “Unselect All”. You can stop the presence detection by clicking on “Stop presence detection”.



Directory and file name prefix:
H:\Flegsens\DataCollectorTest\test

Information and Control

Radio channel: 18
Pattern: string
Separator: ;
ID index: 1

```
(Re-)Loading configuration from H:\Flegsens\ISense\iShell\iShell.properties
Changing connection type to serial
Supplied device is null. Using Null device
Error on new Jennic device connection: gnu.io.PortInUseException: Unknown Application
```

5.8.2. Parsing of Packets

Additionally, you have to configure how to parse the received packets. Packets can be parsed as numbers or as a character string. In the text field “Pattern” you enter the parsing pattern. This string is either string or a concatenation of the following symbols separated by blank:

Symbol	*Semantic*
u8	Read next byte as unsigned integer
u16	Read next two bytes as unsigned integer
u32	Read next four bytes as unsigned integer
u64	Read next eight bytes as unsigned integer
i8	Read next byte as signed integer
i16	Read next two bytes as signed integer
i32	Read next four bytes as signed integer
i64	Read next eight bytes as signed integer
u8*	Read next byte as unsigned counter (for counters, see the example below)
u16*	Read next two bytes as unsigned counter
u32*	Read next four bytes as unsigned counter

The counters and a subsequent substring tell the parser to expect *n* values of the type defined by the subsequent substring where *n* is the value of the counter. For example, the format string

```
u16 u8* u16
```

would result in parsing the first two bytes as unsigned integer, the third byte as a one-byte-counter and a number of two-byte-unsigned integers where as this number depends on the value of the previously read counter.

Instead of using these symbols, you can also parse the packets as separated character strings. For, this, you just have to enter “string” into the pattern text field.

All the symbols for number parsing and the symbol “string” can be selected in the combo in the “Pattern” group.

Now the plugin knows how to parse the received packets. In the text field “ID index” the user can select the position in the packets that contains the device address. Indexing starts at position 0 and is interpreted as value position. The plugin needs to know where to find the node address to assign the packets to the individual files in case you want to store the data in separate files for each device.

For example, if your devices send packets that contain a type (u16), the id (u16), and a data value (u32), you would have to enter 1 as ID index.

In case you want to parse the packets as strings, you have to additionally enter the separator within the packets. Then the string is split according to the separator. The optional ID index works the same way.

For example, if your devices send packets that contain three values; a type (2 characters), the device id (6 characters) and a data value (5 characters); each of these values separated by semicolon, you have to enter 1 as ID index.

You can also enter -1 as ID index, then the data of the packets will not be assigned to any node ID.

If you enter no separator, the packets will not be split, and will hence be parsed as one value.

5.8.3. Data Exchange

After the selection of the nodes and the configuration of the parsing, you can start to collect the data with a click on “Start Exchange” (5). The exchange process will start and the data of the selected nodes will be requested sequentially.

The “Progress” column in the device list will inform you about the state of the different devices. If a device has sent all its data the progress entry for this device will we set to “Done”, if the exchange stopped ahead of time it is set to “Failed”.

You can stop the exchange process by clicking on “Stop Exchange” (the former “Start Exchange” button, 5).

During the data exchange, the received data is displayed in the text window. With a click on “Pause” (8) this output is paused, but can be resumed with another click on “Resume”. However, the Data Collector stores all the collected data regardless of whether the output is paused or not.

With a click on “Clear” (9), the text window and all data collected so far is cleared.

5.8.4. Data Storage

With a click on “Save” (10) all data is written into one CSV-file (`prefix + “.csv”`) and an Excel file (`prefix + .xls`).

With a click in “Save single” (11) you can write the data for each node into individual files.

The output file names follow the pattern `prefix+”_”+id +”.csv”` and `prefix+”_”+id+”.xls”`.

You can also set the set option “Append” (12). With this option set all data will be appended to these output files if they already exist, otherwise they will be created. With another click you can reset the option to the default behaviour, which is overwriting the output files.

Additionally, you can enter an output prefix (7). This prefix is put in front of each line in the output files. It has to be terminated with a semicolon because of the file format of .csv-files. For example, with option “Append” set, you can distinguish different runs of data collection within one file.

5.9. Hibernation

iSense devices that run a firmware that has the “Hibernation” feature enabled can enter a special mode of operation, the so called hibernation.

When a device enters hibernation, it stops the execution of the application code and goes into the sleep mode. A hibernating device wakes up once a minute, reads out battery state information (if the “Solar Power Module” feature or the “Energy Module” feature are enabled in the firmware and one of the according modules is connected to the node), broadcasts a hibernation radio packet (with the battery information if available) via the radio, stays awake for 300ms, and goes back to sleep.

Hibernation is intended e.g. to store iSense devices that are switched on but that are not used currently at a minimum of current consumption while not having to switch them off mechanically (e.g. in order to avoid opening their housing to access the switch).

With the duty cycle of about 0.5 percent, the current consumption of hibernating devices will be around 300µA (depending on the connected Power Module).

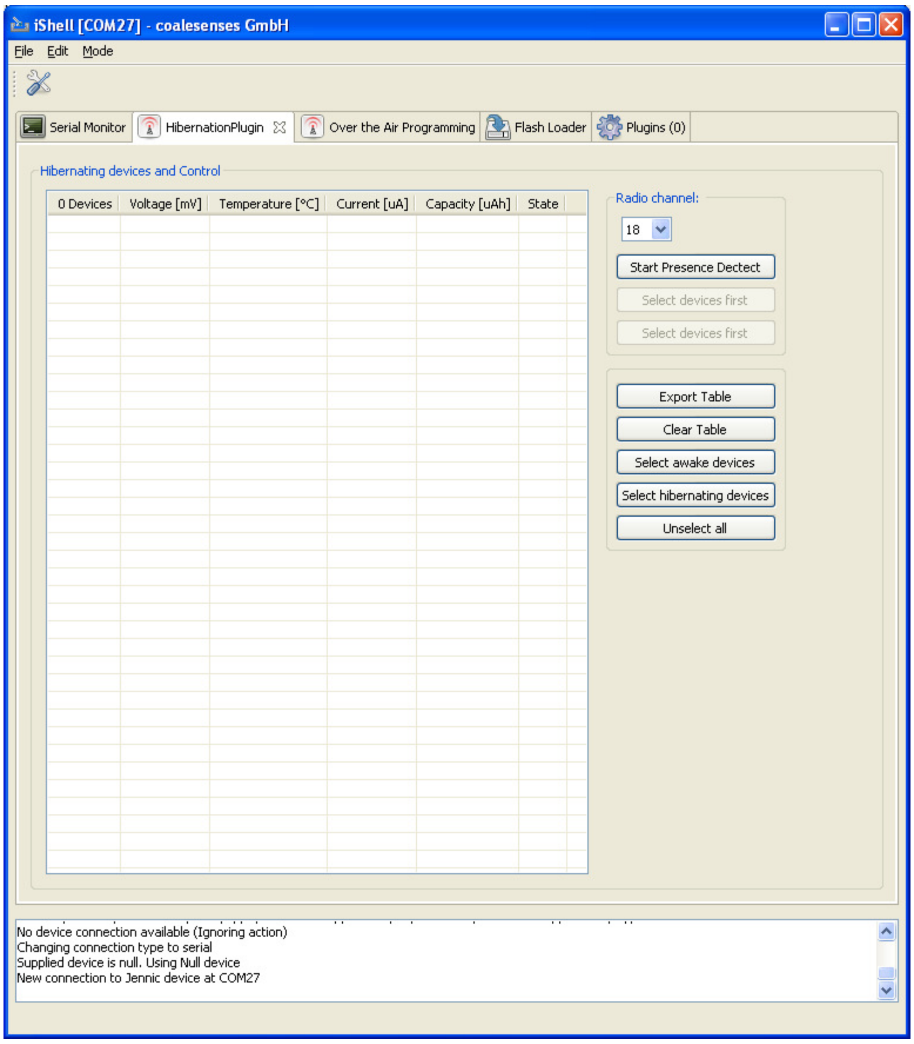
The iShell “Hibernation” plugin can be used to

- detect awake devices that support hibernation,
- send them into hibernation,
- display battery state information emitted by hibernating devices and
- wake them up again.

For its operation, the “Hibernation” plugin needs an iSense device connected to it that either

1. runs the iSerAerial application [1] or
2. runs a firmware that has the “iSerAerial” feature enabled (for details on how to generate customized firmware revisions, see [2]), and that runs an application that instantiated the `isense::IShellInterpreter` and enabled the iSerAerial functionality by calling `isense::IShellInterpreter::enable_seraerial()`.

To use the “Hibernation” plugin, set the radio channel of the connected device first.

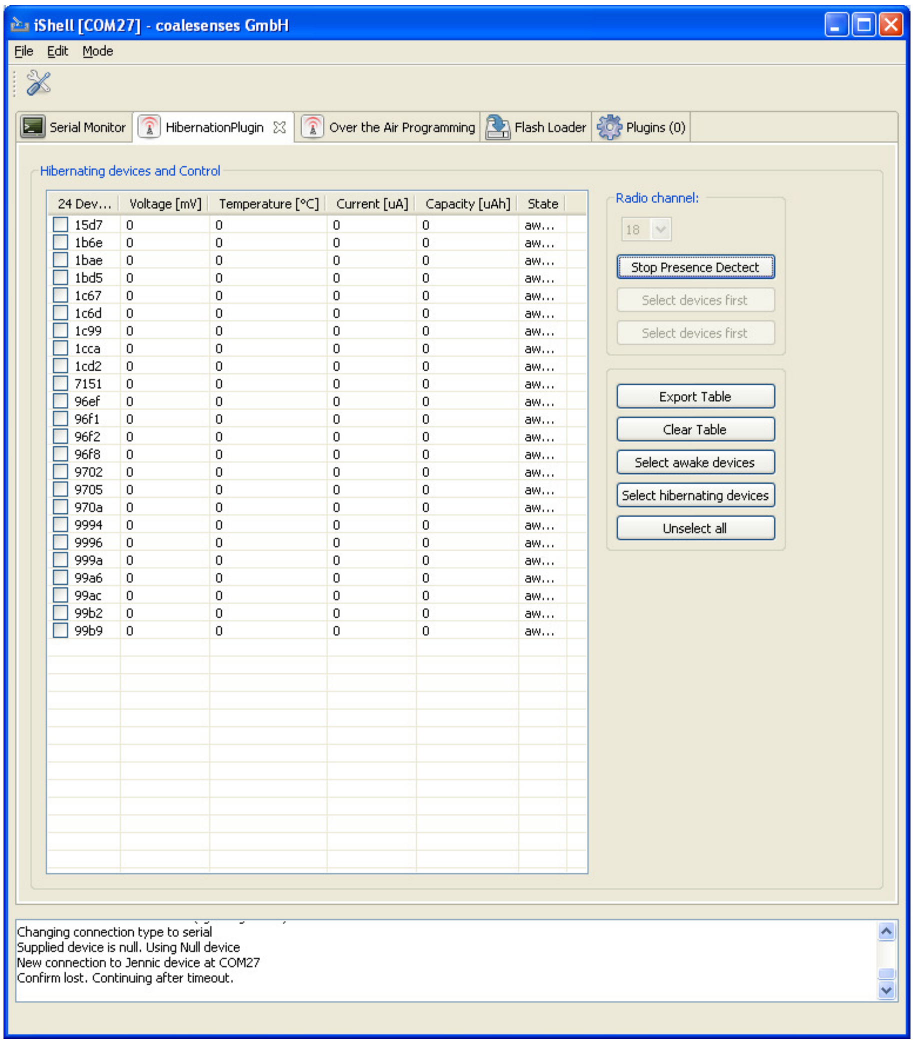


```
No device connection available (Ignoring action)
Changing connection type to serial
Supplied device is null. Using Null device
New connection to Jennic device at COM27
```

Note that setting the channel of the connected device is only supported by the iSerAerial application, as it is assumed that a device running a different application with the iSerAerial firmware feature enabled is already communication on the correct radio channel.

To detect awake devices in the one-hop neighborhood that support hibernation, click on “Start Presence Detect”. Note that hibernating devices in the one-hop neighborhood are detected automatically once a radio broadcast is received from them. Hibernating devices will vanish again from list after they have not been heard again for 4 minutes and 30 seconds.

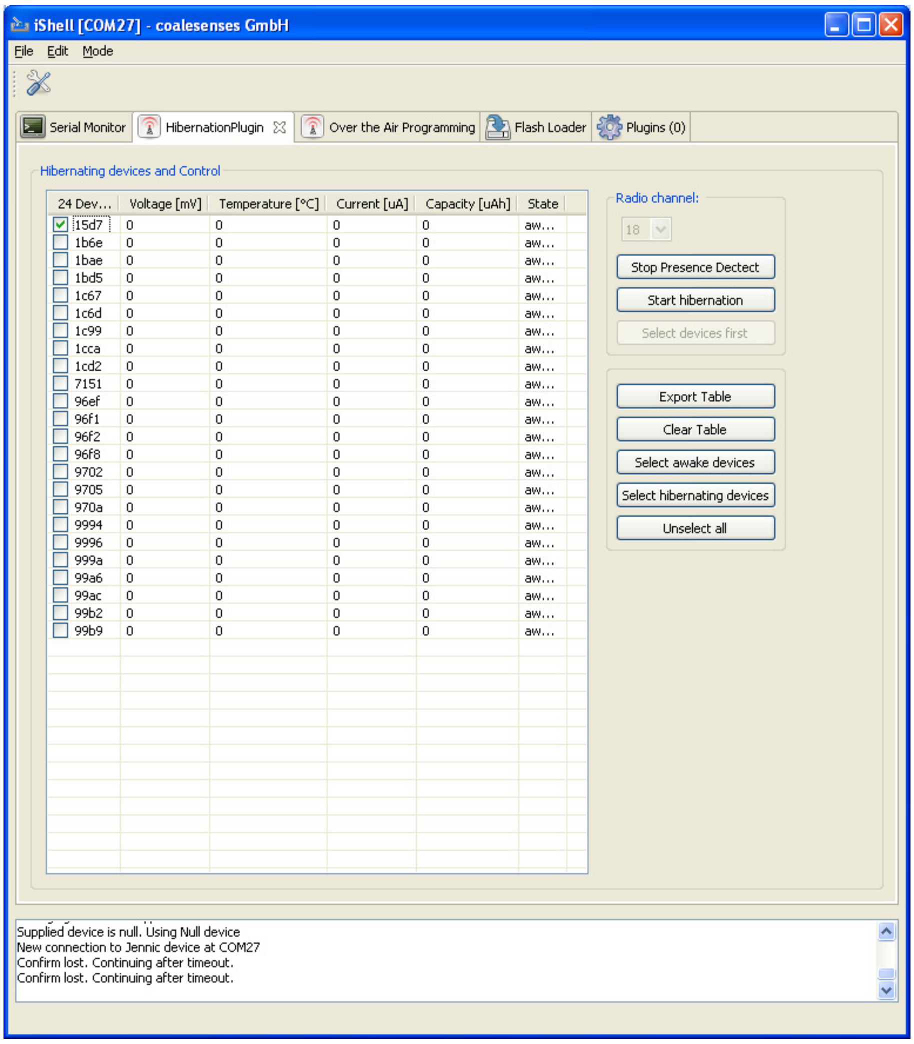
On click, the button text will change to “Stop Presence Detect”, and awake devices in the one-hop neighborhood will appear in the table as “awake” in the “State” column.



24 Dev...	Voltage [mV]	Temperature [°C]	Current [uA]	Capacity [uAh]	State
☐ 15d7	0	0	0	0	aw...
☐ 1b6e	0	0	0	0	aw...
☐ 1bae	0	0	0	0	aw...
☐ 1bd5	0	0	0	0	aw...
☐ 1c67	0	0	0	0	aw...
☐ 1c6d	0	0	0	0	aw...
☐ 1c99	0	0	0	0	aw...
☐ 1cca	0	0	0	0	aw...
☐ 1cd2	0	0	0	0	aw...
☐ 7151	0	0	0	0	aw...
☐ 96ef	0	0	0	0	aw...
☐ 96f1	0	0	0	0	aw...
☐ 96f2	0	0	0	0	aw...
☐ 96f8	0	0	0	0	aw...
☐ 9702	0	0	0	0	aw...
☐ 9705	0	0	0	0	aw...
☐ 970a	0	0	0	0	aw...
☐ 9994	0	0	0	0	aw...
☐ 9996	0	0	0	0	aw...
☐ 999a	0	0	0	0	aw...
☐ 99a6	0	0	0	0	aw...
☐ 99ac	0	0	0	0	aw...
☐ 99b2	0	0	0	0	aw...
☐ 99b9	0	0	0	0	aw...

```
Changing connection type to serial
Supplied device is null. Using Null device
New connection to Jennic device at COM27
Confirm lost. Continuing after timeout.
```

Select the devices you want to send to hibernation by checking the checkbox in the leftmost column. Once one or more awake devices are selected, the “Start hibernation” button will be enabled. To hibernate the selected devices, click it.



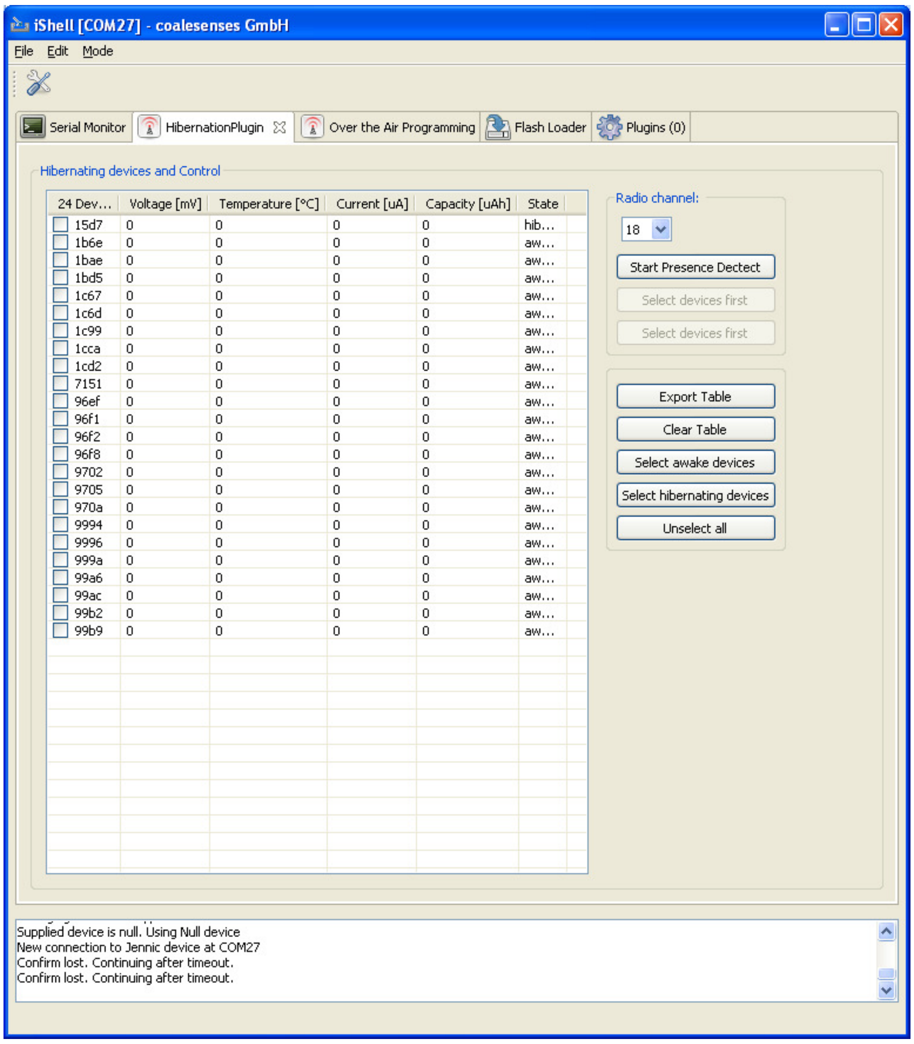
24 Dev...	Voltage [mV]	Temperature [°C]	Current [uA]	Capacity [uAh]	State
✓ 15d7	0	0	0	0	aw...
1b6e	0	0	0	0	aw...
1bae	0	0	0	0	aw...
1bd5	0	0	0	0	aw...
1c67	0	0	0	0	aw...
1c6d	0	0	0	0	aw...
1c99	0	0	0	0	aw...
1cca	0	0	0	0	aw...
1cd2	0	0	0	0	aw...
7151	0	0	0	0	aw...
96ef	0	0	0	0	aw...
96f1	0	0	0	0	aw...
96f2	0	0	0	0	aw...
96f8	0	0	0	0	aw...
9702	0	0	0	0	aw...
9705	0	0	0	0	aw...
970a	0	0	0	0	aw...
9994	0	0	0	0	aw...
9996	0	0	0	0	aw...
999a	0	0	0	0	aw...
99a6	0	0	0	0	aw...
99ac	0	0	0	0	aw...
99b2	0	0	0	0	aw...
99b9	0	0	0	0	aw...

```
Supplied device is null. Using Null device
New connection to Jennic device at COM27
Confirm lost. Continuing after timeout.
Confirm lost. Continuing after timeout.
```

As a result, the devices will enter hibernation. This is also visualized in the table as indication their state as “hibernating”. While at first the columns “Voltage”, “Temperature”, “Current” and ”Capacity” will display a 0 to indicate that no current values are available, the battery information will be available over time. With regard to the current, negative values indicate that the battery is depleted due to the nodes current consumption, positive values indicate that the battery is charged (e.g. due to the iSense Solar Power Harvesting System).

If all four values contain 255, this indicates that no battery information is available (e.g. because the according node is not equipped with an iSense Solar Power Module, Rechargeable Battery Module or 1/2 AA Battery Module, but only with a AA Battery Holder that does not feature a battery monitor).

To wakeup hibernating devices, select them by checking the checkbox in the leftmost column of the according table rows.



Radio channel: 18

24 Dev...	Voltage [mV]	Temperature [°C]	Current [uA]	Capacity [uAh]	State
15d7	0	0	0	0	hib...
1b6e	0	0	0	0	aw...
1bae	0	0	0	0	aw...
1bd5	0	0	0	0	aw...
1c67	0	0	0	0	aw...
1c6d	0	0	0	0	aw...
1c99	0	0	0	0	aw...
1cca	0	0	0	0	aw...
1cd2	0	0	0	0	aw...
7151	0	0	0	0	aw...
96ef	0	0	0	0	aw...
96f1	0	0	0	0	aw...
96f2	0	0	0	0	aw...
96f8	0	0	0	0	aw...
9702	0	0	0	0	aw...
9705	0	0	0	0	aw...
970a	0	0	0	0	aw...
9994	0	0	0	0	aw...
9996	0	0	0	0	aw...
999a	0	0	0	0	aw...
99a6	0	0	0	0	aw...
99ac	0	0	0	0	aw...
99b2	0	0	0	0	aw...
99b9	0	0	0	0	aw...

```
Supplied device is null. Using Null device
New connection to Jennic device at COM27
Confirm lost. Continuing after timeout.
Confirm lost. Continuing after timeout.
```

Once one or more hibernating devices are selected, the “Start reanimation” button will be enabled. To wake the selected devices, click it.

The button text will change to “Stop Hibernation”. Note that hibernating devices cannot be woken instantly. Instead, the “Hibernation” plugin waits for the radio broadcast of the hibernating devices, and wakes them in the subsequent wake phase by sending them an according radio packet.

Once a device has been woken, it appears in the table with the state “awake” again. After all selected devices have been woken, the “Stop reanimation” button is disabled again.

For exporting the table contents into a file with comma separated values at any time of the process, click on “Export Table”.

6. Developing your own plugins

1.) Get the iShell development jar-file (e.g., ishell-win32.jar) from the “Development Environment” subsection in the “Download” section of the coalesenses web site.

2.) Write a plugin (either plain or seraerial).

a. The plain plug-in will receive all packets in the same format as they have been sent from iSense using the Uart::write_packet method. This is primarily intended for interacting with the node that is attached to the serial interface.

Extend the abstract "ishell.plugins.Plugin" class and implement the abstract methods. The init method returns an array of integer values that represent the packet-types that this plug-in will receive (the same value as used when invoking iSense's Uart::write_packet() method). HandlePacket is called whenever a packet has been received via the serial link. The interface is as follows:

```
public class ExamplePlugin extends ishell.plugins.Plugin
{
        public String getName() { ... }
        public String getDescription()  { ... }
        public int[] init()  { ... }
        public void handlePacket(Packet p)  { ... }
        public void shutdown()  { ... }
}
```

b. The SerAerial plug-in type is the counterpart of the isense SerAerial functionality. Here, all packets that are received on the wireless interface are transmitted inside a special SerAerial packet to the SerAerial plug-ins. SerAerial plug-ins can furthermore transmit packets via the serial link that will then be transmitted. Extend the abstract SerAerialPlugin class and implement the abstract methods. The *seraerialHandlePacket* method is called whenever a packet has been received on the wireless interface.

```
public class ExampleSeraerialPlugin extends ishell.plugins.seraerial.SerAerialPlugin
{
        public abstract void seraerialInit();
        public abstract void seraerialHandlePacket(SerAerialPacket p);
        public abstract void seraerialShutdown();
}
```

3.) Compile the plug-in

```
javac -cp .;ishell-dev-win32.jar ExamplePlugin.java ExampleSeraerialPlugin.java
```

4.) Tell iShell to load the plugin. Put the following into your ishell.properties file:

```
plugin_classpath = c:/path/to/your/plugin/;c:/path/to/your/required.jar
plugin_classes = ExamplePlugin ExampleSeraerialPlugin
```

6.1.The iShell plugin API

This is the base class for all iShell plugins. It provides features most plugins have in common (toolbar, tab item, plugin container widget). It also has some abstract methods (like handlePacket) which you will have to override for your plugin to work.

6.1.1. sendPacket

public final void **sendPacket**(int type,

byte[] b)

Sends a packet over the serial port of the plugin's device monitor and hence to a connected device.

Parameters:

type - The packet type.

b - The actual packet as a byte array.

6.1.2. getMainTextView

public final org.eclipse.swt.custom.StyledText **getMainTextView**()

Returns the text view of the plugin's monitor view.

Returns:

Text view of the monitor view.

6.1.3. setTabView

public final void **setTabView**(TabView tabView)

Sets the tab view this plugin is displayed in. This method is only used internally and should not be called by plugin developers.

Parameters:

tabView - New tab view of the plugin.

6.1.4. getTabItem

public final org.eclipse.swt.custom.CTabItem **getTabItem**(java.lang.String name)

Creates and returns the tab item for this plugin. If you want your plugin to be displayed in iShell calling this function is the way to go. If you want your plugin to be found more easily in the tab view, you can add an icon from the IconTheme to it.

Parameters:

name - Plugin name to be displayed in the tab item.

Returns:

The tab item for this plugin.

6.1.5. removeTabItem

public final void **removeTabItem**()

Removes the tab item from the tab view in the iShell window. This is only used internally and should not be called by plugin developers.

6.1.6. getTabContainer

public org.eclipse.swt.widgets.Composite **getTabContainer**(boolean withCoolBar)

Creates and returns the main container for your plugin content. It optionally creates a CoolBar and a ToolBar for you if you need it.

Parameters:

withCoolBar - Whether your plugin needs a CoolBar/ToolBar.

Returns:

The main container for your plugin content.

6.1.7. addToolBarAction

public void **addToolBarAction**(org.eclipse.jface.action.IAction action)

Adds an action to the toolbar. This is the typical way of creating a toolbar item: First, create an action with a title and an icon and then add it to the plugin toolbar by calling this method.

Parameters:

action - Action to be added to the toolbar.

6.1.8. isVisible

public boolean **isVisible**()

Returns whether the plugin is currently visible in the iShell window or not.

Returns:

True if the plugin is visible, false otherwise.

6.1.9. pause

public final void **pause**()

Pauses the plugin. If a plugin is paused, it is no longer asked to handle incoming packets. This can be used to stop monitoring incoming data for a while in order to analyze the data your plugin has received so far.

6.1.10. isPaused

public final boolean **isPaused**()

Returns whether the plugin is paused or not.

Returns:

True if the plugin is paused, false otherwise.

6.1.11. resume

public final void **resume**()

Starts handling incoming packets again. This is the opposite of the pause() call.

6.1.12. notifyTimeout

public final void **notifyTimeout**()

6.1.13. getName

public abstract java.lang.String **getName**()

Returns the unique name of the plugin. This name is displayed in the plugin manager view in iShell. Please note that this name has to be unique for the compareTo() method to work properly.

Returns:

Display name of the plugin.

6.1.14. getDescription

public abstract java.lang.String **getDescription**()

Returns the description for this plugin. This is used in the plugin manager view in iShell.

Returns:

Description for your plugin.

6.1.15. init

public abstract int[] **init**()

This method is called when the plugin is initialized. It is usually used to construct the user interface for the plugin and to decided which packet types your plugin will handle.

Returns:

Array of packet types to be handled by your plugin.

6.1.16. handlePacket

public abstract void **handlePacket**(Packet p)

If your plugin is registered itself for handling certain packet types in the init() method handlePacket() will be called whenever a packet to be handled by your plugin is received over the serial port.

Parameters:

p - Received packet to be handled by your plugin.

6.1.17. *shutdown*

public abstract void **shutdown**()

This method is called when a plugin will no longer be used by iShell. Usually this happens when you close the plugin either in the tab view or in the plugin manager view. Override it to clean up whatever you need before your plugin is destroyed.

6.1.18. *compareTo*

public int **compareTo**(java.lang.Object obj)

Compares two plugins. This happens by comparing the name returned by getName().

Specified by:

compareTo in interface java.lang.Comparable

Parameters:

obj - Another plugin.

Returns:

0 if both plugins are equal, -1 or 1 if their names differ.

7. Command line options

No particular command line options are required for typical operation. However, the following options are available:

Short option name	*Long option name*	*Description*
-i	--volatile-config	If present, the configuration changes will not be saved to disk
-c	--config-file	The configuration file to use
-p	--serial-port	serial port to use (e.g. COM1 on Windows, /dev/ttyS0 on *nix)

8. Configuration file options

The configuration file consists of simple name=value lines. The #-character starts a comment and the rest of the line is ignored. The configuration file follows the rules of the java.util.Properties file format (http://java.sun.com/j2se/1.5/docs/api/java/util/Properties.html).

The following options are recognized:

Name	*Default value*	*Description*
serial_port	COM1	The serial port to use (e.g. COM1 on Windows, /dev/ttyS0 on *nix)
read_baudrate	115200	The baud rate to use when talking to an iSense node. Do not change this value unless you know what you are doing.
flash_baudrate	38400	The baud rate to use when flashing an iSense node. Do not change this value unless you know what you are doing.
message_mode	1	Use 0 for packet based, and 1 for plain mode
max_retries	30	Used to parameterize the "Over the Air programming". Number of retries until a device gives up to re-request missing program chunks.
timeout_multiplier	100	Used to parameterize the "Over the Air programming". Sets to number of milliseconds that is used as a factor when calculating timeouts.
window_height	600	Height of the GUI window in pixels.
window_width	800	Width of the GUI window in pixels.
plugin_classpath	plugins/	The additional classpath to use when loading plug-ins. This setting follows the format of the standard Java classpath specification and is operating system dependent.
plugin_classes		Space-separated list of Plug-in class names that should be loaded.
bin_file		The filename used for flashing the device over the serial port.
otap_file		The filename used for flashing devices over the air.

An exemplary configuration may look like as follows:

```
#coalesenses iShell settings
#Fri May 11 13:02:28 CEST 2007
bin_file=C\:\\Programme\\cygwin\\jennic\\developer\\iApps\\CountToRadio\\CountToRadio.bin
window_width=1208
otap_file=C\:\\Programme\\cygwin\\jennic\\developer\\iApps\\CountToRadio\\CountToRadio.bin
plugin_classes=testplugin
window_y=18
window_x=-4
or32_file=C\:\\Programme\\cygwin\\jennic\\developer\\iApps\\PaketverlustTest\\bin\\PaketverlustTest.or32
serial_port=COM1
window_maximized=true
timeout_multiplier=12
message_mode=0
plugin_classpath=plugins
max_retries=90
window_height=1586
```

9. References

[1] coalesenses Development Environment Setup User Guide, online available at http://www.coalesenses.com/index.php?page=development-environent

[2] iSerAerial Application Binary, online available at http://www.coalesenses.com/index.php?page=isense-applications

[3] coalesenses Writing iSense Applications User Guide, online available at http://www.coalesenses.com/index.php?page=isense-applications

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

www.coalesenses.com
sales@coalesenses.com