IrScrutinizer documentation

Table of contents

1 Introduction....3
1.1 Background....4
1.2 Copyright and License....4
1.3 Privacy note....5
2 Overview....5
3 Installation....6
3.1 General....6
3.2 Properties....6
3.3 Windows....6
3.4 MacOSX....7
3.5 Other systems (Linux etc)....7
4 Concepts....7
5 Analyzing a single IR Sequence or IR Signal....8
6 Adding new export formats....8
6.1 Format of the file exportformats.xml....8
7 GUI Elements walk through....9
7.1 The "Scrutinize signal" pane....9
7.2 The "Scrutinize remote" pane....9
7.3 The "Generate" pane....10
7.4 The Import pane....10
7.5 The Export pane....13
7.6 The "Capturing HW" pane....15
7.7 The "Sending HW" pane....17
8 Command line arguments....19
9 Questions and Answers....19

9.1 Does IrScrutinizer completely replaces IrMaster?..... 19
9.2 How do I emulate the war dialer in IrScrutinizer?.....20
9.3 Can I use this program for conveniently controlling my favorite IR controlled device from the sofa?.....20
9.4 The pane interface sucks.....20
9.5 I did something funny, and now the program does not startup, with no visible error messages.....20
9.6 (Windows) When I double click on the IrScrutinizer symbol, instead of the program starting, WinRar (or some other program) comes up.....20
10 References.....21

Note:

This document is not finished yet. In particular, there **are** broken links. See the release notes. Please ask questions in the JP1-forum.

Warning:

Sending undocumented IR commands to your equipment may damage or even destroy it. By using this program, you agree to take the responsibility for possible damages yourself, and not to hold the author responsible.

Date	Description
2013-11-12	Initial version.
2013-12-01	Next unfinished version.
2014-01-22	Version for release 1.0.0, still leaving much to be desired.
2014-06-07	Version for release 1.1.0, some fixes, much still to be done.

Table 1: Revision history

1 Introduction

IrScrutinizer is a powerful program for capturing, generating, analyzing, importing, and exporting of infrared (IR) signals. For capturing and sending IR signals several different hardware sensors and senders are supported. IR Signals can be imported not only by capturing from one of the supported hardware sensors (among others: IrWidget, Global Caché, and Arduino), but also from a number of different file formats (among others: LIRC, Wave, Pronto Classic and professional, RMDU (partially), and different text based formats; not only from files, but also from the clipboard, from URLs, and from file hierarchies), as well as the Internet IR Databases by Global Caché and by IRDB. Imported signals can be decoded, analyzed, edited, and plotted. A collection of IR signal can thus be assembled and edited, and finally exported in one of the many supported formats. In addition, the program contains the powerful IrpMaster IR-renderer, which means that almost all IR protocols known to the Internet community can be generated.

Written in Java (with the exception of two native libraries), most of the functionality of the program is available on every Java platform. The native libraries (DecodeIR and RXTX) are presently available for 32- and 64-bit versions of Windows, Linux (x86 and amd-64), and MacOsX, and can with moderate effort be compiled for other platforms.

For someone with knowledge in the problem domain of IR signals and their parameterization, this program is believed to be simple to use. This knowledge is assumed from the reader. Other can acquire that knowledge either from the JP1 Wiki or, e.g., this link.

Note that screen shots are included as illustrations only; they may not depict the current program completely accurately. They come from different versions of the program, using different platforms (Linux and Windows), and using different "look and feels".

The present document is written more for completeness than for easy accessibility. Possibly, in the future, there will be a user's manual as well as a reference manual.

Here are the current release notes.

1.1 Background

First, in 2011, I wrote an IR signal "engine" called IrpMaster. It can also be invoked as a command line program. Then a program called IrMaster was released, which among other things constitutes a user friendly GUI front end to IrpMaster. The present program, IrScrutinizer, is also based on IrpMaster, and adds functionality from IrMaster, in particular the possibility to collect IR signals, a vastly improved import and export facility, and edit collections of IR commands. IrScrutizer almost completely replaces IrMaster. It is planned to release a final version before the end of 2013 (slightly ironically called version 1.0.0). No further development is planned.

1.2 Copyright and License

The program, as well as this document, is copyright by myself. My copyright does not extend to the embedded "components" Analyze, Makehex, DecodeIR, and Jirc. ExchangeIR was written by Graham Dixon and published under GPL3 license. Its Analyze-function has been translated to Java by Bengt Martensson. DecodeIR was originally written by John S. Fine, with later contributions from others. It is free software with undetermined license. IrpMaster is using ANTLR3.4 and depends on the run time functions of ANTLR3, which is free software with BSD license.

The "database file" IrpProtocols.ini is derived from DecodeIR.html, thus I do not claim copyright.

The program uses JCommander by Cédric Beust to parse the command line arguments. It is free software with Apache 2 license.

Icons by Everaldo Coelho from the Crystal project are used; these are released under the LGPL license.

The Windows installer was built with Inno Setup, which is free software by Jordan Russel. To modify the user's path in Windows, the Inno extension modpath by Jared Breland, distributed under the GNU Lesser General Public License (LGPL), version 3.

Serial communication is handled by the RXTX library, licensed under the LGPL v 2.1 license.

JSON handling is implemented using the "fast and minimal JSON parser for Java" by Ralf Sernberg, licensed under the Eclipse Eclipse Public License Version 1.0.

LIRC (Linux Infrared Remote Control) is according to its web site copyright 1999 by Karsten Scheibler and Christoph Bartelmus (with contribution of may others), and is licensed under GPL2. The parts used here have been translated to Java by myself, available here.

Tonto was written by Stewart Allen, and is licensed under the "Artistic License".

The contained Arduino firmware contains code copyrighted by Michael Dreher (IrWidget, GPL2 or later) and Chris Young (IRLib LGPL).

The program contains icons from Global Caché, Dangerous Prototypes, Arduino, and IrTrans. These are used exclusively in the context of these firms, and only used to illustrate their products. The icons for JP1 and LIRC are also exclusively used to illustrate themselves.

The program and its documentation are licensed under the GNU General Public License version 3, making everyone free to use, study, improve, etc., under certain conditions. File formats, their description in human- or machine-readable form (DTDs, XML Schemas), are placed in the public domain.

1.3 Privacy note

Some functions (Help -> Project Home page, Help -> IRP Notation Spec, Help -> Protocol Specs, Tools -> Check for updates) access the Internet using standard http calls. This causes the originating machine's IP-address, time and date, the used browser, and possibly other information to be stored on the called server. If this is a concern for you, please do not use this (non-critical) functionality (or block your computer's Internet access).

2 Overview

Next a high-level view of the different use cases will be given.

Analyze ("Scrutinize") individual IR Signal/Ir Sequence
An IrSignal or IrSequence can be captured from connected hardware, or imported from files in different formats, the clipboard, or from Internet databases. The IrSequence can be broken into a beginning-, repeat-, and ending sequence, and decoded, analyzed, and plotted. It can be exported in different formats, or sent to different transmitting hardware.

Analyze/edit/compose/export collections of IR signals ("remotes")
A collection of commands can be assembled either from individual IR signals (as above), captured several at a time, or imported from files in different formats, the clipboard, or from Internet databases. The collection and the individual commands can be edited as in a spreadsheet. It can be exported in a number of different formats.

Generate IR Signals from known protocols
IR Signals can be generated from the Internet's largest protocol data base, containing over 100 protocol. Necessary protocol parameter values are to be entered. Thus

generated signals can be analyzed as single signals, incorporated into remotes, or exported to files — also from e.g. intervals of parameters.

3 Installation

3.1 General

IrScrutinizer, and all but two of its third-party additions, are written in Java, which means that it should run on every computer with a modern Java installed; Windows, Linux, Macintosh, etc. Java 1.6 or later is required. The two exception are DecodeIR and the native part of RXTX, which are written in C++ and C respectively, and invoked as shared library (`.dll` in Windows, `.so` in Linux, etc). If DecodeIR or RXTX are not available on your platform it is not a major problem, as IrScrutinizer will work fine without it; just the DecodeIR-related functions or the serial hardware access will be unavailable.

There is unfortunately no good `make install` or such in the source distribution, so also source code distribution users are recommended to install the binary distribution. Also, all necessary third-party components are included in the binary distribution.

Both under Windows as well as under other operating systems, IrScrutinizer behave civilized, in that they do not write in the installation directory after the initial installation. In both cases (in contrast to the source distribution), the distribution contains everything needed including third party libraries.

3.2 Properties

Under Windows, the properties are stored in `%LOCALAPPDATA%\IrScrutinizer\IrScrutinizer.properties.xml` using Windows Vista and later (on my Windows 7 system, this is `%HOME%\AppData\Local\IrScrutinizer\IrScrutinizer.properties.xml`), otherwise in `%APPDATA%\IrScrutinizer\IrScrutinizer.properties.xml`. Using other operating systems, it is stored under `$HOME/.IrScrutinizer.properties.xml`. It is not deleted by un-install. (If weird problems appear when updating, try deleting this file.)

3.3 Windows

Download the Window setup file, save, and double click. Select any installation directory you like; suggested is `C:\Program Files\IrScrutinizer`, unless you are installing from an account without administrator rights. Unless reason to do so, create the start menu folder, and the desktop icon. Administrator rights are not needed, unless are installing in a directory like `Program Files`. IrScrutinizer can now be started from `Start -> IrScrutinizer -> IrScrutinizer`, or from the desktop icon.

To un-install, select the un-install option from the Start menu. Very pedantic people may like to delete the properties file too, see above.

3.4 MacOSX

Download and double click the binary distribution. Unpack it to a directory of your choice, e.g. on the desktop. Just double clicking the file IrScrutinizer.jar should now start the program. Otherwise, try the "Other systems" instructions and adapt the wrapper `irscrutinizer.sh`.

3.5 Other systems (Linux etc)

For some reason, double clicking an executable jar file in my Gnome installation does not start the program, but starts a browser for the jar file (which is really a form of Zip-Archive). Instead:

Create an installation directory (suggestion; `/usr/local/irscrutinizer`), and unpack the current binary distribution therein. Examine the wrapper `irscrutinizer.sh`, and, if desired, make desired changes to it with your favorite text editor. Then make a symbolic links from a directory in the path (suggestion; `/usr/local/bin` to the newly installed `irscrutinizer.sh`, using the name `irscrutinizer`. Example (using the suggested directories)

```
cd /usr/local/bin
ln -s ../irscrutinizer/irscrutinizer.sh irscrutinizer
```

(`su` (or `sudo`) may be necessary to install in the desired locations.)

To un-install, just delete the files. Very pedantic people may like to delete the properties file too, see above.

4 Concepts

For anyone familiar with the problem domain, this program is believed to be intuitive and easy to use. Almost all user interface elements have tool-help texts. Different panes have their own pop-up help. In what follows, we will not attempt to explain every detail of the user interface, but rather concentrate on the concepts. Furthermore, it is possible that new elements and functionality has been implemented since the documentation was written.

This program does not disturb the user with a number of annoying, often modal, pop ups, but directs errors, warnings, and status outputs to the *console window*, taking up the lower third of the main window. This window is re-sizeable. There is a context menu for the console, accessible by pressing the right mouse button in it.

In the upper row, there are four pull-down menus, named File, Edit, Actions, Options, Tools, and Help. Their usage is believed to be mainly self explanatory, with some the exceptions.

Options to the program are in general found in the Options menu, or its subordinate menus. Some parameters for particular export formats are found in the sub-panes of the "Export" pane. Also the hardware configuring panes contain user parameters.

The main window is composed of seven sub panes denoted by "Scrutinize signal" (for processing single signal), "Scrutinize remote" (for collecting several signals to one "remote"), "Generate" (generates IR signal from protocol name and parameters), "Import", "Export", "Capturing Hardware", and "Sending Hardware" respectively. These panels will be discussed in Section GUI Elements walk through

5 Analyzing a single IR Sequence or IR Signal

The pane "Scrutinize Signal" is devoted to the analysis of one single IR sequence.

To capture IR Sequences from a hardware sensor, first set it up and open it, see Section Capturing Hardware. An IR Sequence is captured by pressing the "Capture" button, and sending an IR signal to the selected hardware. Note that the hardware captures an *IR Sequence*, not an IR Signal. It consists of an (sometimes empty) start sequence, an unknown number of repeat sequences, and sometimes an ending sequence.

6 Adding new export formats

Only some of the many export formats are defined in the Java code of the program, the rest in the file `exportformats.xml`, located in the root of the install directory. By modifying this file, the user can simply add his/her own export formats according to own needs. An exporformat consists of a number of properties, together with a small "program" written in the transformation language XSLT, for transforming a Girr-XML-tree to the desired text format.

The rest of this section documents the format of the file, and is supposed to be read only when needed. Fundamental knowledge of XML and XSLT transformations are assumed.

6.1 Format of the file exportformats.xml

The file is an XML file read in without validation. The syntax and semantics are beleived to be essentailly self explaining or clear from the examples aleady in there. An export format is packed in an element of type `exportformat`. It contains the following attributes:

name
Text string used for identifying the format.
extension
Text string denoting preferred file extension (not including period) of generated files.
multiSignal
Boolean (value: true and false). Denotes if several signals can be represented in one export, or only one.
simpleSequence
Boolean, (values `true` of `false`). If true, the export really describes an sequence rather than am signal (with intro-, repeat- , and ending-sequences), therefore the user must explictly state a particular number of repetitions for an export.

The element contains a single child element, namely the XSLT transformation, which is an element of type `xsl:stylesheet`, with attributes as in the examples. It should transform a Girr XML DOM tree in the "fat" format, with `remotes` as root element, into the desired text format. It may be advisable to use the already present formats as guide.

After editing the file, it can be reloaded either by re-starting the program, or by selecting Options -> Export formats database -> Reload.

7 GUI Elements walk through

7.1 The "Scrutinize signal" pane

This panel is devoted to the analysis of a *single IR signal* or IR sequence. A (single) signal is either read from hardware using the "Capture" button (requires that the capturing hardware has been set on the "Capturing Hardware" pane), imported from a file (using the context menu in the data window, or through File -> Import -> Import as single sequence), or pasted from the clipboard. Also, some other panes can transfer data to this pane. For text import, the signal can be in either Pronto CCF format, raw format (indicated by a leading "+"-sign), or in the UEI learned format. The signal is printed in the data window, in the preferred text format, which can be selected from the options menu. The text representation may be edited (assuming sufficient knowledge!), after which the edited signal is analyzed and plotted again by pressing the "Scrutinize" button. The signal may be sent to the sending hardware by pressing the "Transmit" button.

Using context menus, the result can be sent to the clipboard or saved to a file.

The plot can be zoomed by pressing the left mouse button and dragging. Printing and exported as graph is presently not implemented.

The menu entry Actions -> Enter test signal (or its accelerator, the F9 key) enters a test signal.

In rare cases, transforming the signal between different formats may introduce some rounding errors causing decoding to fail.

7.2 The "Scrutinize remote" pane

This panel is devoted to the capturing/import/editing of a collection of IR signals, called "a remote" in the sequel. The panel contains two sub-panels: for parametric signals and for non-parametric, "raw", signals.

A "parametric" signal is determined by its protocol name, and the values of the protocol's parameters. A "raw" signal is determined by its timing pattern, and its modulation frequency. It may have one or many decodes, or none. Nevertheless, by definition, a raw signal is determined by its timing, not the decodes.

In both cases, the sub panes consists of tables with a number of columns. Every signal takes up a row in the table. The content of the individual cells (with the exception of its number and date) can be individually edited, like in a spreadsheet program.

In both tables, the right mouse button opens a context menu containing a number of ways to manipulate the table, its view, or the data contained therein. By enabling the row selector, the rows can be sorted along any of the present columns.

To capture a number of IR signals, first configure the hardware using the capturing hardware pane. Next press the Capture button. The program will now run the capturing in a separate thread, so the user just have to press the buttons of the remote. The signals will be received, interpreted, decoded, and entered on subsequent lines in the selected table (raw or parameterized). The capture thread will continue until the captured button is pressed again. (Note that this is completely different from the capture button on the "Scrutinize signal" panel.) The user may mix captures with other activities, like entering information (name, comments,...) in the table.

The export button exports the content of the currently selected table (raw or parameterized) according to the currently selected export format.

The menu entry Actions -> Enter test signal (or its accelerator, the F9 key) enters a test signal, either as parametric signal, or as a raw signal.

7.3 The "Generate" pane

In the upper part of this pane, an IR protocol is selected, identified by name, and the parameters D ("device", in almost all protocols), S ("sub-device", not in all protocols), F ("function", also called command number or OBC, present in almost all protocols), as well as T, "toggle" (in general 0 or 1, only in a few protocols). These number can be entered as decimal numbers, or, by prepending "0x", as hexadecimal numbers.

By pressing "Generate", the signal is computed, and the middle window is filled with a textual representation, in the form selected by Options -> Output Text Format.

The Export button initiates an export to a file format selected by the Export pane. The three lower buttons transfer the signal(s) to the scrutinize signal panel, the raw remote table, or the parameterized panel.

7.3.1 Accessing a number of different parameter values

For the export and the transfer to the "scrutinize remote" tables, not only a single parameter value can be selected, but whole sets. The complete syntax and semantics is given in the IrpMaster documentation, we here just mention that e.g. 12:34 means all numbers between 12 and 34 (inclusive), and * denotes all possible values (as defined by the protocol's IRP notation).

7.4 The Import pane

The import pane allows for selective import of collection of IR commands. Both Internet data bases and file formats are supported. Import can take place from local files or even file hierarchies, from the clipboard, or from Internet URLs.

There are a number of elements common to most of the sub-panes, so these will be described next.

For file/URL based imports, there is a text field, named File or File/URL. For the latter case, an URL (like `http://lirc.sourceforge.net/remotes/yamaha/RX-V995` can be entered, for subsequent import without downloading to a local disc. By pressing the "..."-Button, a file selector allows the selection of a local file. For files and URLs, the "Edit/Browse" button allows to examine the selected file/URL with the operating system's standard command.

If the option "`Open ZIP files`" (accessible from `Options -> Import`) is selected, zip files can be selected, opened, and unzipped "on the fly", without the need for the user to unzip to intermediate files.

When pressing one of the "Load", "Load File/URL", or "Load from clipboard" button, the selected information is downloaded, and presented in the format of an expandable tree. By placing the mouse cursor over a command, additional information, like decode, is presented. A single command can be selected by mouse click, a sequence of adjacent commands by shift-click, a subset of not necessarily adjacent commands be selected by Ctrl-click, as usual from most GUIs. A single selected command can be transferred to the "Scrutinize signal" pane by pressing "Import signal". The "Import all" ("Import selection") button transfers all commands (the selected commands) to the "Scrutinize remote" pane, sub-pane "Parametric remote" (without overwriting already present commands), while the buttons "Import all/raw" and "Import selected/raw" transfer to the sub-pane "Raw remote".

The key "Transmit selected" transmits the (single) selected signal to the selected sending hardware.

7.4.1 Global Caché Database

Global Caché maintains a data base of IR signals, made available free of charge. "Contains over 100,000 Infrared codes for over 2,000 different remotes from over 500 manufacturers". To use from IrScrutinizer, an API Key has be be retrieved. This can be done from a Facebook, Google, or Yahoo account. After pressing the "APIKey" button, the API key is entered in the pop-up window. It is subsequently saved to the program's properties. To use, select, in order, a manufacturer, a device type, and a setup code, the latter possibly by trial-and-error.

7.4.2 The IRDB Database

IRDB is "one of the largest crowd-sourced, manufacturer-independent databases of infrared remote control codes on the web, and aspiring to become the most comprehensive and most accurate one."

To use, select, in order, a manufacturer, a device type, and a protocol/parameter combination, the latter possibly by trial-and-error.

Pressing the "Load all" button transfers all present protocol/parameters combinations to the tree.

7.4.3 Girr (the native format of IrScrutinizer)

The Girr format is the native format of IrScrutinizer.

7.4.4 LIRC

The LIRC import feature is based upon Jirc, which is basically LIRC translated into Java. The LIRC importer can even import a file system hierarchy by selecting the top directory as File/URL. (Importing the entire lirc.org database with over 100000 commands takes around 1 minute and 1GB of memory.)

7.4.5 RemoteMaster

The JP1 community has a large data base of parametric IR commands. IrScrutinizer has support for importing RMDU files for RemoteMaster. Unfortunately, the signals are stored as parameters for so-called executors, with sometimes different parameterization ("hex", "efc") than the IRP protocols. Translating these files to one of the known protocol/parameter format is nothing but straightforward. It uses protocol information contained in protocols.ini. IrScrutinizer reads this file, and can do some computations, for example on NEC1 protocols, but not on all protocols.

For signals without recognized protocol name, importing as raw signals, or to "Scrutinize signal", is not possible. However, they can always be imported as parametric signals, possibly for manual edit.

7.4.6 Pronto Classic (CCF format)

Many Pronto CCF files are available in Internet, in particular by Remote Central. IrScrutinizer can read in these files to its import tree, even preserving the Pronto "devices" as nodes in the tree.

Unfortunately, the Tonto library used produces the warning message `Error loading JComm JNI library`. This should be ignored.

7.4.7 Pronto Prof. (XCF format)

Pronto Professional XCF files are found for example at Remote Central. IrScrutinizer can read in these files to its import tree, even preserving the Pronto "devices" as nodes in the tree.

7.4.8 ICT IrScope format

The ICT format, introduced by Kevin Timmerman's IrScope, contains the timing pattern, the modulation frequency, and optionally a name ("note") of one or many IR signals.

7.4.9 Text format

In the Internet, there are a lot of information in table-like formats, like Excel, describing the IR commands of certain devices. IrScrutinizer has some possibilities of importing these — after exporting them to a text format, like tab separated values (tsv) or comma separated values.

7.4.9.1 Raw

The sub-pane allows for the parsing of text files separated by a certain characters, like commas, semicolons, or tabs. The separating characters is selected in the "Field separator" combo box. The column to be used as name is entered in the "Name col." combo box, while the data to be interpreted either as raw data or CCF format, is entered in the "Raw signal col.". If the "... and subsequent columns" is selected, all subsequent columns are added to the data.

7.4.9.2 Raw, line-based

This pane tries to interpret a line-based file as a number of named IR commands, using heuristics.

7.4.9.3 Parameterized

The sub-pane allows for the parsing of text files separated by a certain characters, like commas, semicolons, or tabs. The separating characters is selected in the "Field separator" combo box. The column to be used as name is entered in the "Name col." combo box, while protocol name and the parameters D, S, and F are entered in their respective combo boxes. They are parsed in the number base selected.

7.4.10 Wave

This pane imports and analyzes wave files, considered to represent IR signals. The outcome of the analysis (sample frequency, sample size, the number of channels, and in the case of two channels, the number of sample times the left and right channels are in phase or anti-phase) is printed to the console.

7.5 The Export pane

Using the export pane, export files can be generated, allowing other programs to use the computed results. Single signals (from the "Scrutinize signal" pane), collections of signals (from the "Scrutinize remote" pane), or generated signals can be exported. Exports can be generated in a number of different formats. Some (Girr (=XML) and text) can contain both the Pronto format and the "raw" format (timings in microseconds, positive for pulses, negative for gaps), as well as other formats. These formats, together with Wave, LIRC, and Pronto Classic, are built-in in the program. However, it is possible

to define new export formats by extending a configuration file, see Adding new export formats. The formats are, at the time of this writing:

Girr
The program's native format, based on XML. Very flexible and extensible. Can contain information like the raw format, CCF format, UEI learned format, and the Global Caché sendir format. Documentation.

Text
The text format is essentially the Girr format stripped of the XML markup information.

Wave
IR sequences encoded as wave audio files.

LIRC
The LIRC-exports are in lirc.conf-format using the raw LIRC format. They can be concatenated together and used as the LIRC server data base. Can also be used with WinLirc.

Pronto Classic
This format generates a CCF configuration file to be downloaded in a Pronto, or opened by a ProntoEdit program.

IrTrans
This export format generates .rem files for the IrTrans system, using its CCF format.

Lintronic
Simple text protocol for describing a single IrSequence.

Spreadsheet
Simple tab separated value export format for importing in a spreadsheet program.

RM Functions
Variant of the Spreadsheet format, this format is intended to be pasted directly into the "Functions" table of RemoteMaster.

C
Intended mostly as an example of generating C code, cf. this article.

TV-B-Gone
Variant of the C format, this format generates C code for the TV-B-Gone.

Export file names are either user selected from a file selector, or, if "Automatic file names" has been selected, automatically generated.

The export is performed by pressing the "Export" button. The "..."-marked button allows for manually selecting the export directory. It is recommended to create a new, empty directory for the exports. The just-created export file can be immediately inspected by pressing the "Open last file"-button, which will open it in the "standard way" of the used operating system. (Also available on the actions menu.) The "Open" button similarly opens the operating systems standard directory browser (Windows Explorer, Konquistador, Nautilus,...) on the export directory.

Some export formats (presently Wave and Lintronic) export an IR sequence rather than an IR signal (consisting of an intro sequence, an repetition sequence (to be included 0

or more times), and an (most often empty) ending sequence). Using these formats, the number of repetition sequences to include can be selected.

Some export formats have some more parameters, configured in sub panes. These will be discussed next.

7.5.1 The Girr sub-pane

A style sheet can be selected to be linked in into the exported Girr file. The type of style file (presently xslt and css) can also be selected.

"Fat form raw" can be selected; this means that the raw signals are not given as a text string of alternating positive and negative numbers, but the individual flashes and gaps are enclosed into own XML elements. This can be advantageous if generating XML mainly for the purpose of transforming to other formats.

7.5.2 The Wave sub-pane

Parameters for the generated Wave export (except for the number of repeats) can be selected here.

7.5.3 The sendir sub-pane

The Global Caché sendir format requires a module number and a connector number. Also, there is a compressed format, that can be enabled by selecting the compressed check-box.

7.5.4 The Pronto Classic sub-pane

A Pronto Classic export consists of a CCF file with the exported signals associated to dummy buttons. The Pronto (Classic) model for which the export is designed is entered in the combo box. Screen size of the Pronto is normally inferred from the model, but can be changed here. The button size of the generated buttons is also entered here.

7.6 The "Capturing HW" pane

The sub-panes of this pane allow for the configuration of capturing hardware. Selecting a sub-pane also selects the associated hardware, if possible. The hardware can also be selected from the tool bar, Options -> Capturing hardware.

Unfortunately, by e.g. selecting non-existing hardware or such, there is a possibility to "hang" the program.

After configuring and opening the capturing hardware, the "Test" button can be used for testing the configuration without switching pane.

Selected ports are stored in the properties, thereby remembered between sessions. So, for future sessions, only opening the preferred device is necessary.

7.6.1 IrWidget

Plug the IrWidget it into the computer. Check that the operating system has assigned a port to it, and note which one it is. On Windows: open the device manager, and check that there is one "USB Serial Port" under Ports. Note the port number (e.g. `COM8`). On a Linux system, it likely shows up as a device like `/dev/ttyUSB0`. If the port does not show up, a suitable driver needs to be installed. If the correct port is already visible in the combo box, just press "Open". Otherwise, press "Refresh", which makes the program determine the available serial ports. Select the correct one. Press "Open". which should now remain "pressed". The port cam be closed again by a repeated press, but there is not much reason to do so, unless another capturing hardware should be used, or the IrWidget should be used from another program.

7.6.2 Global Caché capture

IrScrutinizer automatically detects alive Global Caché units in the local area network, using the AMX Beacon. However, this may take up to 60 seconds, and is not implemented in very old firmware. Using the "Add" button, the IP address/name of older units can be entered manually.

The "Browse" button points the browser to the selected unit.

The reported type and firmware version serves to verify that the communication is working.

7.6.3 LIRC Mode2

`mode2` is a program from the LIRC distribution, that prints timing information in a simple text format to its standard-out. In theory, any program that prints information in that format can be used. The command line for the program with possible parameters is to be entered as command. With the Start button, a sub-process is started, running the given command line. The "Stop" button stops the sub-process — although sometimes this may not stop the started program.

Has been tested only on Linux, should however work on all systems.

7.6.4 Arduino

To use the Arduino with a non-demodulating receiver for IR capture, the sketch `GirsLite` needs to be loaded.

After connecting the Arduino to a real or virtual serial port, select the port in the combo box, pressing "Refresh" if necessary. "Open" the port thereafter.

At least on Linux, the ACMX port may be finicky. Sometimes disconnecting and reconnecting the device may help.

7.6.5 IrToy

After connecting the IrToy to an USB port, select the virtual serial port in the combo box, pressing "Refresh" if necessary. "Open" the port thereafter.

The ending timeout is 1.4 seconds, and cannot be changed. This is not optimal for most IR signal capture use cases.

At least on Linux, the ACMX port may be finicky. Sometimes disconnecting and reconnecting the device may help.

7.7 The "Sending HW" pane

The sub-panes of this pane allows for the selection and configuration of the deployed IR sending hardware.

7.7.1 The "Global Caché" pane.

IrScrutinizer automatically detects alive Global Caché units in the local area network, using the AMX Beacon. However, this may take up to 60 seconds, and is not implemented in very old firmware. Using the "Add" button, the IP address/name of older units can be entered manually.

The "Browse" button points the browser to the selected unit.

The reported type and firmware version serves to verify that the communication is working.

"Stop IR"-Button allows the interruption of ongoing transmission, possibly initiated from another source.

The user can select one of the thus available Global Caché units, together with IR-module and IR-port (see the Global Caché API specification for the exact meaning of these terms).

7.7.2 The "LIRC" pane

To be fully usable for IrScrutinizer, the LIRC server has to be extended to be able to cope with CCF signal not residing in the local data base, but sent from a client like IrScrutinizer, thus mimicking the function of e.g. a Global Caché. The needed modification ("patch") is in detail described here. However, even without this patch, the configuration page can be used to send the predefined commands (i.e. residing it its data base `lirc.conf`). It can be considered as a GUI version of the `irsend` command.

The LIRC server needs to be started in network listening mode with the `-l` or `--listen` option. Default TCP port is 8765.

After entering IP-Address or name, and port (stay with 8765 unless a reason to do otherwise), press the "Read" button. This will query the LIRC server for its version (to replace the grayed out "<unknown>" of the virgin IrScrutinizer), and its known remotes

and their commands. Thus, the "Remote" and "Command" combo boxes should now be selectable. After selecting a remote and one of its command, it can be sent to the LIRC server by pressing the "Send" button. If (and only if) the LIRC server has the above described patch applied, transmitting signals to "LIRC" now works.

Due to LIRC's peculiar form of API stop command, the "Stop IR" command presently does not work. See this thread in the LIRC mailing list for a background.

7.7.3 The "IRTrans" pane

The configuration of IRTrans is similar to LIRC, so it will be described more briefly.

Enter IP name or -address and select an IR port (default "intern"). If the Ethernet IRTrans contains an "IR Database" (which is a slightly misleading term for an internal flash memory, that can be filled by the user), its commands can be sent from this pane. By pressing the "Read" button, the known remotes and commands are loaded, and the version of the IRTrans displayed. The selected command can now be sent by the "Send" button. (However, this functionality is otherwise not used by IrScrutinizer.) Selecting "IRTrans" on the "Analyze" and "War dialer" pane should now work. The IRTrans module is then accessed using the UDP text mode.

7.7.4 The "IrToy" Pane

Using this pane, the IrToy (version 2) can be used to transmit IR signals.

7.7.5 The "Arduino" Pane

Using this pane, an Arduino equipped with a suitable IR Led can be used to transmit IR signals. First however, the sketch `GirsLite` from the arduino subdirectory has to be downloaded to the Arduino.

7.7.6 The "Audio" Pane

As additional hardware device, IrScrutinizer can generate wave files, that can be used to control IR-LEDs. This technique has been described many times in the Internet the last few years, see for example this page within the LIRC project. The hardware consists of a pair of anti-parallel IR-LEDs, preferably in series with a resistor. Theoretically, this corresponds to a full wave rectification of a sine wave. Taking advantage of the fact that the LEDs are conducting only for a the time when the forward voltage exceeds a certain threshold, it is easy to see that this will generate an on/off signal with the double frequency of the original sine wave. (See the first picture in the LIRC article for a picture.) Thus, a IR carrier of 38kHz (which is fairly typical) can be generated through a 19kHz audio signal, which is (as opposed to 38kHz) within the realm of medium quality sound equipment, for example using mobile devices.

IrScrutinizer can generate these audio signals as wave files, which can be exported from the export pane, or sent to the local computers sound card. There are some settings

available: Sample frequency (44100, 48000, 96000, 192000Hz), sample size (8 or 16 bits) can be selected. Also "stereo" files can be generated by selecting the number of channels to be 2. The use of this feature is somewhat limited: it just generates another channel in opposite phase to the first one, for hooking up the IR LEDs to the difference signal between the left and the right channel. This will buy you double amplitude (6 dB) at the cost of doubling the file sizes. If the possibility exists, it is better to turn up the volume instead.

Most of "our" IR sequences ends with a period of silence almost for the half of the total duration. By selecting the "Omit trailing gap"-option, this trailing gap is left out of the generated data – it is just silence anyhow. This is probably a good choice (almost) always.

Note that when listening to music, higher sample rates, wider sample sizes, and more channels sound better (in general). However, generating "audio" for IR-LEDs is a completely different use case. The recommended settings are: 48000kHz, 8bit, 1 channel, omit trailing gap.

7.7.7 The "General Serial Port" Pane

This pane contains the controls for sending a signal in a general format to one of the serial ports available on the system.

8 Command line arguments

Normal usage is just to double click on the jar-file, or possibly on some wrapper invoking that jar file. However, there are some command line arguments that can be useful either if invoking from the command line, or in writing wrappers, or when configuring custom commands in Windows.

The options `--version` and `--help` work as they are expected to work in the GNU coding standards for command line interfaces. Use the `--help`-command to see the complete list of command line parameters. The `-v/--verbose` option set the verbose flag, causing commands like sending to IR hardware printing some messages in the console.

For automating tasks, or for integrating in build processes or Makefiles or the like, it is probably a better idea to use IrpMaster instead, which has a reasonably complete command line interface.

The program delivers well defined and sensible exit codes.

9 Questions and Answers

9.1 Does IrScrutinizer completely replaces IrMaster?

Almost. Using MakeHex as renderer (or more correctly, its Java version) instead of IrpMaster is not implemented. (The practical usage of this feature is probably *very* limited, and IrMaster is still available, should it ever be needed.) The "war dialer" is also

not implemented, but see next question. For the wave export, some rarely used options (the possibility to select big-endian format (for 16-bit samples), the possibility *not* to half the carrier frequency, and the possibility to select sine (instead of square) for modulation) have been removed. Finally, there is some stuff that simply works differently, like the export function.

9.2 How do I emulate the war dialer in IrScrutinizer?

Use "Scrutinize remote" -> Parametric Remote. Fill in the table with signals to be tested, either using the pop-up button (right mouse in the table) Advanced -> Add missing F's, or from the Generate pane, using suitable parameter intervals (see TODO), and transfer them using the "To parametric remote" button. Then test the candidate signals one at a time by transmit-ting them, using suitable sending hardware. The comment field, or the "verified" check-box, can be used for note taking.

A "war dialer" like in IrMaster may be implemented in a later version.

9.3 Can I use this program for conveniently controlling my favorite IR controlled device from the sofa?

No, the program is not meant for that. While you definitely can assemble a "remote" on the "scrutinize remote" panel, and transmit the different commands with mouse commands (appropriate hardware assumed), the program is intended for developing codes for other deployment solutions.

9.4 The pane interface sucks.

Yes. There are several use cases when the user would like to see several "panes" simultaneously. Also, it should be possible to have several windows of the same sort (like the "scrutinize signal") simultaneously. Replacing the top level panes with something "Eclipse-like" (sub-windows that can be rearranged, resized, moved, iconized) is on my wish list.

9.5 I did something funny, and now the program does not startup, with no visible error messages.

Try deleting (or renaming) the properties file. If that does not help, try starting the program from the command line, which may leave hopefully understandable error message on the console.

9.6 (Windows) When I double click on the IrScrutinizer symbol, instead of the program starting, WinRar (or some other program) comes up.

The program that comes up has "stolen" the file association of files with the extension `.jar`. Restore it. (Allegedly, WinRar can gracefully "unsteal" file associations.)

10 References

1. IrpMaster. Also a GPL3-project by myself. Much harder to read than the present document :-). See also this discussion thread in the JP1 forum.
2. IpMaster. Also a GPL3-project by myself. The precessor of the this program. See also this discussion thread in the JP1 forum.
3. The Harctoolbox project, also a GPL3-project by myself.
4. DecodeIR. This shared library tries to identify protocol name and parameters of an IR signal in raw form. Thus, it is in a sense, it implements the "inverse mapping" of IrpMaster.
5. GlobalCaché, a manufacturer of Ethernet connected IR hardware. Note that I have only tried with the GC-100 series, but the IR sending models of theiTach family are believed to work too. (Feel free to send me one :-).)
6. IRTrans, another manufacturer of Ethernet connected IR-hardware. The "IRTrans Ethernet" module, preferably with "IRDB Option" (internal flash memory), is directly supported by the current software.
7. LIRC, Linux InfraRed Control This project contain drivers for almost everything IR-related. The present project is able to use a modified LIRC-server for transmitting IR signals.