IrScrutinizer documentation

Table of contents

Note:

This is a reference manual. It is written for completeness and correctness, not for accessibility. For an easier introduction, see this tutorial. Or, just try the program, and come back to the manual if and when you need it.

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.
2014-09-21	Version for release 1.1.1, some more fixes and enhancements.
2015-04-07	Version for release 1.1.2, installation issued reworked. Misc. minor improvements.
2015-08-19	Version for release 1.1.3. Misc. minor improvements.
2016-01-13	Added description for user selectable character sets for import and export.
2016-04-29	Version for release 1.2. Misc. minor improvements.
2016-08-30	Version for release 1.3. Misc. minor improvements.
2017-03-12	Version for release 1.4. Misc. minor improvements.
2019-08-09	Version for release 2.0.0. Not complete...
2020-05-03	Updated installation instructions.
2020-05-20	Minor updates for version 2.2.6.
2020-07-30	Updated building instructions.
2022-05-13	Updated for 2.4.0: Considerable update and rewrite, for upcoming release 2.4.0.
2022-12-19	Misc. fixes and updates.

Date	Description
2023-12-17	Accessing HarcHardware command line is now supported in AppImages.
2023-12-26	Removed leftover references to the "minimal json" as well as Fedore dnf tonto.
2024-01-05	Removed GlobalCache IR database.
2024-03-12	Added link to Lirc imports with and without timing info. Updated command line parameters. Misc fixes and updates.

Table 1: Revision history

1 Introduction

IrScrutinizer is a powerful program for capturing, rendering, 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, Command Fusion, and Arduino), but also from a number of different file formats (among others: Lirc, Wave, CML, Pronto Classic and -professional, 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 visualized. 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 IrpTransmogrifier 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 library (NRJavaSerial) is presently available for 32- and 64-bit versions of Windows, Linux (x86, amd-64, arm version 7), and MacOsX, and can with moderate effort be compiled for other platforms. (The library DevSlashLirc is available (and meaningful!) for Linux only.)

For someone with knowledge in the problem domain of IR signals and their parametrization, 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".

Sources are hosted on Github. Bug reports and enhancement requests are welcome (e.g. as discussions, issues), or as contributions (code, testing, documentation, use cases, protocols etc.).

The present document is written more for completeness than for easy accessibility. For an easier introduction, see this tutorial.

Here are the current release notes.

1.1 Background

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, was 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. The final version of the latter was released in February 2014, slightly ironically called version 1.0.0. No further development is planned.

The final version of IrScrutinizer using IrpMaster and the decoding engine DecodeIR is called 1.4.3, and is available here. Since version 2.0.0, the newer IR engine IrpTransmogrifier has replaced IrpMaster, DecodeIR and Analyzer.

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", like Jirc. IrpTransmogrifier is using ANTLR4 and depends on the run time functions of ANTLR4, which is free software with BSD license.

The "database file" `IrpProtocols.xml` is derived from DecodeIR.html, as well as many, many discussions with, and contributions from, members of the JP1 community, 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 Neuron Robotics NRSerialPort library licensed under the LGPL v 2.1 license. (This is a fork of the discontinued RXTX library.)

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 with the name Jirc.

For Pronto Classic support, Tonto by Stewart Allen was used. It is licensed under the "Clarified Artistic License".

The program contains icons from Dangerous Prototypes 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 the community, their products, and files.

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 (render) 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 Download

The latest official release is always available here.

In addition, there are continuous integration builds, at least most of the time. These are build from a snapshot of the current sources. Because of this, they are more likely to be unstable, contain bugs, or even to be outright useless.

3.2 General

IrScrutinizer, and all but two of its third-party additions, are written in Java, which means that it should run on almost all current computer: Windows, Linux (including Raspberry Pi), Macintosh, etc. Many of the installation options come with its own contained Java, for the other options, the Java runtime (presently version 8 or later) must be installed. The exception is the native part of `NRJavaSerial`, which are written in C , and invoked as native, shared library (`.dll` in Windows, `.so` in Linux, `.jnilib` in Mac OS X). For Linux, there is also the `DevSlashLirc` library, for accessing Lirc compatible hardware. If the shared libraries are not available on for a particular platform it is not a major problem, as most parts of IrScrutinizer will work fine without it; just the serial hardware access, or the access to certain hardware will be unavailable.

In all versions, IrScrutinizer behave civilized, in that the installation directory is not written to, and thus can be read-only as soon as the installation has taken place.

To use all functionality of the program, it may be necessary to explicitly white-list some ports in a firewall. These are listed here.

There are five different way of installing the program, described next.

3.3 Windows

Download the Window setup file and double click it. Select any installation directory you like; suggested is `C:\Program Files\IrScrutinizer`. Select creation of the start menu folder and the desktop icon (unless you prefer it otherwise). If administrator rights are present, the recommended installation location is in a subdirectory of `Program Files`. Otherwise the program can be installed in any directory writable by currently available user rights. The setup comes with a Java runtime, but it is also possible to use a compatible, existing installation. IrScrutinizer can now be started from `Start -> IrScrutinizer -> IrScrutinizer`, or from the desktop icon.

Sometimes Windows will refuse to install programs from sources that it does not consider as trustworthy. It may then be necessary on Settings -> Apps & features to select on "Choose where to get apps" to something else than "The Microsoft store only".

If desired, the installer will associate `.girr` files with the program, allowing them to be opened by double clicking them.

The installer will also install the command line program IrpTransmogrifier, which can be called as `irptransmogrifier` from a command line window. This also goes for HarcHardware.

Sometimes, when starting the program for the first time, the Windows Defender Firewall blocks some features in the program, asking for permission to allow the Java virtual machine (typically ending with `javaw.exe`) access. This should be granted.

To uninstall, select the uninstall option from the Start menu, or (on Windows 10, right click `Start -> IrScrutinizer -> IrScrutinizer` and select Uninstall). Or use the operating system's program management for de-installing (Settings -> Apps and Features). Very pedantic people may like to delete the properties file too, see properties.

3.4 AppImage for 64-bit Linux

An AppImage is a distribution consisting of one large monolithic executable file that is downloaded, made executable, and run, without any installation in the classical sense. At this moment, only 64-bit x86 Linux is supported. It is believed to run on all modern, 64-bit x86 Linux distributions. Just download, make executable, and double click (or start from the command line). It comes with its own Java runtime.

It called using the name `irptransmogrifier`, the IrpTransmogrifier program will be invoked, taking its usual command line arguments. Similarly, if called using the name `harchardware`, the HarcHardware main routine will be invoked.

To uninstall, just delete.

3.5 MacOS app

Download the disk image file (`*.dmg`). Mount it by double clicking. A mounted disk image `IrScrutinizer` will then appear on the desktop. Opening it will show an app, and a few documentation files. The app can just be dragged to the desktop, to the tray, to `Applications` or any other location the user prefers. IrScrutinizer now behaves like any other app in Mac OS X.

Sometimes the operating system will refuse to install or run programs from sources that it does not consider as trustworthy. It may then be necessary on System Preferences -> Security & privacy -> General to select on "Allow apps downloaded from" to "Anywhere".

To uninstall, just drag the program to the trash can.

The command line programs IrpTransmogrifier and HarcHardware are not supported in this mode. (For this, the generic binary distribution has to be used.)

3.6 Generic Binary

The generic binary version consists of all the Java classes packed in one executable jar file, together with supporting files, like all the compiled shared libraries for the different

operating systems Linux (x64-32, x86-64, armv7), Windows (x86, 32 and 64 bits), and Macintosh. It can be used on all those systems. (In other environments, the shared libraries may be compiled with moderate effort.)

The generic binary distribution also can be used whenever using the setup.exe/app installation is not possible, not up-to-date, or just not desired. It can also be used to update an existing installation on, e.g., Windows.

First make sure that the Java runtime of correct version (currently version 8 or later) is installed.

To install, unpack in an empty directory of your choice; suggested (for Unix/Linux) is `/usr/local/share/irscrutinizer`. On Windows, the program can be started by double clicking the .jar file.

The rest of this section applies to Unix/Linux.

There are a few additional steps, which, on Unix/Linux the script `setup-irscrutinizer.sh` can take care of. If this is not desired, or does not work, these steps will be described next.

Inspect the wrapper `irscrutinizer.sh`, and make changes if necessary.

It is recommended to make links from a directory in the path to the wrapper script, e.g.

```
                    ln -s /usr/local/share/irscrutinizer/irscrutinizer.sh /usr/local/
bin/irscrutinizer
                    ln -s /usr/local/share/irscrutinizer/irscrutinizer.sh /usr/local/
bin/irptransmogrifier
                    ln -s /usr/local/share/irscrutinizer/irscrutinizer.sh /usr/local/
bin/harchardware
```

The desktop file `irscrutinizer.desktop` should, if desired, be installed in `/usr/local/share/applications` alternatively `~/.local/share/applications`.

The program can now be started either as a desktop program, or by typing `irscrutinizer` on the command line. Also the command line programs IrpTransmogrifier and HarcHardware can be started by the command `irptransmogrifier` and `harchardware` respectively. It is also possible to start IrScrutinizer by double clicking on the jar file. In case this brings up the archive manager, the desktop needs to be taught to open executable jar files with the "java" application (i.e. the JVM). For this, select a jar file the file browser, select the properties, and select "java" as the application to "open with". (The details might vary.)

On Linux, it is general necessary to check, and possibly fix, the access to the serial devices. The previously mentioned wrapper does this.

On Gnome-like desktops, by copying the file `girr.xml` to the system's data base of known file types (normally located in `/usr/local/share/applications`), `*.girr` files can be opened in IrScrutinizer by double click.

To uninstall, just delete the files. Some may like to delete the properties file too.

3.7 Source distribution

Compiling the sources is covered in the Appendix. This generates the generic binary distribution file, which is installed as described above. Alternatively, it can be installed with the command

```
make install
```

possibly with `sudo` or as root.

3.8 Linux/Unix Serial device access

To access serial devices, including "serial" devices connected over USB, most current Linuxes require the current user to be a member of the group `dialout` (some distributions use the legacy name `uucp`), in some cases (e.g. Fedora) also the group `lock`. To grant user `$USER` this membership, typically a command like `sudo usermod -aG dialout $USER`, and/or `sudo usermod -aG lock $USER` is used.

It has been brought to my attention that Arch Linux (and likely others) still use the older name `uucp` for `dialout`.

Do not run this program as root (except possibly for debugging device access).

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 six 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.

Options are in general saved persistently between sessions, in the properties file.

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

5 GUI Elements walk through

5.1 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 up, see Section Hardware, 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.

The button `Capt. (cont.)` a program tread is started, that continuously captures signals from the hardware. Only the last signal is kept.

As shown, an IR signal consists of an (sometimes empty) start sequence (red), a (sometimes empty) repeat sequences (blue), and sometimes an ending sequence (green).

For text import, the signal can be in either Pronto Hex format or in raw format (indicated by a leading "+"-sign). 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.

Sometimes the "scrutinization" of a signal changes the numbers in an unexpected way. If desired, the previous content of the data window is recalled by `Actions -> Undo Scrutinizer data`.

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

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

Using context menus, the result can be sent to the clipboard or saved to a file. The menu entry `clone plot` makes a clone of the plot. This can be used for visually comparing different sequences or signals.

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

A Girr file can be dropped on the text- or plot windows. This will display the concatenation of the signals in the file.

5.2 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.

By clicking and dragging in the table head, the columns can be rearranged with the mouse.

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.

Clicking a row with the middle mouse button selects that row, and (ir possible) transmits it using the currently selected hardware.

To capture IR signals, first configure the hardware using the 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.

A Girr file can be dropped on the parameter as well as the raw table, which will import the contained signals.

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.

5.2.1 The parametric table columns

#
A unique number assigned to the signal when creating. It is not editable and not exported.
Timestamp

A timestamp generated when the signal was entered or created. It is not editable and not exported.

Protocol

Name of the protocol. In the table, anything can be entered, however, to transmit or export, it has to match (case insensitively!) a known protocol.

D

The `D` parameter value, as defined in the protocol IRP.

S

The `S` parameter value, as defined in the protocol IRP.

F

The `F` parameter value, as defined in the protocol IRP.

T

The `T` parameter value, as defined in the protocol IRP.

Misc. params

All parameters not called `D`, `S`, `F`, or `T`, using syntax *name* `=` *value*, for example `X=42 Y=73`. (Note that there are no spaces around the equal ("=") sign, but between the assignments.

Name

Name of the command. This is in principle arbitrary, however, to be able to constitute a meaningful export to a target device, the names have to be unique within the table.

Comment

A textual comment. In most export formats, it is transmitted to the export, and the export format can treat it anyway it desires.

5.2.2 The raw table columns

#

A unique number assigned to the signal when creating. It is not editable and not exported.

Timestamp

A timestamp generated when the signal was entered or created. It is not editable and not exported.

Intro

The intro sequence of the command.

Repeat

The repeat sequence of the command.

Ending

The ending sequence of the command.

Name

Name of the command. This is in principle arbitrary, however, to be able to constitute a meaningful export to a target device, the names have to be unique within the table.

Decode

The result of feeding the signal to the decoder. Not editable, not considered as authoritative information; serves only for information.

Analyze

The result of feeding the signal to the analyzer. Not editable, not considered as authoritative information; serves only for information.

Comment

A textual comment. In most export formats, it is transmitted to the export, and the target can treat it anyway it desires.

Frequency

Modulation frequency in Hz.

5.3 The "Render" 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. Numbers with a leading "0" are interpreted as octal numbers.

Earlier versions of this program (including documentation) used the word *generate* instead of *render*.

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

For protocols with a toggle, leaving it unassigned (`T` left empty) makes the rendering engine "toggle" in the sense that it changes its value on every invocation, in accordance with the IRP. The rendering engine is invoked not only by `Render`, but also by `Transmit`, `Export`, and the `To ...` buttons.

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

5.3.1 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:

1. Of course, there is the singleton set, just consisting of one value
2. There is also a possibility to give some arbitrary values, separated by commas. Actually, the commas even separate sets, in the sense of the current paragraph.
3. An interval, optionally with a stride different from 1, can be given, either as `min..max++increment` or `min:max++increment`, or alternatively, simply as *, which will get the min and max values from the parameter's parameter specs.

4. Also, a set can be given as `a:b<<c`, which has the following semantics: starting with `a`, this is shifted to the left by `c` bits, until `b` has been exceeded (reminding of the left-shift operator `<<` found in languages such as C).
5. Finally, `a:b#c` generates c pseudo random numbers between a and b (inclusive). The "pseudo random" numbers are completely deterministically determined from the seed. The parameters `a` and `b>` are optional. If left out, the values are taken as from the protocol parameters min and max respectively, just as with the `*` form.

5.4 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.

5.4.1 Tree importer

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.

5.4.2 Data bases

5.4.2.1 Global Caché's Control Tower data base

This is the new data base from Global Caché. Unfortunately, its Terms of Service are fairly restrictive. Also, non-premium users are not allowed to download codes using the API. For this reason, the program really only "browses" the data base.

To use its codes, log in with the browser (the `Web site` button goes to the home page), and have the code set mailed in CSV format (press `Send Code Set`). The mail received can be imported by the import text pane, raw subpane, Name col. = 1, Raw signal col = 2 (or 3), Field separator: comma.

5.4.2.2 Remotelocator

RemoteLocator is a program to locate and download infrared remotes. The `RemoteLocator` pane constitutes a GUI to RemoteLocator.

Currently, these four freely usable collections are supported:

IRDB
This collection was until recently available at the server `irdb.ik`, which is now defunct. The IRDB format is a very simple CSV format, containing command name, protocol and the device and subdevice parameters, but no meta information. Instead, information on the manufacturer and the device class are gleaned from the file path.

Lirc remotes
Lirc's collection of remotes, which are collected in the Sourceforge project. This format also contains no meta information. The manufacturer information is gleaned from the file path; however there are no information on device class, so all the Lirc remotes have device class "unknown" in the generated list.

GirrLib
Girr is a very versatile format for IR remotes. It is the native format for IrScrutinizer, and also supported by main program RMIR of the JP1 project. Meta information, including manufacturer and device class, is contained in the Girr file. There is a small collection (Girrlib), which is more of a proof-of-concept than a sizeable collection of remotes.

JP1
The JP1 project has a large collection of "device upgrades". These are summarized in an Excel file. This list also contains meta information such as manufacturer and device class. It can be read into the program, and used to browse the therein contained remotes (or rather, "device upgrades"). These can however not be directly loaded, but can with some manual work be translated to Girr files.

To use, first select `Select me to load` to load the list of the manufacturers. Then select, in order, a manufacturer, a device type, and a remote name. The field `kind` now indicates what kind of remote that was found. Pressing `Load` (if enabled) will load the remote into the tree importer, where it can be further manipulated.

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

5.4.3 File importers

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.

Several of the formats (the one based on text files, but not on XML (which carries its own character set declaration) allow the user to select the character set to be used for the

import to be selected. This option is found as `Options -> Import options -> Character Set...`

Most of the file based importers support dropping a compatible file from the GUI on the import window.

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 a tree importer.

The file based import formats allow a file to be "dropped" with the mouse on the main window.

5.4.3.1 Girr (the native format of IrScrutinizer)

The Girr format is the native format of IrScrutinizer. The importer is capable of importing directory hierarchies.

5.4.3.2 Lirc

The Lirc import feature is based upon Jirc, which is basically a subset of Lirc 0.9.0 translated to 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.)

Only Lirc remotes with timing information can be imported, not the so-called lircccode remotes. See this article for a more extensive discussion.

5.4.3.3 CML

The CML format is the format used by the RTI Integration Designer software. Many CML files are available in Internet, in particular on RemoteCentral. Particularly noteworthy is the "megalist" by Glackowitz. IrScrutinizer can import these files to its import tree, making every remote a nodes in the tree. Note that there is no need to unzip such a file; IrScrutinizer will unzip it on the fly.

5.4.3.4 Command Fusion

The native format that Command Fusion uses, file extension `.cfir`, can be imported here.

5.4.3.5 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.

5.4.3.6 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.

This is not extensively tested.

5.4.3.7 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.

5.4.3.8 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.

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.

Raw, line-based

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

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.

5.4.3.9 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.

5.4.3.10 IrTrans

IrTrans' configuration files (`.rem`) can be imported here.

5.5 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 rendered signals can be exported. Exports can be generated in a number of different formats. Some (Girr 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 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 file names of the exports 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 one of the Export buttons. 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 operating system. (Also available on the actions menu.) (Girr exports become a special treatment in order not to invoke another instance of the IrScrutinizer. They are copied to a temporary `.txt` file.) The `Open` button similarly opens the operating systems standard directory browser (Windows Explorer, Konquistador, Nautilus,...) on the export directory.

Some export formats (presently Wave, mode2, 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). When using these formats, the number of repetition sequences to include can be selected.

The character set used for the export can be selected through `Options -> Export options -> Character set...` Note however that this makes sense only for text based formats, including XML.

Some export formats have some more parameters, configured in sub panes. These will be discussed in the context of the containing formats.

(Most of) the formats presently implemented will be described next, in alphabetical order.

5.5.1 AnyMote

Generates a configuration file for the AnyMote IR remote control app for Android and iOS.

5.5.2 Arduino/Raw

Generates a complete C++ sketch for the Arduino. This uses one of the three Arduino Infrared libraries IRremote, IRLib, and Infrared4Arduino. As the name suggests, the raw form of the signals are used.

5.5.3 Arduino/Infrared4Arduino

Generates a similar C++ sketch for the Arduino as in the raw case, but tries to use the parametric form whenever possible, i.e., whenever the protocol is one that can be generated by the underlying infrared library. This version supports Infrared4Arduino only.

5.5.4 Arduino/IRremote

Like the preceding, but supports the IRremote library instead. (Currently, needs update to the current IRremote library.)

5.5.5 BracketedRaw

This export format generates an text file of the command(s) in the bracketed raw text format.

5.5.6 C

Generates a C code fragment consisting of declarations of the signals in raw- and CCF format. Intended mostly as an example.

5.5.7 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. Official documentation site. There are some extra parameters that can be configured in subpanes, described next:

5.5.7.1 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.

5.5.8 HTML

The purpose with the HTML export is to generate a browse-able page, not something for other IR-programs to feed upon.

5.5.9 ICT

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. Can be imported by IrScope and RemoteMaster.

5.5.10 irplus

Generates a configuration file for the Android ID remote irplus.

5.5.11 IrToy

A text version of the following

5.5.12 IrToy-bin

Generates a binary file "in IrToy format" that can be send to the IrToy using the *program* `irtoy[.exe]`, see this thread.

5.5.13 IrTrans

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

5.5.14 Lintronic

Simple text protocol for describing a single IrSequence.

5.5.15 Lirc Raw

The Lirc-exports are in `lircd.conf` - raw format. These use the raw Lirc format, except for a few well known protocol (presently NEC1 and RC5). They can be concatenated together and used as the Lirc server data base. Can also be used with WinLirc.

5.5.16 Lirc

Like `Lirc Raw`, but recognizes a large number of the protocols, for which lircd.conf files in "cooked" (opposite of raw) are generated. For other protocols, it falls back to the raw form.

5.5.17 mode2

A primitive debugging "signal format" used by Lirc, consisting on interleaved on- and off durations.

5.5.18 Pronto Classic

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

5.5.18.1 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.

5.5.19 Pronto Hex Oneshot

This export format takes a single signal, makes a sequence of it consisting of the intro sequence, an arbitrary number of copies of the repeat sequence, and the (in general empty) ending sequence. This is packed into a Pronto Hex format, having the said sequence as its intro, and empty repeat.

5.5.20 Spreadsheet

Simple tab separated value export format for importing in a spreadsheet program. This format is mainly meant to demonstrate a simple format in XSLT, more than a practically useful format.

5.5.21 Text

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

5.5.22 TV-B-Gone

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

5.5.23 Wave

IR sequences encoded as wave audio files.

5.5.23.1 The Wave sub-pane

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

5.6 The "Hardware" pane

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

A opposed to versions earlier than 2.4.0, there is exactly one *selected device*, corresponding to the the selected sub-pane of the `Hardware` pane. This device is, if capable, used for both sending and capturing. The selected device may be opened or not. An opened device may be selected or not. There may be more than opened device. The hardware can also be selected from the tool bar, Options -> Capturing hardware.

Note that by e.g. selecting non-existing hardware or such, there is a possibility encounter long delays, or even to "hang" the program.

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

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

5.6.1 Capturing parameters

Capturing an IrSequence is governed by the parameters `beginTimeout`, `captureMaxSize`, `endingTimeout`, which can be accessed in the `Options -> Timeouts` submenu. `beginTimeout` and `endingTimeout` bothe use the unit milli seconds. Unfortunately, not all hardware respect these parameters.

beginTimeout
When capturing starts, this determines how long the capturer is waiting for the first flash. Default is 3000ms.

captureMaxSize
This is the maximal duration the capturing hardware will accept. Default is 1000.

endingTimeout
The silence period required at the end of an IrSequence. Default is 300ms.

5.6.2 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 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).

For this to work, port 9131/udp must be open in a used firewall

The `Browse` button points the user's web 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.

5.6.3 /dev/lirc (Linux only)

On Linux, so-called mode-2 capable IR hardware using the `/dev/lirc` device, can be accessed directly, both for sending and receiving. When connected, and in some cases after loading suitable drivers, a device file `/dev/lirc` *n* (for *n* = 0, 1, 2,...) will be created. Of course, this must be read- and/or writeable by the current user; in e.g. Fedora this is the case for member of the group `lirc`.

After opening the device, its properties will be listed. See the man page lirc(4) for an explanation.

Although not a priori impossible, to my knowledge no `/dev/lirc` device implements frequency measurement.

5.6.4 The "Audio Port" Pane

As additional "hardware sending 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.

It can only send, not receive/capture.

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.

5.6.5 The "Girs Client" (previously "Arduino") Pane

Using this pane, a Girs server (e.g. running on an Arduino equipped with a suitable hardware) can be used to transmit and capture IR signals. The server can be connected to a serial port, or through Ethernet to the local area network, connected by a TCP socket. For the Arduino, the sketch `GirsLite` (or `Girs`) can be used.

The serial port may sometimes be finicky. Sometimes disconnecting and reconnecting the device may help.

To use the device connected to a real or virtual serial port, select the port in the combo box, pressing `Refresh` if necessary. `Open` the port thereafter. The opening of LAN connected device is simliar.

Normally, the Girs server uses the Girs `analyze` command to capture IR signals, which is normally deploying a non-demodulating sensor, providing a frequency measurement. If this is not desired (for example when the non-demodulating sensor is missing), selecting `Use receive for capture`, instead the `receive` command will be used, which normally uses a demodulating sensor. In this case, no frequency measurement will be produced, and instead the fallback frequency (selectable/changeable as `Options -> Fallback frequency`) will be used.

5.6.6 The "CommandFusion" Pane

Using this pane, the CommandFusion learner can be used to transmit and capture IR signals.

After connecting the Command Fusion learner to a USB port, select the the actual (virtual) serial port in the combo box, pressing `Refresh` if necessary. `Open` the port thereafter.

5.6.7 The "IrWidget" pane

Using this pane, the IrWidget can be used to capture IR signals.

After connecting the IrWidget to a USB port, select the the actual (virtual) serial port in the combo box, pressing `Refresh` if necessary. `Open` the port thereafter. If using the original Kevin Timmerman widget, for example built and sold by Tommy Tylor, the option `lower DTR abd RTS on opening` must be selected.

5.7 Key bindings and accelerators

The GUI contains a number of key bindings and accelerators. These can be found in menus and in the text of GUI elements. A more substantial description is planned, but not yet written.

6 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.

Girr files given as command line argument will be imported to the `parametric remote` table. Accordingly, if the system is configured to associate `*.girr` with IrScrutinizer, these files can be opened by double clicking them.

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 (which can also be accessed under the GUI as `Options -> verbose`, causing commands like sending to IR hardware printing some messages in the console.

Using the option `--properties` (or `-p`), an alternative property file can be used. If installing more than one version of the program, this is recommended practice.

The option `--nuke-properties` makes the program delete the property file, and exit immediately.

The option `--scaling` (or `-s`, `--scale`) set the scaling of the GUI. Accepted values and their semantics depend on the JVM used.

IrpTransmogrifier as well as HarcHardware as command line program can be invoked with the AppImage installation by calling it using the name `irptransmogrifier` or `harchardware` (through copying or linking).

The MacOS App does not support calling IrpTransmogrifier or HarcHardware; install the generic binary version if needed.

The program delivers well defined and sensible exit codes, listed in the code.

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

7 Adding new export formats

Only a few of the many export formats are defined in the main Java code (`Girr`, `Text`, `Pronto Classic` and `Wave`), the rest are defined in files in the directory `exportformats.d`, located in the root of the install directory. By adding a file here,

the user can simply add his/her own export formats according to own needs. An export format 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 these files, and is supposed to be read only when needed. Fundamental knowledge of XML and XSLT transformations are assumed.

7.1 Format of the files in exportformats.d

The file is an XML file supposed to be a valid instance of the XML Schema `exportformats.xsd`, although it is read without validation. The semantics is believed to be essentially self explaining, or clear from the examples already in there. An export format is packed in an element of type `exportformats: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 explicitly state a particular number of repetitions for an export.

The element `exportformats:exportformat`contains as a child element the XSLT transformation, which is an element of type `xsl:stylesheet`, with attributes as in the examples.

The task of the transformaton is to 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.

For testing and developing new export formats in this for, the `main` routine of the class `org.harctoolbox.irscrutinizer.exporter.DynamicRemoteSetExportFormatMai` may be used. It can be used to transform a Girr file from the command line.

After editing or adding export format files, they can be reloaded either by re-starting the program, or by selecting `Options` -> `Export formats database` -> `Reload`.

8 Properties

Under Windows, the persistent properties are stored in `%LOCALAPPDATA%\IrScrutinizer\IrScrutinizer.properties.xml` (typically `%HOME%`

`\AppData\Local\IrScrutinizer\IrScrutinizer.properties.xml`). Using other operating systems, it is stored according to the FreeDesktop specification. Per default, this is `$HOME/.config/IrScrutinizer/properties.xml`. It is not deleted by un-install or by an update of the program. If weird problems appear, for example after an update, try deleting this file, from the GUI `File -> Set properties to default`. There is also a command line option `--nuke-properties` that can be used to conveniently delete the present property file, without remembering its exact path.

9 Questions and Answers

9.1 How do I verify the integrity of my downloads?

On the official download site, the checksums (using the MD5, SHA-1, and SHA-512 algorithms) are available in the files `checksums.md5`, `checksums.sha1`, and `checksums.sha512`. With appropriate software, these can be used for validating the downloaded files.

To prevent from tampering of *both* the program files *and* the checksum files, the checksum files are also PGP signed with the cryptograph key for `Bengt Martensson <barf@bengt-martensson.de>`, finger print `5F492CB76BEB2E6B1CAA63A3EE1F17D4ECF8AF9C`. The full key can be downloaded here, or found in the program as `Tools -> Author's Public PGP key`. (For practical reasons, the shapshot releases are, in general, not signed.)

How to check these checksums and signatures differ between different programs and operating systems, and is therefore not described here. However, searching on the Internet will yield many descriptions.

9.2 My virus program says that your program contains virus or malware.

My programs do not come with virus. Downloading from un-official sites are discouraged, in particular if they do not keep the signed checksum files. (Most likely, they do not.)

First check the integrity of the downloaded files as described in the previous question. Assuming that the problem remains, this is a problem with your virus program, and not with the present program, and it should be handled as such. If it is a release version, please inform the author of the virus program, and/or create a white list. What you do not need to do, is to tell me, or "the world". Remember, it is a problem with the virus program, not with my software.

9.3 Why have RemoteMaster Import and Export been removed?

The RemoteMaster export and import found in earlier versions (up to 2.3.0) of this program have been removed. The reason is that the current version of that program

contains elaborate facilities for the import and export of Girr files — the native format for IrScrutinizer. As opposed to the previous import and export of the current program, they contain elaborate rules for transforming between protocols and the JP1 executors.

9.4 Can I run more than one instance simultaneously?

Yes, you can fire up several instances, for example, one for sending and another for receiving the sent signals. The only possible problem is that the properties are read from/written to the same file, so the instance that ends last will overwrite the properties of the first one. This does not have to be a problem. However, if this is an issue, you can start the program with the `--properties` *filename*, pointing to an alternative properties file. With Windows, you can duplicate the icon, and edit the command line on one of those.

9.5 I miss DecodeIR!

Support for DecodeIR was *removed*, not just hidden and deprecated, because keeping it would increase maintenance and installation effort substantially. For accessing DecodeIR, it is recommended to install version 1.4.3 (which is the final release with DecodeIR) in parallel to the current one, see the previous answer for details.

9.6 Firewall issues?

Except for the usual HTTP(S) ports, the program uses some TCP and UDP ports for communicating with some of the supported hardware. These are:

- For GlobalCache this is TCP port 4998. Its discovery beacon uses UDP port 9131.
- Lirc uses a TCP port, per default 8765.
- AGirs on an ethernet connected board uses a TCP port, per default 33333.

9.7 Does IrScrutinizer completely replaces IrMaster?

Almost. Using MakeHex as renderer (or more correctly, its Java version) 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.

9.8 How do I emulate the war dialer of IrMaster?

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 Render pane, using suitable parameter intervals (see this), 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 can be used for note taking.

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

9.9 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.

Check out JGirs, which is an IR server implementing the Girs specification. It can be considered as "IrScrutinizer as a server".

9.10 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.

9.11 What about the "fishy" icon?

It is a Babel fish, as found in The Hitchhiker's Guide to the Galaxy, having the property, that "... if you stick one in your ear, you can instantly understand anything said to you in any form of language". This symbolizes the program's ability to "understand" (read and write) a large number of different IR formats.

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

Try deleting the properties file. (Note the command line option `--nuke-properties` which will do exactly that, without having to manually find the file or its name.) If that does not help, try starting the program from the command line, which may leave hopefully understandable error message on the console.

9.13 (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.)

9.14 (Windows) Why is the program so slow at start-up?

Normal start up time is a few seconds, about the same time as it takes for (e.g.) a web browser to start. If it takes considerably longer than that, the problem is almost surely hardware related, i.e., the program looks for possibly non-existent hardware. To fix this, make sure that the selected capture- and sending devices (these are save between sessions!) do not correspond to something non-existent. The most "innocent" settings are: capture: Lirc mode 2, and sending: Audio port. These are also the defaults when the program starts up for the first time. Also, consider deleting possible "junk devices" in the Windows device manager under `COM & LPT` (unused Bluetooth-to-serial ports etc.).

9.15 (Linux) I get error messages that lock files cannot be created, and then the serial hardware do not work.

This answer needs updating.

When starting IrScrutinizer, or by pressing the `Refresh` button, error messages occur like

```
check_group_uucp(): error testing lock file creation Error details:Permission deniedcheck_lock_status: No permission to create lock file.
```

(and a number of them...) The problem is that the library `rxtx` (like some other program accessing a serial interface) wants to create a lock file in a particular directory. This is/was traditionally `/var/lock`, which is often a symbolic link to `/var/run/lock`. This directory is normally writable by members of the group `lock`. So your user-account should probably be a member of that group. (How to perform this is different in different Linux distributions.) The rxtx library delivered with IrScrutinizer expects the lock directory to be `/var/lock`. However, recently some Linux-distributions (e.g. Fedora 20), instead are using `/var/lock/lockdev` as its lock directory (while `/var/lock` still is a link to `/var/run/lock`. To solve this, I recommend, if possible, installing rxtx provided by the Linux distribution used, i.e. not using the one with IrScrutinizer. For example, on Fedora, the command is

```
sudo dnf install rxtx
```

which installs the library in `/usr/lib/rxtx` or `/usr/lib64/rxtx`, depending on the operating system. (Other distributions uses other commands, for example `apt-get` on Debian-like systems.) Finally, the correct installation directory of the library (`librxtxSerial-2.2pre1.so`) has to be given to the JVM running IrScrutinizer. For this, see the wrapper `irscrutinizer.sh` and the comments therein, and make the necessary adaptations.

9.16 What is on the splash screen?

From left to right, a Global Caché iTach Flex, an IrToy, and a low-cost clone of an Arduino Nano, the latter equipped with a non-demodulating IR detector (TSMP4138) for capturing and an IR diode (SFH415) for sending. These are all hardware which work well with IrScrutinizer, both for sending and capturing.

9.17 Why do I get multiple "Really exit?" confirmation questions when exiting?

If the "Scrutinize Remote" parametric table is non-empty and not saved, the user is asked to acknowledge the exit, since he/she may otherwise loose carefully entered data. Exactly the same thing goes for the raw table. So, it is not "multiple", it may be at most two; they are not identical (but very similar). There is also a "Do not ask this again" checkbox.

10 Appendix. Building from sources

"IrScrutinizer" is one subproject within harctoolbox.org. It depends on several other subprojects within harctoolbox. The repository IrScrutinizer consists of this subproject.

The released versions are found on the download page. The development sources are maintained on my GitHub repository. Forking and pull requests are welcome!

I go to great lengths ensuring that the program runs equally well on all supported platforms. I do not care too much that all aspects of the build runs equally well on all platforms. I build with Linux (Fedora), the continuous integration build runs on GitHub Actions (deploying Ubuntu). Other platforms are treated step-motherly.

10.1 Dependencies

As any program of substantial size, IrScrutinizer uses a number of third-party components. All of these are also free software, carrying compatible licenses. The dependent packages need to be installed also in the host-local Maven repository in order for the build to work. In some cases (basically the ones with a version not ending with `-SNAPSHOT`), the packages are uploaded to the Maven central repository, and will be automatically downloaded to the local host by a Maven invocation.

There are some scripts to aid downloading and building, described next. It is assumed that git and Maven are installed and available as commands `git` and `mvn` respectively.

Of course, it is also possible to manually download or clone these packages from my Github repositories, then build and install them locally by `mvn install`.

10.1.1 IrpTransmogrifier, Girr, HarcHardware, Jirc

These are all Java packages that are required to build IrScrutinizer. HarcHardware requires nrjavaserial. They can be downloaded and built by the script `common/scripts/build-harctoolbox-project.sh`, using an argument of

`IrpTransmogrifier`, `Girr`, `HarcHardwar`, or `Jirc`. See the file `.github/workflows/maven.yml` for the complete commands.

10.1.2 DevSlashLirc

This library is used to access `/dev/lirc`-hardware. It is used by the Linux version only. It is a Java JNI library, written in Java and C++. It is written by myself, and available here.

The subdirectories `native/Linux-amd64`, `native/Linux-i386`, and `native/Linux-arm` contain compiled versions for the x86_64, x86_32, and ARM processors respectively.

The package can be downloaded, and the Java part built, by the script `common/scripts/build-harctoolbox-project.sh` using the argument `DevSlashLirc` (see `.travis.yml` for an example).

10.1.3 nrjavaserial

This is a fork of the legacy RXTX library for serial communication with Java. Version 5.2.1 is currently used.

10.1.4 JCommander

Normally, this component is downloaded and installed automatically by Maven.

10.1.5 Tonto

Tonto can be downloaded and installed by the script `common/scripts/build-tonto.sh`. It requires the Apache ant build system to be installed as the command `ant`. Note that the shared library `libjnijcomm`, which is required by the Tonto program for communicating with a Pronto remote through a serial interface, is not required for use with IrScrutinizer, and can therefore be left out. Using the option `-n` to the script (see `.travis.yml` for an example), the script will not try to build and install the shared library.

10.2 Building

The Maven "software project management and comprehension tool" is used as building system. Modern IDEs like Netbeans and Eclips integrate Maven, so build etc can be initiated from the IDE. Of course, the shell command `mvn install` can also be used. It creates some artifacts which can be used to run IrScrutinizer in the `IrScrutinizer/target` directory.

Two additional shell tools are needed. These are:

- The `unix2dos` and `dos2unix` utilities, typically in the `dos2unix` package.
- The `icotool` utility, typically in the `icoutils` package.

10.3 Windows setup.exe creation

For building the Windows setup.exe, the Inno Installer version 6 is needed. To build the Windows `setup.exe` file, preferably the work area should be mounted on a Windows computer. Then, on the Windows computer, open the generated file `IrScrutinizer/target/IrScrutinizer_inno.iss` with the Inno installer, and start the compile. This will generate the desired file `IrScutinizer-` *version* `.exe`.

Alternatively, the "compatibility layer capable of running Windows applications" software application Wine (included in most Linux distributions) can run the ISCC compiler of Inno. The Maven file `IrScrutinizer/pom.xml` contains an invocation along these lines, conditional upon the existence of the file `../Inno Setup 6/ISCC.exe`.

10.4 Mac OS X app creation

The Maven build creates a file `IrScrutinizer-` *version* `-macOS.dmg`. This file can be directly distributed to the users, to be installed according to these instructions.

The icon file `IrScrutinizer.icns` was produced from the Crystal-Clear icon `babelfish.png` in 128x128 resolution, using the procedure described here.

10.5 AppImage creation

To build the x86_64 AppImage, define `bundledjdk_url_sans_file` and `bundledjdk` in `pom.xml` to point to a suitable JDK distrubution file. If a file with name given by `bundledjdk` is not present in the top level directory, it can be downloaded by the script `tools/get-jdk-tar.sh`. Then the maven goal `make-appimage` (which will be invoked during a normal build) will build an appimage for x86_64.

10.6 Test invocation

For testing purposes, the programs can be invoked from their different target directories. IrScrutinizer can be invoked as

```
$ java -jar target/IrScrutinizer-version-jar-with-dependencies.jar
```

or, if the shared libraries are required, with *path-to-shared-libraries* denoting the path to a directory containing the shared libraries.

```
$ java -Djava.library.path=path-to-shared-libraries -jar target/
IrScrutinizer-version-jar-with-dependencies.jar
```

IrScrutinizer can also be started by double clicking the mentioned jar file, provided that the desktop has been configured to start executable jar with a Java virtual machine.

10.7 Installation

Maven does not support something like `make install` for deployment installing a recently build program on the local host. Instead, the supplied `tools/Makefile` can install the program to normal Linux locations (in the Makefile `INSTALLDIR`),

```
sudo make -f tools/Makefile install
```

Equivalently, the just created generic-binary package `IrScrutinizer/target/IrScrutinizer-*-bin.zip`) can be installed using these instructions.