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.

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.

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, Command Fusion, and Arduino), but also from a number of different file formats (among others: Lirc, Wave, CML, 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 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 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 three 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, 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 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".

Sources are hosted on Github. Bugreports are enhancement requests are welcome (e.g. as issues), as is contributions (code, 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, 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. The final version was released in Febrary 2014, 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, DecodeIR, and Jirc. ExchangeIR was written by Graham Dixon and published under GPL3 license. A subset has been translated to Java by Bengt Martensson. DecodeIR was originally written by John S. Fine, with later contributions from others. It is public domain software. 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 "Clarified Artistic License".

The contained Arduino firmware contains code copyrighted by Michael Dreher (IrWidget, GPL2 or later). Some code within that firmware is derived from IRremote, which was originally written by Ken Shirriff, and is licensed under the LGPL 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 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 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 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 every computer with a modern Java installed; Windows, Linux (including Raspberry Pi), Macintosh, etc. Java 8 or later is required. The 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, `.jnilib` or `.dylib` in Mac OS X). For Linux, there is also the *DevSlashLirc* library. If DecodeIR, RXTX, or DevSlashLirc 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 DecodeIR-related functions, 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.

As of version 1.3, there are six different way of installing the program, described next.

3.3 Windows

First make sure that the current Java is installed.

Download the Window setup file and double click it. Select any installation directory you like; suggested is `C:\Program Files\IrScrutinizer`, unless you can not install with administrator rights. Unless reason to do so, create the start menu folder, and the desktop icon. If administrator rights are present, the recommended installation location is in a subdirectory of `Program Files`. If not, installing is still possibly, using any installation directory writeable by the current user. IrScrutinizer can now be started from `Start -> IrScrutinizer -> IrScrutinizer`, or from the desktop icon.

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 IrpMaster, which can be called as `irpmaster` from a so-called DOS box.

To uninstall, select the uninstall option from the Start menu. 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 application that is downloaded, made executable, and run, without any installation in classic sense. At this moment, only 64-bit x86 Linux is supported, but it is believed to run on all modern, 64-bit x86 Linux distrubutions. Just download IrScrutinizer-x86_64.AppImage, make executable, and double click (alternativelly, start from the command line). Possibly contrary to the AppImage philosophy, this appimage does not come with its own Java, but requires it to be installed separatly. (Otherwise it would have been 10 times larger.)

3.4.1 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`, 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.

Do not run this program as root.

3.5 Mac OS X app

As opposed to Linux and Windows, the "JDK" (Java Development Kit) needs to be installed on MacOS X, not just "Java" (JRE).

Download the compressed app. Uncompress it by double clicking. Opening it will show an app, and a few documention 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.

To unistall, just move the app to the trash.

The command line program IrpMaster is 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 Mac. It can be used on all those systems. (In other environments, the shared libraries may be compiled with moderate effort.)

The generic binary distribution can be used whenever using the rpm/setup.exe/app installation is not possible, not up-to-date, or not desired.

First make sure that the current Java is installed.

To install, unpack in an empty directory of your choice, suggested is `/usr/local/irscrutinizer`. 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/irscrutinizer/irscrutinizer.sh /usr/local/bin/irscrutinizer
ln -s /usr/local/irscrutinizer/irscrutinizer.sh /usr/local/bin/irpmaster
```

If your system natively supports the RXTX, you should preferably use that. See the comments in the wrapper `irscrutinizer.sh`.

The JNI libraries `libDecodeIR.so`/`DecodeIR.dll`/`libDecodeIR.jnilib` are contained in the distribution and should be found by the program in the installed location.

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 program IrpMaster can be started by the command `irpmaster`. It is also possible to start the program 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. 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, `*.girr` files can be opened in IrScrutinizer by double click. For this, follow this description.

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

3.7 Fedora Linux

Fedora rpm packages are available for the now obsolete version 1.1.3.

To install, the command `sudo dnf install harctoolbox harctoolbox-doc` should be issued in a terminal window. This will install IrScrutinizer as a desktop program, as well as the command line commands `irscrutinizer` and `irpmaster`. To uninstall, use the command `sudo dnf erase harctoolbox harctoolbox-doc`.

Unfortunatelly, the packaging contains a few semi-serious problems. These are reported, but stay unfixed (since over a year). Fedora users are recommended instead to use AppImage or the generic binary package.

3.8 Source distribution

Compiling the sources is covered in the Appendix. This allows to install the different components in a way compliant with e.g. the installation by the Gnu Autotools, normally in subdirectories of `/usr/local`.

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.

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", "Sending Hardware", and "Capturing 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 a few of the many export formats are defined in the main Java code, 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.

6.1 Format of the files in exportformats.d

The file is an XML file read without validation. The syntax and semantics are believed to be essentially self explaining, or clear from the examples already 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 or adding export format files, they can be reloaded either by re-starting the program, or by selecting Options -> Export formats database -> Reload.

7 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 according to the FreeDesktop specification. Per default, this is `$HOME/.config/IrScrutinizer/properties.xml`. It is not deleted by un-install. (If weird problems appear, for example after an update, try deleting this file. For this, there is a command line option `--nuke-properties` that can be used to conveniently delete the present property file, without knowing its exact name.)

8 GUI Elements walk through

8.1 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. The button "Capt. (cont.)" a tread is started, that continously captures signals from the hardware. Only the last signal is kept. For text import, the signal can be in either Pronto Hex 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.

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 comparing with subsequent plots.

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.

8.2 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.

Clicking a row with the middle mouse button selects that row, and transmits it using the currently selected sending hardware.

To capture 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.

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.

8.3 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.

8.3.1 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).

8.4 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.

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 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.

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

8.4.1 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, first select `Select me to load` to load the list of the manufacturers. Then select, in order, a manufacturer, a device type, and a setup code, the latter possibly by trial-and-error.

This data base is no longer maintained; replaced by the "Control Tower data base", described next.

8.4.2 Global Caché's Control Tower data base

This is the new data base from Global Caché. Unfortunately, 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.

8.4.3 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, first select `Select me to load` to load the list of the manufacturers. Then 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.

8.4.4 Girr (the native format of IrScrutinizer)

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

8.4.5 Lirc

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

8.4.6 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.

8.4.7 Command Fusion

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

8.4.8 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.

8.4.9 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.

8.4.10 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.

8.4.11 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.

8.4.12 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.

8.4.12.1 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.

8.4.12.2 Raw, line-based

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

8.4.12.3 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.

8.4.13 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.

8.4.14 IrTrans

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

8.5 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 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 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.

The formats presently implemented will be described next, in alphabetical order.

8.5.1 AnyMote

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

8.5.2 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.

8.5.3 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.

8.5.4 Arduino/IRremote

Like the preceding, but supports the IRremote library instead.

8.5.5 C

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

8.5.6 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. There are some extra parameters that can be configured in subpanes, described next:

8.5.6.1 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.

8.5.6.2 The sendir sub-pane

The Global Caché sendir format is optionally included within the Girr and the text format. It requires a module number and a connector number. Also, it is possible to generate the compressed form of sendir. This can be enabled by selecting the `compressed` check-box.

8.5.7 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.

8.5.8 irplus

Generates a configuration file for the Android ID remote irplus.

8.5.9 IrToy

A text version of the following

8.5.10 IrToy-bin

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

8.5.11 IrTrans

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

8.5.12 Lintronic

Simple text protocol for describing a single IrSequence.

8.5.13 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.

8.5.14 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.

8.5.15 mode2

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

8.5.16 Pronto Classic

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

8.5.16.1 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.

8.5.17 RemoteMaster

Produce a rudimentary device update (`.rmdu` file) for RemoteMaster. After importing, RemoteMaster will have the parameters, but not an executor, nor button assignments. These have to be entered manually in RemoteMaster. The procedure is described in detail here.

8.5.18 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.

8.5.19 Text

The text format is essentially the Girr format stripped of the XML markup information. The sendir sub-pane is used for selecting the parameters of the optionally contained sendir format.

8.5.20 TV-B-Gone

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

8.5.21 Wave

IR sequences encoded as wave audio files.

8.5.21.1 The Wave sub-pane

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

8.6 The "Sending HW" pane

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

Note that there is exactly one *selected device*, corresponding to the the selected sub-pane of the "Sending HW" pane. This device may be opened or not. An opened device may be selected or not. There may be more than opened device.

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.

8.6.1 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 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.

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).

8.6.2 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, sources 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.

When selecting the Lirc sub-pane, the therein selected Lircd is inquired for its version and its remotes. If successful, the "Remote" and "Command" combo boxes should now be enabled. After selecting a remote and one of its command, it can be sent to the Lirc server by pressing the "Send" button.

After entering another IP-Address or name, and port (stay with 8765 unless a reason to do otherwise), pressing the "Reload" button will update the Lirc server with its known remotes and their commands (however, not the version).

Transmitting other signals to Lirc works only if the Lirc server has the above described patch applied.

8.6.3 /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.

8.6.4 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.)

The IRTrans module is then accessed using the UDP text mode.

8.6.5 The "IrToy" Pane

Using this pane, the IrToy (version 2) can be used to transmit IR signals. To my knowledge, only the firmware version 2.2.2 works.

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

8.6.6 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.

8.6.7 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.

8.6.8 The "Audio" 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.

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.

8.7 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. In many cases, one piece of hardware can be used both for sending and capturing. In these cases, it is in generaly opened and closed under the "Sending hw" pane, but still needs to be selected for capturing.

It is unfortunately not possible to have e.g. one IrToy for sending and *another one* for capturing.

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.

8.7.1 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 currently available serial ports. Select the correct one. Press "Open". which should now remain "pressed". The port can be closed again by another 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.

8.7.2 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.

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

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

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

8.7.3 Lirc Mode2

`mode2` is a program from the Lirc distribution, that prints timing information in a simple text format to standard output. (Note that in Lirc terminology, there are really three meanings of the term "mode2".) IrScrutinizer can read a stream in this format, either on its standard input, or from a program run in an internal subprocess. This way, effectively any Lirc mode2 driver can be used to capture IR signals for Lirc (assuming "normal" modulation frequency). This makes IrScrutinizer a viable replacement of the `irrecord` program.

8.7.3.1 Standard input

To use with standard input, the program can be started for example with a command like

```
$ mode2 --driver default | irscrutinizer &
```

As capturing device "Lirc Mode 2" and the subpane "Standard input" is selected.

8.7.3.2 Command in sub-process

In theory, any program that prints information in the mode2 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.

8.7.4 /dev/lirc (Linux only)

The device should be opened on the "Sending hw" pane, and if desired, selected here for capturing.

8.7.5 Girs client (previously Arduino)

The device should be opened on the "Sending hw" pane, and if desired, selected here for capturing.

8.7.6 IrToy

The device should be opened on the "Sending hw" pane, and if desired, selected here for capturing.

8.7.7 Command Fusion Learner

The device should be opened on the "Sending hw" pane, and if desired, selected here for capturing.

9 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, causing commands like sending to IR hardware printing some messages in the console.

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

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.

IrpMaster as command line program can be invoked with the AppImage installation by giving `--irpmaster` as the *first* argument to the program; all of the rest will be handled over to IrpMaster. However, the `--config` is not needed, it is generated automatically.

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

The program delivers well defined and sensible exit codes.

10 Questions and Answers

10.1 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.

10.2 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 Generate 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, or the "verified" check-box, can be used for note taking.

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

10.3 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.

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

10.4 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.

10.5 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" a large number of different IR formats.

10.6 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.

10.7 (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.)

10.8 (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-existant hardware. To fix this, make sure that the selected capture- and sending devices (these are save between sessions!) do not correspond to something non-existant. 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.).

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

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.

10.10 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.

10.11 Do you solicit or accept donations?

No.

11 Appendix. Building from sources

"IrScrutinizer" is one subproject (corresponding to a Java *package*) within harctoolbox.org. It depends on several other subprojects within harctoolbox. The project `harctoolboxbundle` consists of these subproject bundled together, with some dependent third-party components added.

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

11.1 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 maven in order for the build to work. With the dependencies available, the script `tools/install-deps.sh` can be used to install them in the local maven repository before building.

11.1.1 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.

11.1.2 The Crystal icons

A subset of the Crystal icons are included, and will be included in a built jar. Most Linux distributions contain these too, so a Linux packaging may like to use the system icons instead.

11.1.3 RXTX

The serial communication packate RXTX is also included in the source package.
This builds a shared library and a jar file. If there is a system supported RXTX (`librxtxSerial` really), it should be preferred. The distribution constains pre-compiled binaries for Linux, Windows, and Mac OS X, both in 32- and 64-bit versions. To compile the C sources, see the sub-directory `rxtx-pre2h` and the instructions therein.

Note that the system supplied RXTX jar on many system (e.g. Fedora 21) has some issues (version number incompatible with the shared library, does not recognize the `/dev/ttyACM*`-ports required by IrToy and many Arduinos, unflexible library loading), so using our RXTX jar together with the system supplied shared library can be sensible.

11.1.4 DecodeIR

If the system supports DecodeIR, use the system version. On recent Fedora, this can be installed with the command `sudo dnf install DecodeIR`. This will install both the shared library `libDecodeIR` as well as the jar file `DecodeIrCaller.jar`.
To download and compile the sources, see (or execute) the script `tools/build-decodeir.sh`.

11.1.5 ExchangeIR

It the system supports ExchangeIR, use the system version. (On recent Fedora, use `sudo dnf install ExchangeIR`.) Otherwise, it can be downloaded and installed by the script `tools/build-exchangeir.sh`.

11.1.6 minimal-json

It the system supports minimal-json, use the system version. (On recent Fedora, use `sudo dnf install minimal-json`.) Otherwise, it can be downloaded and installed by the script `tools/build-minimal-json.sh`.

11.1.7 jcommander

It the system supports jcommander, use the system version. (On recent Fedora, use `sudo dnf install beust-jcommander`.) Otherwise, it can be downloaded and installed by the script `tools/build-jcommander.sh`.

11.1.8 Tonto

It the system support Tonto, use the system version. (On recent Fedora, use `sudo dnf install tonto`.) Otherwise, it can be downloaded and installed by the script `tools/build-tonto.sh`.

Note that the shared library `libjnijcomm`, which is required by the program Tonto for communicating with a Pronto remote through a serial interface, is not required for use with IrScrutinizer, and can therefore be left out.

11.2 Building

As of version 1.1.2, 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.

It also creates a `package/dist` directory containing jars (without dependencies), docs, and configurations files. This is intended to support packaging.

To prepare the Windows version, some shell tools are needed. These are:

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

11.3 Windows setup.exe creation

For building the Windows setup.exe, the Inno Installer version 5 is needed. To build the Windows `setup.exe` file, preferrably 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 experimental invocation along these lines.

11.4 Mac OS X app creation

The Maven build creates a file `IrScrutinizer-`*`version`*`-app.zip`. 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.

11.5 Test invocation

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

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

and IrpMaster as

```
$ cd IrpMaster
    $ java -jar target/IrpMaster-jar-with-dependencies.jar [arguments...]
```

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

11.6 Installation

For reasons unclear to me, Maven does not support something like `make install` for installing a recently build program on the local host. Instead, the supplied script `tools/install-irscrutinizer.sh` installs the program to normal Linux/autotools locations. (Actually, invoking Maven with the "deploy" target will invoke this script too.) Alternativelly, the just created generic-binary package (`IrScrutinizer/target/IrScrutinizer-bin.zip`) can be installed using these instuctions.

12 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 Remote Control.