LORRIS TOOLBOX

Microprocessor Application Development Kit

http://tasssadar.github.io/Lorris

Author: Vojtěch Boček
vbocek@gmail.com
http://tasssadar.github.io
School: Secondary School of Engineering
Sokolská 1, Brno, Czech Republic

Introduction

Lorris is an extensive software package of tools which all share the same goal – to help with development, debugging and control of electronic devices, mainly robots. This chapter briefly describes the most important features of all modules and each module is throughly described in its own chapter further on.

Module: analyzer

- Its main purpose is to graphically display data from the device.
- Analyzer uses widgets to display data – small "windows", each showing certain part of data.
- Each widget has individual settings and the user can place them anywhere on the workspace.
- Thanks to widgets, it is possible to assemble interface suitable for virtually any device.
- Several types of widgets are available in Lorris, for example *Number, Color, Column bar, Circle* (displaying angle within circle) or *Graph.*
- Analyzer is also ideal for easy displaying of data from components for which using only numbers is not eligible, e.g. color sensor.
- Some widgets can also send data to the device. Consequently, it means that beside displaying data, widgets can also control the device.
- Of all the widget types, *Script* is the most notable one. The user writes his own script, which processes the incoming data. User's script can use other widgets and other parts of Lorris, which means that it can display or in other ways interpret virtually any data.
- Using script, the user can modify the behavior of Lorris itself.

Module: programmer

- Graphical interface for several types of bootloaders and programmers of microchips.
- It can write program to chip, read and erase chip's memory or program chip's fuses.
- Official GUI for Shupito programmer.
- Shupito is microchip programmer. One end goes to computer, the other one to the chip – you need programmer to program some types of chips.

Module: terminal

- Regular terminal – displays incoming data either as text or as hexadecimal byte dump.

Module: proxy between serial port and TCP socket

- Proxy creates server connected to serial port. The serial port is then accessible from anywhere via the internet.
- It makes it possible to debug, control or otherwise communicate with the device via internet network.

1 Motivation

I'm a member of one of the teams which build robots for various competitions, and I've met a problem while we were building one of our robots – such robot usually contains a pretty large number of various sensors (ultrasound range meters, encoders for measuring covered distance, buttons which detect collision with borders of the game field ...), and there was no way to show data from those sensors comfortably and clearly.

In order to simplify and speed-up development of the robot, I decided to find some computer application which would show data from the robot in clear and well-arranged manner. Requirements which I set for this application are specified in chapter 1.1.

1.1 Requirements for application

I require following features from the application:

1. Ability to process data from device and show them clearly
2. Support for many formats of incoming data
3. Quick and simple to use
4. Support for other operating systems than MS Windows
5. Low price

6. Ability to easily expand program, ideally open-source
7. No dependencies on other applications (eg. MS Office Excel)

I've decided to write my own program which will meet all the requirements because no such application exists.

2 Lorris

Lorris is a program written in C++ with use of Qt Framework. Qt is multiplatform framework, which (among other things) makes it possible to run Lorris on multiple operating systems – I'm using Debian Linux (Wheezy, 64bit) and Windows 7 for testing.

2.1 Website and repository

Lorris' GIT[1] repository is hosted on GitHub.com. GitHub also provides hosting for project's website, which contains links to prebuilt Lorris binaries for Windows, description of program, video introduction to Lorris (6 min.), screenshots of Lorris and information how to build Lorris under MS Windows and Linux.

- Repository: https://github.com/Tasssadar/Lorris
- Website (English):
  http://tasssadar.github.com/Lorris/index.html

There is still an ongoing development in application's repository.

2.2 Application's structure

The program is designed as modular application, so that it can accommodate several parts which, although they are separate, share the same area of use. Base part of application provides connection to device (e.g. to robot or to development board with chip), tab-based user interface and storage for application settings, but data processing itself takes place in individual modules.

Modules are opened as tabs, much like pages in web browser. Lorris can open several windows at once and it can split each window to multiple parts.

Connection options:

- Serial port
- TCP socket[2]
- Loading data from file

It is possible to connect multiple modules to one device.

[1] *GIT* – distributed version control system
[2] *Transmission Control Protocol* – connection via internet.

2.3 Sessions

Lorris can save everything user opened (tabs, their layout, connection, data of each tab, ...) as session. User can later load saved session and thus return to his previous work. Lorris automatically saves session before it is closed, so when user starts Lorris again, all his work is in the same state as it was before he left.

2.4 Automatic updates

Lorris can update itself under MS Windows. It checks for new version on start, and if there is one available, it shows little notification. In case user confirms the update, Lorris closes itself and runs little updater application. Updater shows changelog and downloads new version and installs it.

3 Module: Analyzer

This module parses incoming data (structured as packets) and displays them in graphical widgets. Application saves processed data into memory – user can go through received packets using slider and textbox in upper part of the window. All data (received packets, packet structure and widgets positions and settings) can be saved to file.

Packet structure is configured in dialog window. It is possible to set packet's length, endianness[3], packet's header and its content – static data ("start byte"), dynamic lenght od packet and command and device ID. Packets can be later filtered by command or device ID.

Incoming data show up in upper part of the window when packet structure is set and user can then "drag" widgets from the list in right part of the window to workspace. Data are assigned to widget again using drag&drop, this time user has to drag first byte of data to widget.

Widget then displays data from that byte (or several bytes if needed). Assigned byte is highlighted when user puts mouse over the widget, so that he can find out which data belong to which widget.

Widget settings are available in context menu under right-click. User can set title and other parameters different for each widgets – these parameters will be described in each widget's section later. Widgets can also be locked, which means the widget can't be closed nor moved or resized.

It is possible to precisely position widgets using grid or by using "alignment lines". User can also easily clone widgets by moving them while holding the control key.

Some widgets might profit from following feature: if user grasps widget with mouse as if he wanted to move it and then "shakes it" from right to left, the widget will expand itself to cover all of the visible workspace. When it is moved, it will shrink to it's original size.

[3] *Endianness* – order of bytes in numbers

3.1 Filters

Analyzer can filter incoming data and each filter may contain several conditions, which determine if packet is filtered out or not. Each condition can check command or device ID from packet's header, value of byte in packet or it can run simple user script. Thanks to the script, it is possible to write almost any kind of condition.

3.2 Widget: number

This widget displays integers (both signed and unsigned, 8 to 64bits) and decimal numbers (single-precision[4], 32 and 64 bit). Widget can align the number to max length of its data type and format as follows:

- Decimal – number as base 10
- Decimal with exponent – uses exponent to display big numbers, available only for decimal numbers
- Hexadecimal – number as base 16, available only for unsigned numbers
- Binary – number as base 2, available only for unsigned numbers

Another feature is option to recalculate widget's value using formula specified by the user. This is useful for example while showing data from infrared range finders, because their output value must be converted to centimeters using equasion. Formula can look like this:

```
2914/(%n+5)-1
```

where `%n` is alias for number which would otherwise be displayed in the widget. This particular formula converts distance measured by Sharp GP2Y0A41 infrared range finder to centimeters.

3.3 Widget: bar

Data in this widget are displayed as bar. User can set data type (same as widget *number*), orientation (vertical or horizontal) and range of displayed values. It can also use formula to re-calculate its value in the same way as widget *number*.

3.4 Widget: color

This widget shows incoming data as colored rectangle. Supported color formats:

- **RGB** (8b/channel, 3x uint8)
- **RGB** (10b/channel, 3x uint16)

[4]Standard floating-point number format used in C and other languages (IEEE 754-2008)

- **RGB** (10b/channel, 1x uint32)
- **Shades of gray** (8b/channel, 1x uint8)
- **Shades of gray** (10b/channel, 1x uint16)

Widget supports brightness correction for all colors at once or for each color of RGB space separately.

3.5 Widget: graph

This widget shows data in graph – order of the data is on the x axis and data values on the y axis. User can set name, color and data type of each graph curve and automatic scrolling, sample size and scale for graph. Graph also has legend which shows curve's names and colors, and curves can be hidden by clicking at their names in legend. Scale of each axis can be changed by scrolling the mouse wheel while hovering the cursor above axis. If the mouse is above graph area, mouse wheel changes scale of both axes at once.

3.6 Widget: script

This widget uses user-written script to process data. Script can be written in Python or QtScript (language based on ECMAScript[5], same as JavaScript[6], which means JavaScript and QtScript are very similar).

Script can process incoming data, react to key presses and send data to device. Basic output can be displayed in terminal, but it is also possible to use other widget types to show data (number, bar, ...).

Script editor has built-in code samples, for example how to set value of existing *number* widget, how to send data to device or how to react to key presses. Editor also has link to automatically generated documentation, which is available on http://technika.junior.cz/docs/Lorris/.

3.7 Widget: circle

Widget *circle* shows incoming data as angle in circle, which is useful for example when displaying rotation of robot's wheel. Incoming data can be in degrees, radians or just number in certain range (eg. data from 12bit encoder in range from 0 to 4095).

3.8 Widgets button and slider

These two widgets are used for interaction with script – callback method in script is invoked on button click. In this method user can for example send a command to robot. Similarly, callback method is invoked after moving slider, so that user can for example change robot's movement speed. Keyboard shortcut can be assigned to button "click" action and for slider to gain focus, so that user can move it using arrow keys.

[5] *ECMAScript* – scripting language accoring to stadard ECMA-262 and ISO/IEC 16262

[6] *JavaScript* – scripting languge used primarily on web

3.9 Widget: status

Status is designed to show state of for example button (pressed/released) or error status from encoder (0 = okay, other values are error codes). User assigns states to incoming values (state consists of text and it's color) and widget then shows active states. It supports "Unknown value", which is shown when incoming data don't match any defined status.

3.10 Other widgets

Analyzer in Lorris includes several more widgets, for example terminal or canvas, unfortunately I can't describe them in detail because of limited space.

4 Module: programmer

This module acts as graphical interface for several types of programmers and bootloaders. The interface has two modes – full and minimal. Full interface contains all buttons and settings for programming all memories of the chip, minimal interface contains only button which flashes main memory and button to stop chip. Minimal interface is convenient when using the split feature, because it uses only a small amount of space.

4.1 Shupito programmer

Shupito is microchip programmer created by Martin Vejnár. It can program microcontrollers using ISP[7], PDI[8] and JTAG[9] interfaces.

Module programmer in Lorris is official interface for Shupito programmer. Most of Shupito communication is written by Martin Vejnár.

4.1.1 UART tunnel

Shupito can create tunnel[10] for UART interface from programmed chip to computer. Lorris can use this feature – active tunnel creates new virtual connection and other modules can connect to it.

4.2 Bootloader avr232boot

Author of this bootloader is also Martin Vejnár. Avr232boot supports only Atmel ATmega chips and it is inspired by reference bootloader code for these chips, but it is designed to be as small as possible. Originally, it could only program flash memory of the chip (the one where program is stored), I added support for programming and reading of EEPROM[11]

[7] *In-system programming* – interface capable of programming chips directly on their PCB

[8] *Program and Debug Interface* – interface by company Atmel with features similar to ISP

[9] *Joint Test Action Group* – interface standard IEEE 1149.1 which can be used to program and debug chips

[10] Direct connection between programmed chip and the computer via programmer

[11] Flash memory which keeps data even without electricity. It is used to store for example program settings.

memory. Lorris can use this bootloader to program flash memory and read and program EEPROM.

4.3 Bootloader AVROSP

AVR Open Source Programmer is protocol used by several bootloaders by Atmel for chips ATmega and ATxmega. Lorris can use this protocol to program and read both flash and EEPROM memory of the chip.

5 Module: terminal

Fundamental tool for every developer, classic text terminal. It shows incoming data in either text mode or as hexadecimal values of each byte and sends key presses.

User can set terminal's colors, font size, which sequence of control characters should be sent after return key press and behavior of several control characters (for example if character `\n` should create new line or not).

6 Joystick support

Lorris supports joystick in module analyzer to for example control robot. At first, I've used SDL library to access joystick, but it was not really suitable for my use – SDL is video game library, joystick support is only one of many subsystems this library contains. Its architecture also wasn't ideal to use in Lorris.

I haven't found any suitable replacement of SDL, so I wrote my own library.

It is called **libenjoy**, it works under Windows and Linux and it is very small and simple. One major advantage over SDL is that it can remember connected joysticks – if you disconnect joystick and then plug it in again (because you want to reorganize cables on your desktop or because of bad USB connection), it will open the joystick again by itself – without any user interaction.

Libenjoy is released under GNU LGPLv2.1 license.

- GIT repository: https://github.com/Tasssadar/libenjoy

7 Android application

Application for Google Android platform is the next step in Lorris' development, because mobile devices with this operating systems are almost always at hand and are sufficient to quickly solve smaller problems.

Application **Lorris mobile** acts as portable addition to desktop version of Lorris – it may not have all the features of desktop version, but helps when you need to quickly correct or debug something out in the field.

App works on all tablets and phones with Android OS version 2.2 and higher, it is optimized also for bigger tablet screens and can be obtained in official distribution channel of Android application – in Google Play Store. You can find it by searching for "Lorris".

Lorris mobile has similar architecture as desktop Lorris. User has to create session first, so that everything he opens can be saved. After user loads the sessions, he gets to main screen of the application, where he can open modules in tabs, much like in desktop Lorris.

7.1 Programmer

Module programmer can program chips using bootloaders **avr232boot** and **AVROSP** and also using Shupito programmer, if the device has USB host capabilities.

This part of Lorris mobile uses pieces of native code from desktop Lorris, which means the code is faster and easier to maintain.

7.2 Terminal

Classic terminal. Is has most features of the terminal in desktop version – it displays data (as text or hexadecimal values), sends key presses and user can set terminal's colors, font size and which control characters are sent after return key press.

8 Real world usage

Members of some of the technical clubs on DDM[12] Junior in Brno became the first users of Lorris toolbox on the very beginning of its development several years ago. Lorris helps there with development of various devices, mainly robots, and kids who learn how to program microchips use *Shupito* programmer and thus also the Programmer module in Lorris. Modules terminal and analyzer are also useful during microchip programming lessons, terminal for simple communication with the chip and later analyzer for more advanced data processing.

Lorris is ideal for use in DDM Junior also because it is free – bigger company which makes microchip applications would probably get expensive commercial program similar to Lorris or develop its own single-purpose applications. However, solution used by large companies is somewhat inaccessible for state-funded institutions.

Nowadays Lorris has about 20 users on DDM Junior alone. Number of users however rising thanks to gradual spreading of *Shupito* programmer among users in whole Czech Republic.

I use Lorris whenever I work with robots and/or microcontrollers – to display data, program chips or control whole devices. Following list presents only some of the most significant applications made by other Lorris users:

- Development of several robots for this year's Robotic day

[12] *Dům dětí a mládeže* – Organization which does classes, clubs, camps or trips for kids. Can be translated as *Children and Youth Center*

- Programming of wide variety of microchips, using either *Shupito* programmer or bootloaders
- Development of *Shupito* itself
- Development of cheap logic probe
- Debugging of chips for control of three-phase motors (i.e. *drivers*)
- Developmnet of system for controlling of up to 128 RGB LEDs for illumination of model plane
- Construction and programming of digital radio transmitter with ARM processor (semestral work)
- Development of line following robot (graduation work)
- Tuning of PID controller

Conclusion

After several years of development, I can certainly say the application meets all the requirements declared in chapter 1. On top of that, the program greatly exceeds original goals – it can also send data to device, program microchips and create proxy between serial port and TCP socket. In comparison to other applications I've found (as described in introduction) Lorris is also the only one which allows user to write his own script to parse data.

Lorris has already been used in several real-world applications and it also has growing ranks of satisfied users.

The application is continuously being enhanced, it is possible to virtually indefinitely add either more widgets to Analyzer (compass, gauge meter, ...) or whole new modules (e.g. interface for cheap logic probe currently in development by Martin Vejnár). The program consist of a little over 38,500 lines of code (without third-party libraries) at this time (8^{th} June 2014).

In the future, I would like to continue in adding new features to both computer and Android version of Lorris and in increasing of awareness about this useful piece of software.