Summer Fellowship Report

On

OpenPLC

Submitted by

Ramana Ranganatham and Rohit Imandi from Bennett University

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

June 29, 2018

Acknowledgment

We would like to take this opportunity to express our greatest gratitude to our mentor, Mr. Akshay Chipkar for guiding, supporting and helping us in every possible way. We were extremely fortunate to have Mr. Akshay as our mentor as he provided insightful solutions to problems faced by us thus contrbuting immensely towards the completion of this project. We would like to thank FOSSEE Team and IIT Bombay for giving us this opportunity and providing a platform to exhibit our skills. We would also like to express our deepest gratitude to Prof. Rama Komaragiri, HoD, Dept. of Electronics and Communication Engineering, School of Engineering and Applied Sciences, Bennett University for giving us an opportunity to be a part of this fellowship.

Contents

Chapter 1

Introduction

Programmable Logic Controllers (PLC) allows controlling high power tools which generally work at really high voltages, unlike the microcontroller that merely works at considerably small voltage values[1]. For example, controlling assembly lines.
Open Source PLC (or OpenPLC) has functionalities similar to that of industrial PLC but at affordable prices. In order to achieve the same and make a ladder logic compiler available across platforms our task was to port an open source PLC compiler- to linux. LDmicro is an open source ladder logic editor, simulator and compiler developed by Jonathan Westhues for 8-bit microcontrollers. It can generate native code for Atmel AVR and Microchip PIC16 CPUs from a ladder diagram[2]. GNOME gtk3 library was used for the development of GUI in C/C++.

Chapter 2

Porting LDmicro

2.1 Typedefs

This section is about all the required typedefs for porting the compiler. This in intended to ensure uniformity throughout the source code across platforms. Following are the typedefs :

Gtk typedefs

Windows	**Linux**
HCRDC	cairo_t *
HWID	GtkWidget *
HWND	GtkWindow *
HLIST	GtkTreeModel *
ITLIST	GtkTreeIter
HAPP	GtkApplication *
HTVC	GtkTreeViewColumn *
GDRECT	GdkRectangle
PGDRECT	GDRECT
HMENU	GtkWidget *
HITLIST	GtkTreeIter *
HICON	GdkPixbuf *

Windows typedefs

Windows	**Linux**
BOOL	bool
BYTE	unsigned char
WORD	unsigned short
DWORD	unsigned int
UINT	unsigned int
SIZE_T	size_t
LONG	long
WCHAR	wchar_t
CHAR	char
LPCWSTR	const wchar_t*
LPCSTR	const char*
ATOM	unsigned short
LPCTSTR	const wchar_t*
LPCTSTR	const char*
LPWSTR	wchar_t*
LPSTR	char*
LPTSTR	wchar_t*
LPTSTR	char*
PVOID	void*
LPVOID	void*
HMODULE	void*
HHOOK	void*
HANDLE	void*
HINSTANCE	void*
HGDIOBJ	void*
UINT_PTR	unsigned int
WPARAM	uint64_t
LONG_PTR	long
LPARAM	long
LRESULT	long

2.2 Cmake file

The compilation of the project is done using the help of the Makefile, which is generated using Cmake. In order to do so, the CMakeLists.txt must be created for every module that needs to be compiled.

This project consists of three modules linuxUI library, freezeLD library, and LDmicro's main program. In each of the library folders a CMakeLists.txt file is created in order to build the libraries. The main CMakeLists.txt file responsible for generating the Makefile is created in the source directory. Using the add_subdirectory command provided by Cmake, the libraries are queued to be built.

The main CMakeLists.txt file also contains commands to enforce C++ 11 standard, identify is the system is x64 or x86 as well as the version of GTK installed. It also runs Perl scripts to generate necessary header and source files required by LDmicro, and to run post build tests to ensure LDmicros compiler is functional.

2.3 FreezeLD library

The freeze library of the original LDmicro for Windows provides functions to store and retrieve window configuration data such as window size and position from the Windows registry.
The freezeLD library of LDmicro for Linux implements a pseudo registry by creating a directory tree inside a hidden folder in the users home folder.

2.4 LinuxUI library

The linuxUI library provides all the necessary functions, definitions and macros that otherwise the Windows library would have provided. This library consists of two sets of header and source files, linuxLD and linuxUI. The linuxLD header and source files combined provide the majority of the definitions and macros that are required, they also provide all the functions that do not require GTK for its functionality such as HeapAlloc a function that is responsible for allocating memory that the program requires.
The linuxUI header and source files provide all the functions that implement GTK such as GetSaveFileName a function that is responsible for providing the user with a GUI to pick a file to save the program.

2.5 Main User Interface - GUI

The main window is the window which interacts with the user. It is a GtkWidget* that includes all the components (other widgets) like menubar, menus, status bar, drawing window, list, etc. and their respective functionalities. LDmicro user interface for Windows consisted of various WinAPI functions which performed alot of automatic background rendering to display the main window along with its functionality.
Unlike Windows, Linux doesn't handle the most of the background processes automatically, thus forcing us to implement every minute detail. However, WinAPI provides functionality to define windows classes that can be used as a template to generate windows. It also handles all the asynchronous calls with minimum efforts.
Ensuring that the main window worked properly actually helped in debugging functionalities of other components (or widgets) as they had to be attached into the main window.

2.6 Adjustments and Functionality of the compiler

No changes were made to the core functionality of the compiler. Any changes made were made only to support the execution of the software in Linux. This includes definition of required functions in various files or re-arrangement of the original source code to comply with gtk. Focus was completely on porting and proper implementation of the given functionality. We also ensured that even the function signatures and originally defined variables remained unchaned.

All the keywords and GUI functions required to run the software in Linux were created in another folder called linuxUI.

Necessary changes were made only to the GUI part of the software.

2.7 Drawing Window

This was the most important part of the main window as the user desgins the ladder logic to run the desired task. The draw window is a GtkDrawingArea that contains a cairo context which can be used to draw text and geometric shapes like rectangles. The ladder logic program that is to be painted on the window is done by a function which is asynchronously called everytime an event occurs like painting something new, changes in the program or cursor movement. This is like refreshing the window whenever an event occurs.
This widget is packed into the main window.

2.8 Dialog boxes and Shortcuts

Finally, we worked on creating various pop-dialogs and other windows in menus like the **Manual window** and **About window** inside the Help menu. Most of the small pop-up dialogs had to be ported carefully as they took inputs for the contacts of the ladder logic. This involved checking wehther the variables were being correctly set and displayed onto the list so that the program runs without any hindrance. For example, the **contacts dialog** pops up on double-clicking (or hitting the **Enter key**) the ladder logic drawn by the user. This dialog asks the user for the voltages to be assigned to the contacts of the ladder drawn.
We also created keyboard shortcuts to draw the required structure on the screen or to open another window. For example, the alphabet key **c** draws a simple structure on the drawing area.

Chapter 3

Reference

1 https://openplc.fossee.in/

2 https://github.com/akshay-c/LDMicro-linux/tree/Akshay/ldmicro

3 https://developer.gnome.org/gtk3/stable/GtkWidget.html

4 https://stackoverflow.com/questions/16539127/gtkentry-change-text-on-user-input?rq=1

5 http://www.mit.edu/afs.new/sipb/project/gtk/gtk_v1.2/tutorial/html/gtk_tut
6.html