Graphical Programming Language LabVIEW

&

Xcos, GNU Radio or Blockly as an Open Source Alternative to LabVIEW for Data Acquisition and Control

Submitted by:

Saruch Rathore

Electrical Engineering Department

Pandit Deendayal Petroleum University

Supervisor:

Prof. Kannan M. Moudgalya

Chemical Engineering Department

Indian Institute of Technology- Bombay

Abstract:

This report is a literature survey of the graphical programming language: LabVIEW, which has been developed by National Instruments. LabVIEW uses G programming language which is a data flow programming language, in which the flow of data determines the execution of the program unlike the traditional text-based programming language, where the instruction decides program execution. Different features of LabVIEW such as its graphical UI, parallel programming and drivers for majority of the hardware makes LabVIEW a user friendly and efficient language compared to text based programming languages. The report also targets upon the need for developing an open source alternative of LabVIEW for data acquisition and control. The alternative framework has been proposed which make use of Xcos, GNU Radio or Blockly and thus enable us to perform some of the functionality available in LabVIEW. All the three gives convenient graphical front end similar to LabVIEW. The computation in Xcos is carried out by itself, whereas in GNU Radio and Blockly it can be through Python, Scilab or Xcos code running at back end. As nowadays majority of the DAQ hardware are based on serial communication protocols, the proposed alternative is only valid for data acquisition through serial port. The GUI feature can also be improved using PyGTK package.

List of Figures:

Contents:

Chapter 1

Introduction

The institutes in developing countries are now concentrating on higher education, which is leading them to invest large amount into its facilities. Considering expensive proposition of software products and its update, they are in need of low cost reliable software for data acquisition and control. Many of the software available in the market are proprietary software, which are very expensive. The students having less experience with such proprietary software find difficulty adapting to them while joining an industry. The disadvantages of the proprietary software are listed below [2]:

1. The source code of the software is closed, restricting modification
2. It takes a long time for installation and fixing the bugs announced by the manufacturer
3. Any update or add-ons released by manufacturer will be require to be purchased
4. Sharing and redistribution of the software is also illegal as the they are licensed
5. There is no possibility of forking

FOSS is defined as software, the source code of which is available to all users. FOSS stands for free and open source software, free refers to the freedom to copy and re-use the software, rather than to the price of the software.

FOSS is a low cost and reliable solution to the problems faced due to proprietary software. There is large community of FOSS, discussion forums where people from around the globe share their ideas. Full access to the codes allows other to contribute to the development of the software. As its existence does not depend upon any particular provider, the user can modify it according to their own need, without concerning to anyone.

The concept of FOSS is slowly making its way into the market. The low cost and vast community of FOSS is encouraging the institute and industries to switch from expensive proprietary software to open source software. Other than low cost of FOSS, they are considered to have better reliability, performance and security.

This report is organized as follows. Chapter 2 describes LabVIEW, its features and application area in brief. Chapter 3 gives an introduction to different possible alternative to LabVIEW. Chapter 4, 5 and 6 explains different available features of GNU Radio, Xcos and Blockly. Chapter 7 gives an introduction on the FOSS enhancement tools used for modification in GNU Radio, Xcos and Blockly. Chapter 8 explains enhancement and the use of the alternatives to perform functionality available in LabVIEW. Chapter 9 compares the proposed alternative framework with LabVIEW and points out the further required improvements.

Chapter 2

LabVIEW

This chapter provides and brief over view of LabVIEW, its features and area of application. First we present and brief introduction of LabVIEW, its area of application is explained in the further subchapter.

2.1 Introduction to LabVIEW

LabVIEW stands for Laboratory Virtual Instrument Engineering Workbench is a graphical programming language developed by National Instruments. LabVIEW is programming development application but unlike C it uses graphical blocks instead of text code for writing programs.

The programming language used in LabVIEW, also referred a G programming, is a data flow programming. In contrast to text-based programming languages, where instructions determine program execution, dataflow programming uses the flow of data determines execution. G represents an extremely high-level programming language whose purpose is to increase the productivity of its users while executing at nearly the same speeds as lower-level languages like FORTRAN, C, and C++.

The programming in LabVIEW only requires a little experience, this is due to interactive GUI of LabVIEW which provides simple blocks and icons to which user is familiar with. This is often reinforced in colleges and universities, where students are encouraged to model solutions to problems as process diagrams. However, most general-purpose programming languages require you to spend significant time learning the specific text-based syntax associated with that language and then map the structure of the language to the problem being solved. LabVIEW allows programming which looks similar to a flowchart; this makes it easier for engineers and scientists to quickly understand because they are largely familiar with visualizing and even diagrammatically modelling processes and tasks in terms of block diagrams and flowcharts. The programs in LabVIEW are known as Virtual Instruments (VIs) because their appearance and operation imitate physical instruments. In LabVIEW, the user has to build a user interface using set of predefined tools and objects, this window is known as front panel. Front panel contains controls and indicators i.e. knobs, buttons, graphs etc. The coding for the front panel is done in another window, known as block diagram. The block diagram contains the graphical source code of the objects placed in front panel, it also contains structures and functions which perform operations on controls and supply data to indicators. LabVIEW allows the user to build a program by dragging and dropping virtual representations of lab equipment, connecting them according to the flow chart in the block diagram.

2.2 Features and advantages of LabVIEW over text based languages [6]

1) Program the way you think

LabVIEW differs from text based programming language in two ways. First, it allows program to be written with the help of graphical icons, which makes it easier for a non-programmer to program. Second, it executes according to the rules of data flow, due to which a user just needs to program similar to a flow graph.

2) Parallel programming and multithreading

To take advantage to multi-core processor you have to manually assign pieces of code to threads, which can be difficult and time consuming. Whereas in LabVIEW, the compiler automatically determines when two sections of code are parallel and independent of each other, they can run at the same time on different cores of multi-core processor by means of multithreading.

3) Large Libraries with built-in signal processing blocks

LabVIEW includes large libraries with a large number of functions for data acquisition, signal generation, mathematics, statistics, signal conditioning, analysis, etc. Number of advanced blocks for functions such as integration, signal generation, filtering, digital signal processing are part of the libraries.

4) Code compilation

The graphical code is translated into executable machine code by interpreting the syntax and by compilation. The run-time environment makes the code portable across platforms. Generally, LabVIEW code can be slower than equivalent compiled C code, although the differences often lie more with program optimization than inherent execution speed.

5) Code re-use

LabVIEW allows code reuse without modifications: as long as the data types of input and output are consistent. User can use the same code with different hardware or systems.

6) Hardware interfacing

LabVIEW gives an extensive support for accessing instrumentation hardware. Drivers and abstraction layers for many different types of instruments are already included; they present themselves as drag-drop blocks. Not only does LabVIEW include drivers for all NI hardware, it also provides over 10,000 instrument drivers for connecting to third-party instruments.

7) Interactive debugging tool

LabVIEW provides unique debugging tools that can be used to watch as data interactively moves through the wires of a LabVIEW program. Always-on compiler is another feature of LabVIEW, which compiles continuously while user is developing the program and provides feedback on the application.

8) Abstraction of low level task

LabVIEW automatically hands many of the low level task that a user faces while using traditional programming language. It automatically handles allocating storage to a data, and de-allocating it while not in use.

9) Combing with other languages

Many times mathematical expressions and formulas can be easily represented using text. LabVIEW provides use of text based language with graphical language. Features such as Formula node and MathScript allows user to write in text format. Mathscript uses .m file syntax.

10) Many hardware targets, programming with FPGA

LabVIEW is a cross-platform, cross-target tool. The user can run a code on an industrial controller or embedded PXI system. LabVIEW also provides even greater parallel execution by extending graphical programming to field-programmable gate arrays (FPGA).

11) NI LabVIEW community

The online LabVIEW community has more than 140,000 registered members from 83 countries, who can communicate through NI discussion forum. Through NI code exchange user can have access to more than 8000 LabVIEW examples.

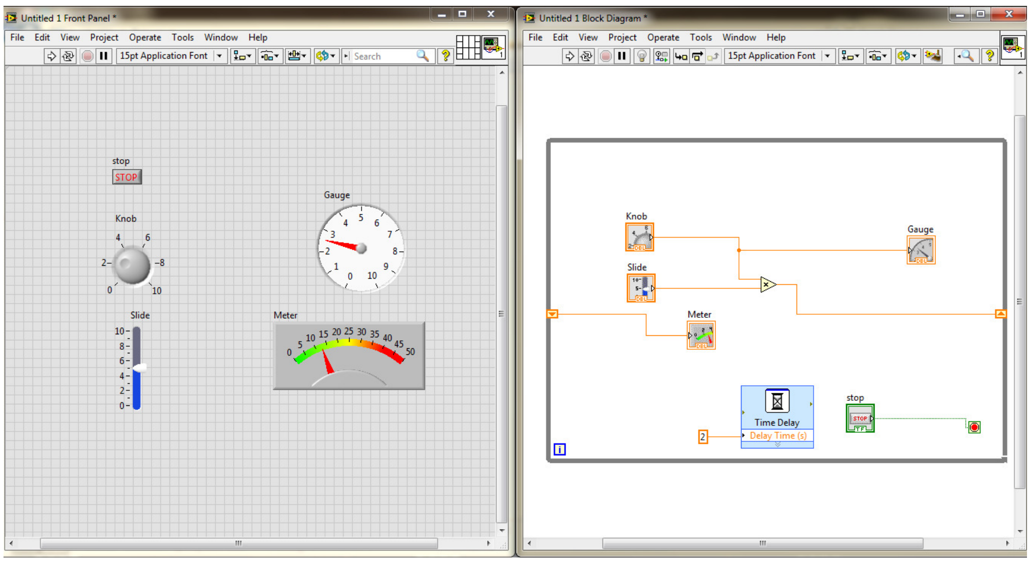


Figure 2.1: LabVIEW window, having Front Panel and Block Diagram

2.3 Application areas of LabVIEW [6]

LabVIEW helps a user to develop applications, whether it's taking simple measurements with simple programming or a complex consisting FPGA technology.

1) Measurements and Processing Signal Data

LabVIEW having drivers for all NI hardware as well as more than 8000 third party DAQ hardware makes measurements in real time a simple task. With availability of filtering and signal processing blocks, raw data can be easily turned to results.

2) Instrument Control

LabVIEW can be used to create PC based controller, having capability to acquire data from any stand alone instrument through any bus.

3) Automating test and validation system

LabVIEW provides user with tools to create powerful test software. The combination of reconfigurable hardware and LabVIEW system design software empowers user to build virtually any automated test system faster.

4) Designing embedded control and monitoring system

LabVIEW can target several embedded system platforms including NI Single-Board RIO and CompactRIO, which contain all of the basic building blocks of an embedded system.

5) Effective Engineering Education

LabVIEW being used in a variety of industries ranging from telecommunications to aerospace and nearly 500 manufacturing companies. Teaching students with the industry graded tool makes them comfortable to use while joining an industry.

Chapter 3

An endeavour to develop an open source alternative to LabVIEW

This chapter gives a brief introduction of the possible FOSS candidate as an alternative to LabVIEW, which are GNU Radio, Xcos and Blockly.

3.1 GNU Radio

GNU Radio is a free and open source software development toolkit that provides signal processing blocks to implement software radio. It can be used with external RF hardware to develop software defined radio, or without hardware in simulation like environment. GNU Radio applications are primarily written using Python programming language, while the supplied, performance-critical signal processing path is implemented in C++ using processor floating point extension, where available. The UI of GNU Radio uses different drag-drop blocks and connecting them to develop an application, which is similar to LabVIEW.

3.2 Xcos

Xcos is a free package that comes with Scilab. Xcos is based upon Scicos, and it is used for modelling and simulating dynamic system graphically, including both continuous and discrete systems. Xcos also provides some of the Modelica based blocks, i.e. Process oriented subcomponents. Models can be designed, loaded, saved, compiled and simulated. The blocks which are required for computation are already available with drag-drop feature, which makes Xcos appearance-wise as well as performance-wise similar to LabVIEW.

3.3 Blockly

Blockly is a web-based, graphical programming editor. Users can drag blocks together to build an application. Rather than being a stand-alone program, Blockly is a code editor that can be embedded into a larger project. Blockly can be useful for a variety of projects, as its all the codes are free and open source. Blockly creates an equivalent JavaScript, Python and other language code, to the UI created by the user.

Chapter 4

GNU Radio

This chapter provides a brief introduction to GNU Radio. Later, it gives an idea about the features available in GNU Radio that makes it a candidate for alternative of LabVIEW.

4.1 Introduction to GNU Radio

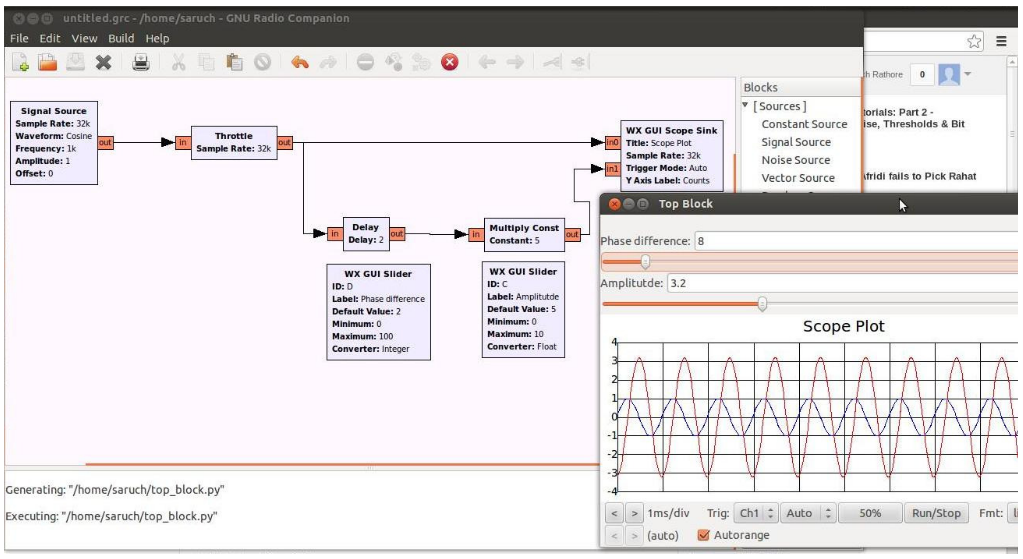
GNU Radio is a free and open source software development toolkit that provides signal processing blocks to implement software radio. It can be used with external RF hardware to develop software defined radio, or without hardware in simulation like environment. GNU Radio is released under the GPL version 3 license.

GNU Radio applications are primarily written using Python programming language, while the supplied, performance-critical signal processing path is implemented in C++ using processor floating point extension, where available.

An application can be created by connecting different pre-defined drag-drop blocks. The stream of data, coming from their port, is processed in the blocks and sent to their output port. The attributes of a block includes the number of input and output ports they have, as well as the type of data that flow through each one of them.

The graphical user interface GNU Radio is known as GNU Radio Companion (GRC). GNU Radio Companion is a graphical tool for creating signal flow graphs and generating flow-graph source code. In GRC, the application is build in form of flow graphs.

For creating new application user needs to place blocks on the canvas of GRC and connect them using wires, similar to LabVIEW. After compiling the flow graph, GRC converts it into a python file and stores it in a specified path with a specified name. We can later directly execute this file from the terminal. The scheduler in GRC uses Python built-in module threading to control the start, stop or wait operations of the signal flow graph.



Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0

Throttle
Sample Rate: 32k

Delay
Delay: 2

Multiply Const
Constant: 5

WX GUI Slider
ID: D
Label: Phase difference
Default Value: 2
Minimum: 0
Maximum: 100
Converter: Integer

WX GUI Slider
ID: C
Label: Amplitutde
Default Value: 5
Minimum: 0
Maximum: 10
Converter: Float

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 32k
Trigger Mode: Auto
Y Axis Label: Counts

Phase difference: 8

Amplitutde: 3.2

```
Generating: "/home/saruch/top_block.py"

Executing: "/home/saruch/top_block.py"
```

Figure 4.1: Graphical User Interface of GNU Radio (GNU Radio Companion)

4.2 Available features and advantages [3]

1) Ready-to-use blocks, which can be easily drag and dropped to GRC canvas and connected to each other using wires.

2) Most of the signal processing blocks are already available.

3) Parallel programming is possible

4) GNU Radio can be integrated with Scilab and Xcos for complex commutation, GNU can start Scilab and send commands to it for processing. It can also retrieve values from Scilab variables present in the Scilab memory for that instance.

5) Features which are not available can be easily included by adding custom blocks. Every block in GRC corresponds to an XML file that describes the block's parameters, input, output, and other attributes. Adding a custom block onto GRC is simply a matter of creating one of these XML block definition files.

6) GNU Radio has the ability to plot the output in real time

7) GNU Radio includes features such as knobs and sliders to change the parameters values during run time.

8) Other than communication through serial port, integrating GNU Radio with COMEDI, which contains device drivers for DAQ devices [1, 10].

9) GNU Radio can be equipped with image processing by integrating it with OpenCV [1, 11].

Chapter 5

Xcos

This chapter provides a brief introduction to Xcos. Later, it gives an idea about the features available in Xcos that makes it a candidate for alternative of LabVIEW.

5.1 Introduction to Xcos

Xcos is a free package that comes with Scilab. Xcos is based upon Scicos, and it is used for modelling and simulating dynamic system graphically, including both continuous and discrete systems. Xcos provides a modular way to construct complex dynamical systems using a block diagram editor. Models can be designed, loaded, saved, compiled and simulated. Xcos can be compared to Simulink from Mathworks.

Xcos is extensively used in engineering colleges and industries for signal processing, mathematical computation, data and statistical analysis and control system design. Using Xcos, the user can construct a library of reusable modules (blocks) that can be used in different models in different projects. It contains many functionalities to help the designer optimize model parameters, validate models, generate C code, etc.

To build any application in Xcos, user needs to drag and drop relevant blocks and connect them according to the flow graph. The benefit of Xcos over GNU Radio is its large library, many of the blocks which are essential for building application is already present. Xcos also provides some of the Modelica based blocks, i.e. Process oriented subcomponents. Other than that user can also create custom user-defined blocks, according to the need.

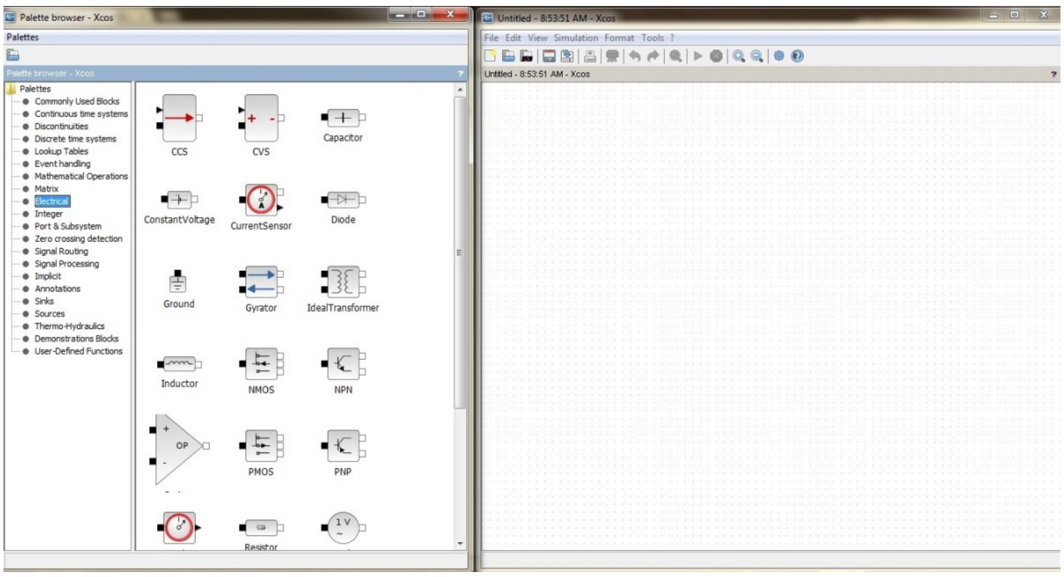


Figure 5.1: Graphical window of Xcos

5.2 Available features and advantages [4]

1) Programming is possible with the use of drag and drop blocks

2) Large library, Xcos's standard palettes and blocks module includes blocks for signal processing, mathematical operations, discrete and continuous system blocks and annotations blocks.

3) 2-D and 3-D plotting – pie charts, histogram, animation is easily possible and can be seen in real time on graph.

4) Super block can be used to create a hierarchical of block diagram

5) C code can be generated from Xcos model

6) User can easily save the data onto a file, using write to a file block, during execution and also read from a file using read from a file block.

7) Xcos also includes Modelica based blocks or implicit blocks, which allows the construction of systems using blocks modelling physical components in a natural way. For example, blocks of components of mechanical, electrical, electronic, hydraulic, thermal, control systems.

8) Parameter optimization can be implemented using Xcos

9) Scicos diagrams can be constructed using existing blocks, sometimes it is more convenient, or even becomes necessary, to develop new blocks. Xcos allows user to create its own custom blocks.

10) Debugging functionality is also available on Xcos, which provides information according to the debugging level.

Chapter 6

Blockly

This chapter includes an introduction of Blockly. Later, it gives an idea about the features available in Xcos that makes it a candidate for alternative of LabVIEW.

6.1 Introduction to Blockly

Blockly is a web-based, graphical programming editor. Users can drag blocks together to build an application. Blockly was influenced by App Inventor, which in turn was influenced by Scratch, which in turn was influenced by StarLogo. Blockly is not a programming language, one cannot 'run' a Blockly program. Instead, Blockly can translate the user's program into JavaScript, Python, or some other language using JavaScript parser. Since Blockly has multiple js files, it builds all the js files into smaller and dense *compressed.js files which are then used by Blockly applications. Blockly can be useful for a variety of projects, as it's all the codes are free and open source. Blockly is designed as an educational tool to introduce programming, a use case which relies on generating human-readable code.

External projects using Blockly include [5]:

BlocklyDuino: Ardiuno code generator.

CustomPacker: Human-robot packing system

GigaBryte: Wearable computers.

MIT App Inventor: IDE for Android apps.

Romotive: Phone-controlled robots.

Seal-blockly: SEAL script support.

Blockly and Espruino: Graphical Programming for Microcontrollers.

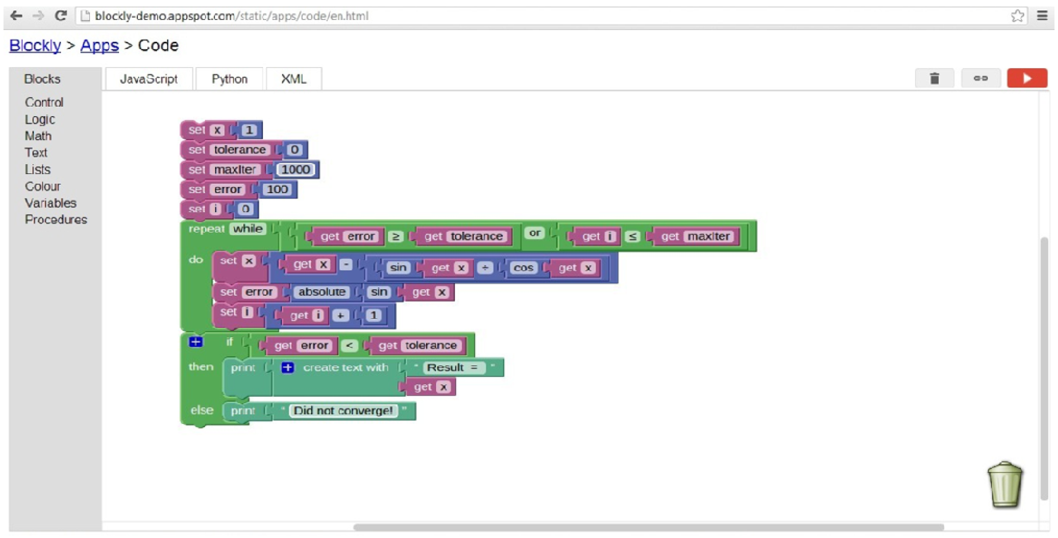

blockly-demo.appspot.com/static/apps/code/en.html
Blockly > Apps > Code
Blocks
JavaScript
Python
XML
Control
Logic
Math
Text
Lists
Colour
Variables
Procedures
set x 1
set tolerance 0
set maxIter 1000
set error 100
set i 0
repeat while get error ≥ get tolerance or get i ≤ get maxIter
do set x get x - sin get x + cos get x
set error absolute sin get x
set i get i + 1
if get error < get tolerance
then print create text with " Result = " get x
else print " Did not converge! "

Figure 6.1: Web-based Graphical Interface of Blockly

6.2 Available features and advantages [5]

1) Blockly comes with a large number of pre-defined blocks.

2) In order to interface with an external application, one can create custom blocks to form an API.

3) Blockly is 100% client-side, requiring no support from the server (unless one wants to use the cloud-storage feature).

4) There are no 3rd party dependencies (unless one wants to recompile the core).

5) User can modify or add source code, which requires installation

6) Blockly executes in a web-browser, no downloads or plug-in are needed.

Chapter 7
FOSS tools used for development of framework

This chapter gives a brief introduction of the FOSS tools such as Scilab, Python and PyGTK libraries which can be used to enhance the alternatives proposed, to make its functionality comparable to LabVIEW.

7.1 Scilab

Scilab is an open source, cross-platform numerical computational package and a high-level, numerically oriented programming language.It is mainly developed for signal processing and control related applications. Scilab is said to be open source alternative to Matlab. The language provides an interpreted programming environment, with matrices as the main data type.Scilab also allows user to create their own function. Any Scilab application can be called using C language. The Scilab package also provides a library of high-level operations such as correlation and complex multidimensional arithmetic. Scilab also includes a source code translator for assisting the conversion of code from MATLAB to Scilab. It has inbuilt functions for signal processing and image processing.

Scilab covers a wide spectrum of areas[9] :

- Aerospace
- Electrical
- Electronics and Communication
- Instrumentation and Control
- Automotive
- Defence
- Chemistry
- Physics, etc

7.2 Python and PyGTK

Python is a general purpose, high-level programming language. Python syntax allows programmer to express concepts in fewer lines of code. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming styles. The Python implementation is under an open source license that makes it freely usable and distributable.

Some of the key features of python include [13]:

- Clear and readable syntax
- Intuitive object orientation
- Very high level dynamic data type
- extensive standard libraries and third party modules for virtually every task
- Extensions and modules easily written in C, C++.

PyGTK is a set of Python wrappers for the GTK+ graphical user interface library. PyGTK lets you to easily create programs with a graphical user interface using the Python programming language [7]. The underlying GTK+ library provides all kind of visual elements and utilities for it. PyGTK is free software and licensed under the LGPL.

Chapter 8

Alternative Frameworks

In this chapter we will see complete framework of development of alternative of LabVIEW using GNU Radio, Xcos and Blockly. Different FOSS tools, which were discussed earlier, will be used in this framework.

8.1 Using GNU Radio

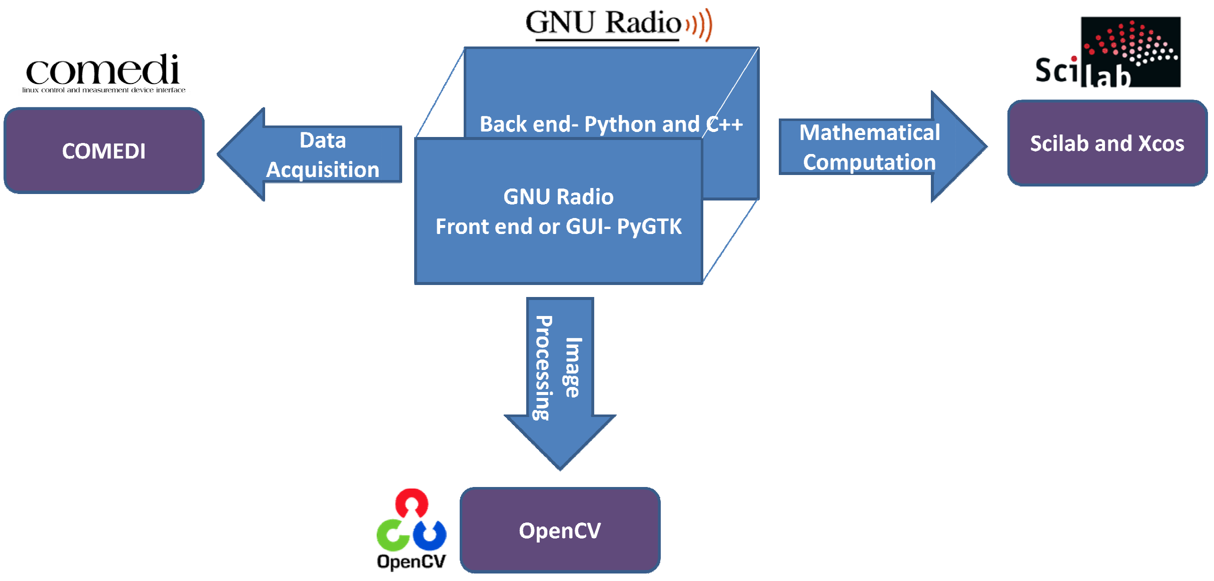


Figure 8.1: Frame work data acquisition and control using GNU Radio

We selected GNU Radio, as it has a GUI which is similar to LabVIEW, with knob and sliders, using which parameter values can be changed during run time. GNU Radio is an open source software and by very nature of the open source software, it is possible to integrate it with different open source system.

For online system analysis and control, we will need a fast and reliable mathematical tool which can be easily integrated with GNU Radio. Scilab and Xcos are the most convenient option for this task. Xcos and Scilab are used for mathematical computation. Scilab can be called using C/C++ code using call_scilab interface. This module can be used to interface GNU Radio with it. Also, to read/write data to/from Scilab memory api_scilab interface is required. It is also required that Scilab header file such as stack-c.h, call scilab.h and api scilab.h are installed. The most basic functions we need to call Scilab from C/C++ are StartScilab, SendScilabJob, SendScilabJobs and TerminateScilab.

For image processing OpenCV is the best tool, which can be integrated with GNU Radio. OpenCV is a library of programming functions mainly aimed at real-time computer vision [11]. The library is written in C and C++. To use features of OpenCV, we need to include OpenCV's libraries into GNU Radio. To compile and run OpenCV code in GNU Radio, we need to set the path of installed OpenCV in corresponding Makefile of a block in GNU Radio. We also need to include some of the OpenCV modules in the same Makefile [1].

For real time data acquisition, we can use COMEDI (Control and Measurement Device Interface) drivers. COMEDI is an open project which develops drivers, tools and libraries for different type of data acquisition [10]. COMEDI has high level library and integrated real-time support for most hardware. Codes in COMEDI are written in C and Python which makes it easy to integrate with GNU Radio.

Other than that we can even write our own custom block in GNU Radio. To get start with custom blocks we can download template folder from GNU Radio website and then provide proper modification according to required logic and attributes. Also, PyGTK can be used to modify and improve GUI of GNU Radio.

8.2 Using Xcos

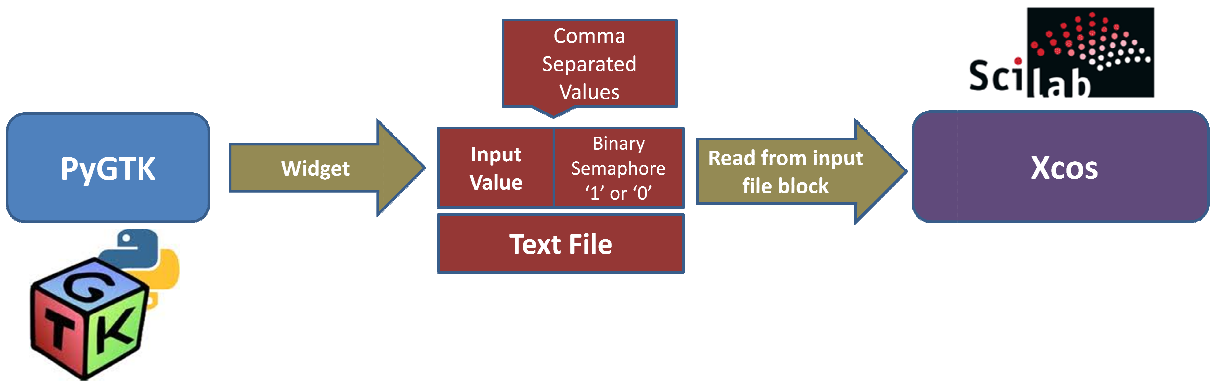


Figure 8.2: Frame work data acquisition and control using Xcos

Xcos is itself a very strong and reliable computation tool. It provides majority of the blocks for application related to Signal processing, Thermo-hydraulic, Mathematical operations, Discrete and continuous system, Electrical. Also, user can define their own block as it is open source. It also has a feature to give output in real time on graph.

The only major feature that Xcos does not provide is changing of parameter values during run time, which is extremely necessary for control application. This feature can be implemented using PyGTK widget and 'read from input file' block of Xcos.

PyGTK widget, which appears as a knob or a slider, can be used to write comma separated value into a text file. The value on the right hand side of the comma will be binary semaphore, which function as a flag. The value on the left hand side of the comma can be any value, which is given by the user as an input.

'Read from input file' block allows user to read a data from a file with the name defined with the Input File Name parameter, in text formatted mode or in binary mode. This block can be modified in such a way that when the value of the flag is 1, it will read the input value and change flag value from 1 to 0. If the value is 0, then Xcos will repeat the previous value. On PyGTK side, when there is difference between current input value and previous input value and also the flag is 0, it will write the new value and change flag value from 0 to 1. And thus we will be able to implement a knob into Xcos.

8.3 Using Blockly

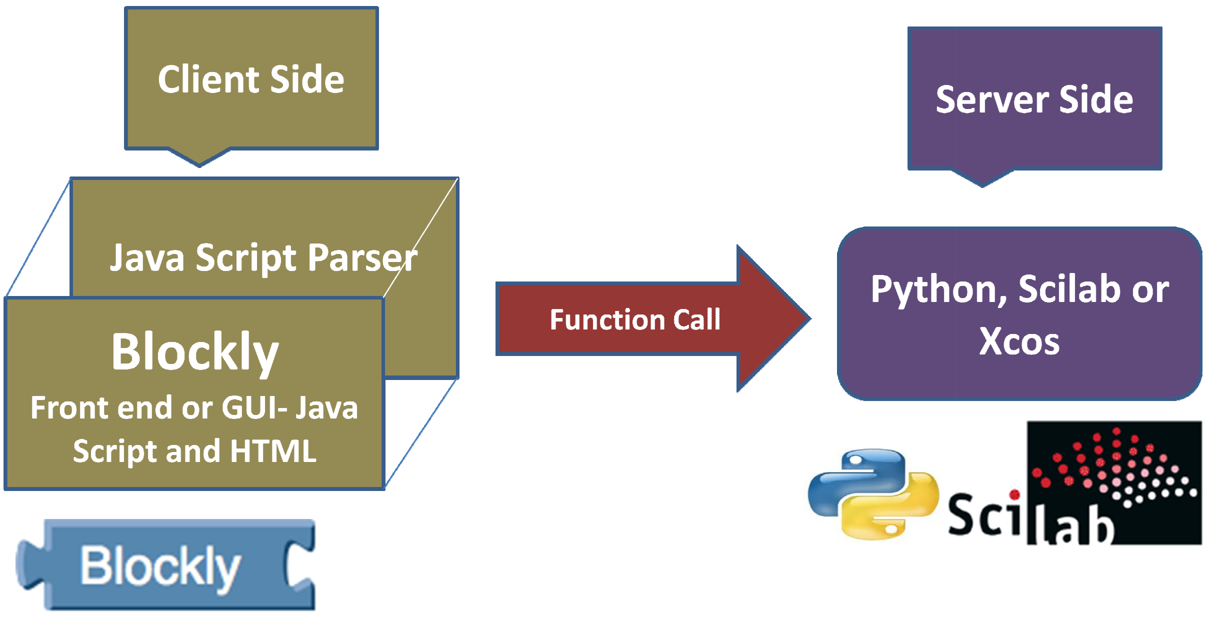


Figure 8.3: Frame work data acquisition and control using Blockly

The best feature of blockly is that it executes in a web browser, it does not require any download or plug-in. Blockly does not run any program, it translates user's program into JavaScript, Python, or some other language using JavaScript parser. User can even have access of blockly through a smart phone.

Scilab and Python can be used for computation. Scilab and Python will be on the server side, where the program will run. User will create an UI on the web browser, blockly will create a function call using JavaScript parser. The functions will be defined on the server side. On the server side the program will run and send the results back to client. Blockly lacks blocks required for building application of control, but feature of adding custom block is available.

Chapter 9

Comparative Analysis

In the previous chapter we discussed different possible framework, which can act as an alternative to LabVIEW for data acquisition and control. In this chapter we will compare these frameworks.

9.1 Comparative analysis among proposed alternative frameworks

Framework using GNU Radio	**Framework using Xcos**	**Framework using Blockly**
Stand Alone Software, client is required to have it installed on their system.	Stand Alone Software, client is required to have it installed on their system.	Web based software; client just needs to have a web browser.
Signal processing blocks are available.	Signal processing blocks are available.	Signal processing blocks are not available.
May require computation tools such as Scilab, Xcos or OpenCV.	Requires no other computation tools, as Xcos itself is a fast and reliable mathematical tool.	Requires computation tools such as Scilab, Xcos or Python for complex mathematical operations.
Graphical User Interface is similar to LabVIEW; drag-drop blocks and connection using wires.	Graphical User Interface is similar to LabVIEW; drag-drop blocks and connection using wires.	Graphical User Interface is different from LabVIEW, it does not have wires for connecting blocks instead of that blocks are directly connected.
Parallel programming is not possible	Possibility of parallel programming	Parallel programming is not possible
Knobs and Sliders are already present through which parameter values can be changed during run time.	UI of Xcos freezes as soon as we run the application, so to implement knob feature PyGTK widget can be used.	Changing parameter value during run time is possible.
Large dependences are involved during compilation	Easy to install	No installation or plug-in is required.
GNU Radio is designed for signal processing application, so it does not include blocks required for control application.	Large library, Xcos includes ready to use blocks required for control application.	Blockly is designed for developing logic for programming, so it does not include blocks for complex operations.
Communication with large number of DAQ hardware is possible, by integrating it with COMEDI.	Communication is only possible through serial communication or USB.	Communication is only possible through serial communication or USB, the device will be on server side.

Through comparative analysis among the proposed frameworks, we conclude that Xcos based framework is better than the other two. Xcos is a graphical simulator is already has majority of the blocks required for signal processing and control application and also for complex mathematical operations.

The only major drawback that Xcos has is unavailability of knob or slider; through which the parameter value can be changed during run time. This feature can be implemented using PyGTK widget.

As Xcos is open source and allows building of custom blocks, new blocks according to the requirement can be added to Xcos library.

9.2 Comparative analysis between LabVIEW and Xcos based framework

LabVIEW	**Xcos based framework**
LabVIEW stands for Laboratory Virtual Instrument Engineering Workbench is a graphical programming language developed by National Instruments.	Xcos is an open source free package that comes with Scilab, used for modelling and simulating dynamic system graphically, including both continuous and discrete systems.
Built-in signal processing blocks analysis blocks: Signal generation, Waveform conditioning, Signal operation and conditioning	It has fewer blocks for signal processing application such as signal generation, signal quantization.
Mathematical module contains blocks ranging from basic to advanced level. Blocks for solving linear algebraic equation & differential equation, curve fitting, interpolation & extrapolation are part of this module.	It contains majority of the blocks required for mathematical computation. It also includes blocks for matrix operations.
Hardware interfacing and data acquisition: USB, Serial port, wireless/Ethernet connection, NI wifi-DAQ cards, RS-232, NI DAQ modules.	USB and Serial port. Interfacing with different hardware is possible using HART toolbox.
It has built-in P, PI and PID controller ready-to-use blocks.	It also provides P, PI and PID controller blocks.
It offers graphical programming for many of the microcontroller and also for FPGA.	Graphical programming is not possible for microcontroller.
Remote monitoring and control feature is available.	Xcos currently doesn’t offer any such feature.
It provides attractive GUI with features such as graphs, knobs, sliders, LEDs, switches and many more.	Features such as knobs and LEDs are not available but can be implemented using PyGTK.
Low level tasks such as memory allocation are handled by LabVIEW itself.	These features are also available in Xcos.

Chapter 10

Conclusion

Xcos is free and open source software which can be used instead of LabVIEW for industrial and academic purpose. Xcos being used and tested as a graphical simulator for long time; it has many blocks available to perform basic as well as advanced level computation.

Xcos already has many blocks available for mathematical operation and signal processing; missing blocks can be build using build custom block feature of Xcos.

Data Acquisition and hardware integration is possible through serial communication protocols. Even communication with different hardware can be made possible using HART toolbox.

Changing of parameter values during run time can be implemented using PyGTK widget and ‘read from input file’ block of Xcos. PyGTK can even be used to improve the GUI of Xcos, so that it can compete with interactive visual appearance of LabVIEW.

Implementation of virtual lab or client-server structure using Xcos based framework can be one of the future work.

Bibliography

[1] J. Y. Patil, K. M. Moudgalya, B. Dubey and R. Peter. GNU Radio, Scilab, Xcos and COMEDI for Data Acquisition and Control: An Open Source Alternative to LabVIEW

[2] A. Khelifi, M.A. Talib, MFarouk, and H. Hamam. Developing an initial open-source platform for the higher education sector-a case study: Alhosn University. IEEE Transactions on Learning Technologies, 2(3):239–248, 2009.

[3] GNU Radio. http://gnuradio.org/redmine/projects/gnuradio/wiki, June 2013.

[4] Xcos. http://www.scilab.org/scilab/features/xcos and http://www.scilab.org/scilab/gallery/xcos, June 2013.

[5] Blockly. https://code.google.com/p/blockly/w/list, June 2013.

[6] LabVIEW. http://www.ni.com/labview/, June 2013.

[7] PyGTK. http://www.pygtk.org/, June 2013.

[8] HART toolbox. http://hart.sourceforge.net/, June 2013.

[9] Scilab. http://www.scilab.org/, June 2013.

[10] COMEDI. http://www.comedi.org/, June 2013.

[11] OpenCV. http://opencv.org/, June 2013.

[12] S.L. Campbell, J.P. Chancelier and R. Nikoukhah. Modelling and Simulation in Scilab/Scicos with ScicosLab 4.4, Springer Publication.

[13] Python. http://www.python.org/, June 2013.