Authorization

In presenting this dissertation in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission to download and/or print my dissertation for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this dissertation for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

DESIGN AND IMPLEMENTATION OF COMPACT RECONFIGURABLE INPUT OUTPUT CONTROLLER FOR CRS ROBOT MANIPULATOR

by

Kashyap Kalavapudi

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Measurement and Control Engineering

College of Science and Engineering

Idaho State University

May 2017

Committee Approval

To the Graduate Faculty: The members of the committee appointed to examine the thesis of Kashyap Kalavapudi find it satisfactory and recommend that it be accepted.

_____ Major Advisor

Dr. Alba Perez-Garcia (Associate Professor, Associate Chair of Mechanical Engineering, College of Science and Engineering, Idaho State University)

_____Committee Member

Dr. S. Hossein Mousavinezhad (Professor, Electrical Engineering, Department of Physics, Nuclear and Electrical Engineering, College of Science and Engineering, Idaho State University)

_____Graduate Faculty Representative

Dr. Chikashi Sato (Professor and Associate Chair, Dept of Civil & Environmental Engineering)

Acknowledgments

This is an acknowledgement of the intensive drive and technical competence of many individuals who have contributed to the success of this thesis.

I'm immensely thankful & obliged to Dr. Alba Perez-Gracia, for giving me the opportunity and facilitating me to do this thesis and for her meticulous guidance and support throughout.

I take pleasure in expressing my respects and thanks to Dr. S. Hossein Mousavinezhad for his advice and support during the course of this thesis.

I express my deep sense of gratitude to Dr. Chikashi Sato for u for being my graduate faculty member.

Last but not least I would love to thank my parents for the bottom of my heart for their support and love without which I could not meet and have a chance to work with these inspiring personalities.

Abstract

The robotic manipulator CRS PLUS A150 have its own language, Robo Comm-II, to program and control it. Having a different coding or programming system for each robot is a problem in learning environments. The maker of this robot is defunct from more than a decade, making it difficult to get information, manuals and parts to keep the robot in working conditions. The ultimate aim of this thesis is to make the robot available to run on open source programming such as Robot operating system (ROS). In order to do so, the signals of motors and sensors have been identified and analyzed, and different controllers and drivers have been tested. The resulting system is open source and has the ability to interface with ROS as future work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The objective of this thesis is to control the CRS PLUS A150 robotic arm using different options: NI cRIO-9074 and LabVIEW programming and Arduino with L298N motor driver module.

CRS PLUS A150 robotic arm is manufactured by CRS Robotics Corporation. The company was acquired by Thermo Electron Corporation in 2002. Further Thermo Electron Corporation became defunct from 2006. As a result, there is no support from the manufacturer for this Robot. Moreover, there are very few manuals available online. The CRS PLUS A150 robotic arm came with CRSPLUS robot system controller uses programming software called ROBCOMM-II. Lack of proper documentation made CRSPLUS A150 robotic arm almost a black box and made it next to impossible to use the controller. This became a challenge to other academics as well. Dr. Saeid saeidi a faculty member of Hakim Sabzevari University mentioned that "We have got some CRS plus A150 robot here but no one can work with them and their catalog and software are lost, unfortunately. Can you or your TA help us on this problem?" [16] the same issue we had at ISU. There is only one technical paper published related to CRS PLUS robotic arm [17].

It is a challenging task to bring such machine into working condition. So, it was opted to use cRIO controller to control the robot with LabVIEW Coding. Advantage with LabVIEW coding is that LabVIEW can interface with ROS. Once the CRS PLUS A150 robot arm is ready to use, this system can be configured so that ROS will send and receive data packages from the local network and LabVIEW code will work as a mediator to these packages for ROS.

Later in this thesis, it was released that along with digital I/O, Driver should be procured as the cRIO cannot handle the voltage and current rage. Alternatively, L298N is selected to drive the motors. Further the controller is changed to Arduino. Selecting Arduino will not affect the aim of the thesis, as Arduino can be interface with LabVIEW. Thus, ROS can still be used even by using the Arduino.

1.2 Thesis overview

The report contains all the information to bring the CRS PLUS A150 to working conditions. Chapter-2 gives details of the CRS PLUS A150 Robot arm, motors and encoders of the robot. Chapter-3 is about other important components and their specifications. Chapter-4 gives the step by step procedure for installation of the important software used in this thesis. Chapter-5 is about the experiments conducted and results. At the end of the report are all the available manuals related to CRS PLUS robot, that have been collected from various resources collected over time.

Chapter-2
CRS Robotic Arm

This chapter discusses CRS Robotic Arm used in this thesis. All the technical details that are needed are given.

2.1 CRS-PLUS A150 robot arm

This robotic ARM belongs to A150 series of small industrial robot system, and was given with a RSC-P8 controller and configured to run with ROBCOMM communication package.

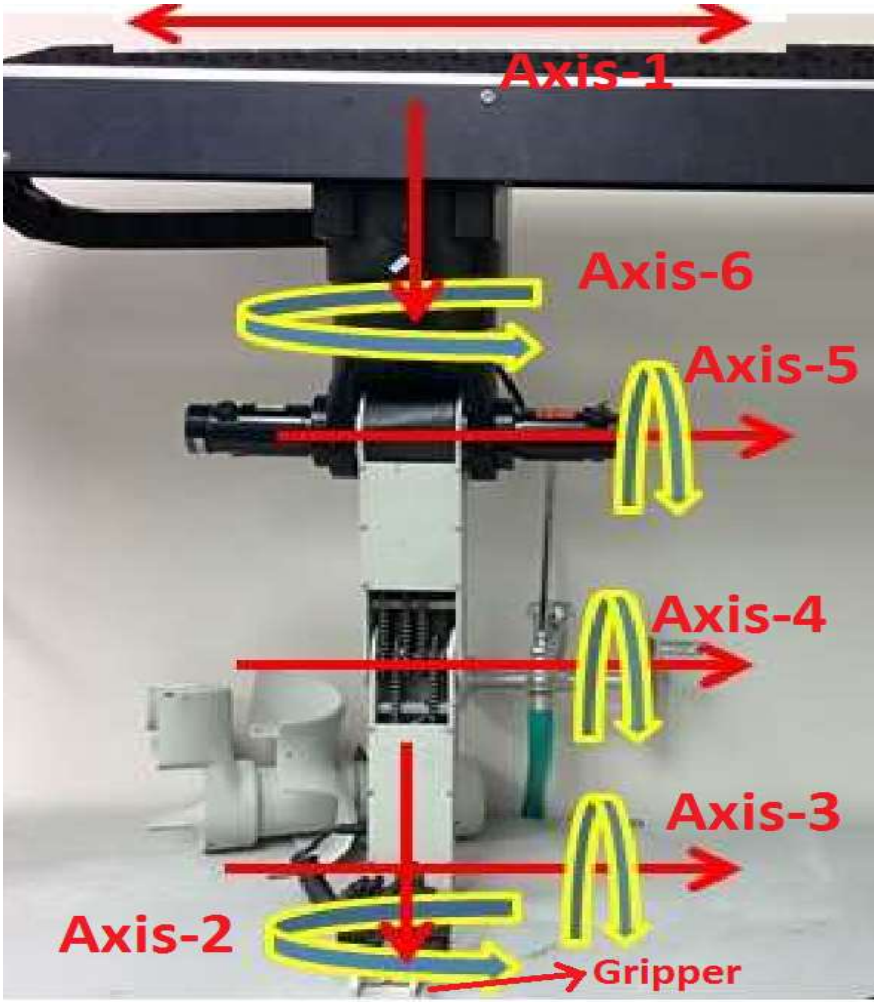


Figure 2.1: CRS-PLUS A150 robot arm labeled with Axis (edited from [7])

The company manufacturing is no more functional. As a result, information about this Robot is difficult to find. As well spare parts including the motors are to be ordered and not available off the desk.

This CRS robotic arm consists of six degrees of freedom. One translation, five are revolute joints and one gripper. These axes are labeled from Axis 1 to Axis 6 as this is a six-axis machine and for our understanding the axis is given with a name identical to the human hand. The details are tabulated in table 2.1. Later in this report the axis are specified by the same axis numbers.

Table 2.1: CRS robot Axis

Axis Number	Joint	Motion
_	Gripper	_
Axis-1	Column	Translational
Axis-2	Wrist roll	Rotational
Axis-3	Wrist pitch	Rotational
Axis-4	Elbow	Rotational
Axis-5	Shoulder	Rotational
Axis-6	Base	Translational

The Robotic Arm is with the above details is shown in the Figure 2.1[7]. It is suspended from the from the Axis-1 which is a linear axis, just below it is the rotational Axis-5 which can rotate from zero to 350 degrees and back to zero. Note Axis-5 can't go over 350. Figure 2.2 gives the elevation view of CRS-PLUS A150 robot arm

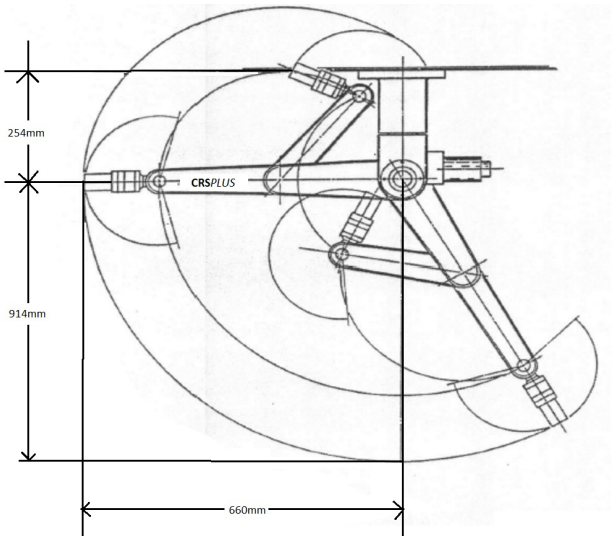


Figure 2.2: Elevation view of CRS-PLUS A150 robot arm (CRS PLUS A150 Manual)

Second, third, and fourth axis are Chain driven. Care must be taken in maintaining the chain system; the chain must be properly lubricated and should be checked. Figure 2.3 gives the plan view of CRS-PLUS A150 robot arm.

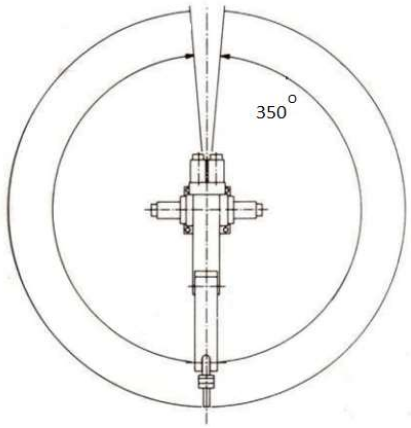


Figure 2.3: Plan view of CRS-PLUS A150 robot arm (CRS PLUS A150 Manual)

CRS-PLUS A150 robot arm can bear up to pay load of 2.2Pounds (1Kilogram) at Maximum speed or 100% duty and a pay load of 4.4 pounds(2Kilograms) at Reduced speed i.e. 80% duty.

The base is projected from a beam or column, the distance from the column to the base of the CRSPULS A150 Arm. The distance between base mounting surfaces to shoulder is 10 inches. The length of the joint from Shoulder to elbow is 10 inches. The length between elbow to wrist pivot is 10 inches.

The length between wrist pivot to tool flange surface is 2inches. The total length of the bot from its base is about 194mm including the gripper and a stroke length of 660mm. Proper care must be taken while running the motor as there are good chances of accident if we don't put soft end limits in the program (note that there are no end limiting sensors or limit switches on this robotic arm). To the range of motion of the robot arm and the maximum speed of each axis (motor) are tabulated in table 2.2.

Table 2.2: CRS robot Axis Maximum Speed

Axis Number	Range of motion	Maximum joint speed(Deg/sec)
Axis-1	1250mm	_
Axis-2	±180 Deg	300
Axis-3	±110 Deg	180
Axis-4	-130 to 0 Deg	100
Axis-5	47-0 to 100Deg	62
Axis-6	±175Deg	100

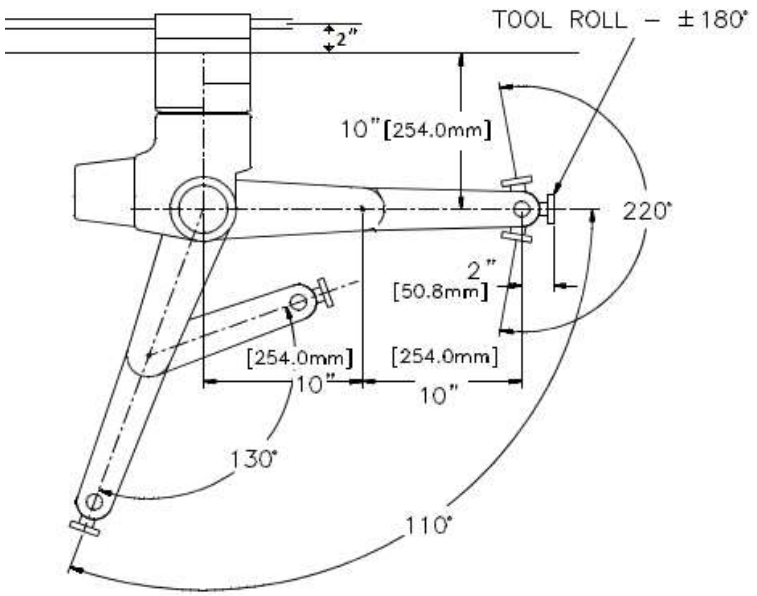


Figure 2.4: Dimension of CRS-PLUS A150 robot arm (Edited from CRS PLUS A150 Manual)

2.2 Motors

CRS-PLUS A150 robot arm is driven with Brushless DC servo motors manufactured by Torque Systems. Before going in to the details of the specific motors used, here is a review of basic information about with Brushless DC servo motors. Brushless DC servo motors are more reliable, more efficient and less noisy. The recent industrial trend is to use Brushless DC servo motors where the operations demand long life and reliability. They are also lighter and need less maintenance than Brushed Motors with equivalent power output.

Now let's have a glance on operational principle of a brushless DC servo motors. The stator of this type of motor has permanent magnets. Rotor has Cu (copper) coil wound as shown in figure 2.5**.** Electrical energy is provided to so that preceding winding(pole) have opposite magnetic polarity to that of permanent magnet and previous pole that just passed will get same polarity to that of permanent magnet.

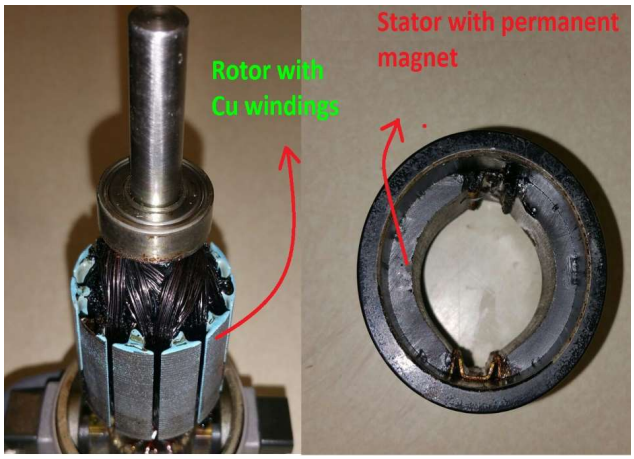


Figure 2.5: Stator and rotor of DC Brushless motor.

Now let's have a glance on operational principle of a brushless DC servo motors. The stator of this type of motor has permanent magnets. Rotor has Cu (copper) coil wound as shown in figure 2.5**.** Electrical energy is provided to so that preceding winding(pole) have opposite magnetic polarity to that of permanent magnet and previous pole that just passed will get same polarity to that of permanent magnet.

Thus, two forces push the rotor converting magnetic energy to mechanical energy. To identify exact point of time to change the polarity of the poles by switch the current brushless DC servo motor accomplishes commutation electronically using rotor position

feedback. The main disadvantage of brushless DC servo motor is cost; construction of a BLDC motor is actually simpler than that of brushed DC motor or AC induction motor. The higher cost of BLDC motor is caused by the additional driver circuit for BLDC motor.

All the motors in CRS-PLUS A150 robot arm are TORQUEMASTER 2110 Series of brush servo motors. They have fast response, accurate control and high torque-to-inertia ratios, capability to provide smooth operation throughout its range of operations. Speed -Torque Characters of motor are shown in figure 2.6.

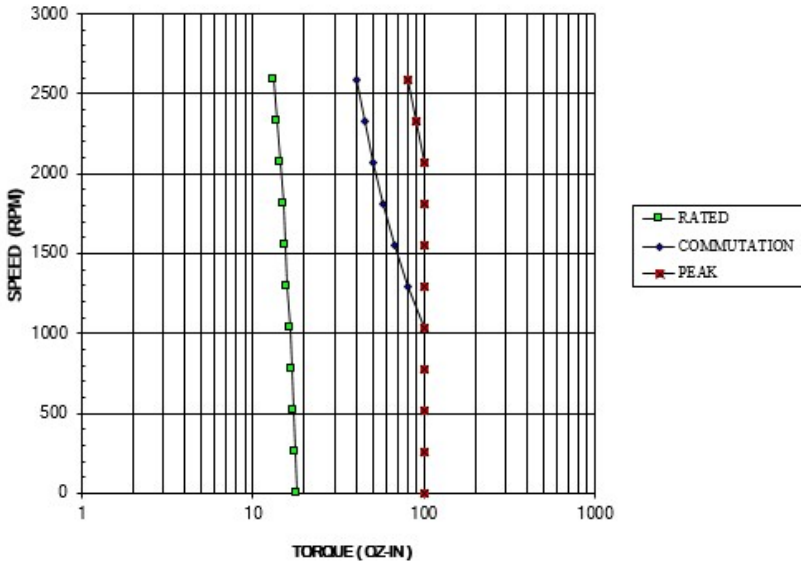


Figure 2.6: Speed -Torque Characters of TORQUEMASTER 2110 Series (TORQUEMASTER 2110 Series manual)

Motor connections were directly given to the Controller provided along with CRS-PLUS A150 robot arm. Because of very little information available about this bot, and for upgrading to better control, National instruments CRio-9074 controller used. So, the cables were detached from previous control board, and for identification of power cables for each axis the cable is cut open and labeled according to the Axis number.

2.3 Encoders

Encoder is electromechanical device that is used for sensing and provide feedback. TORQUEMASTER 2110 Series motor of CRS-PLUS A150 robot arm have incremental optical encoder. Digital signals are generated by this encoder in response to rotary moment of the motor, thus converting mechanical moment in to electrical signal to obtain position or speed measurement. There will be a LED or infrared Diode in the circuit and a LED or infrared sensor separated by a metal or glass grating disc. This disc is connected to the shaft of the motor and so, rotates along with the motor. The Light or Infrared rays are sensed by the sensor each time the Light passes through grating. Sensed signal is then converted to measurable electrical pulse by encoder.

The resolution of the encoder depends on number of increments per revolution. A picture of optical sensor is given in Figure 2.7[6](the picture is taken from an internet resource[6], author of this report doesn't claim credit for this Figure 2.7). Model/part number of the encoder on TORQUEMASTER 2110 Series motor is unknown; however in the manual of the CRSPLUS A150 robot Arm specifies that there are 1000 lines per revolution. (Refer page C-12 in the manual). So 1000 pulses for each revolution of shaft can be read at A and B and 1 pulse can be read at Z. One may contact Torque System for any further details if needed. As depicted in the figure 2.7, we have three signals for this optical sensor, namely A,B and Z. pulse signals from A and B are used to find the direction of rotation and measuring speed/position. Whereas, Z pulse signal indicates zero reference of the motor (note that there will be only one Z pulse signal per revolution).

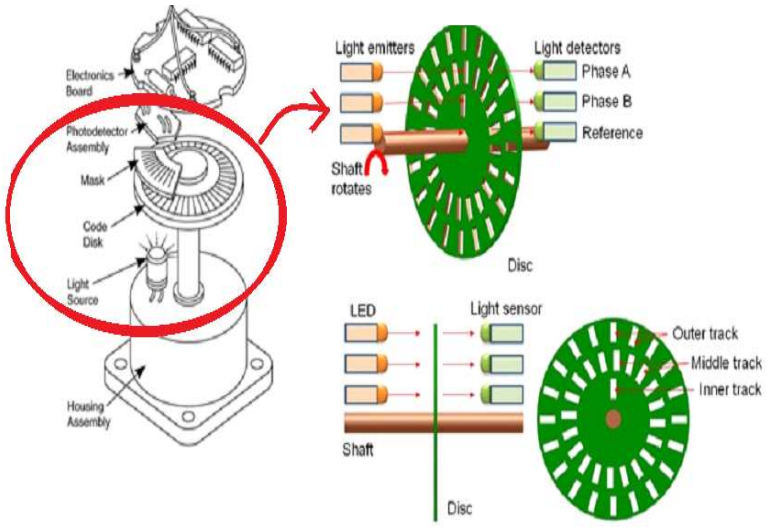


Figure 2.7: Optical Encoder (CRSPLUS A150 Manual, [6])

Figure 2.8 shows the relation between A, B and Z pulses of the encoder. This information is very necessary to understand and write the code for designed control of the robotic arm.

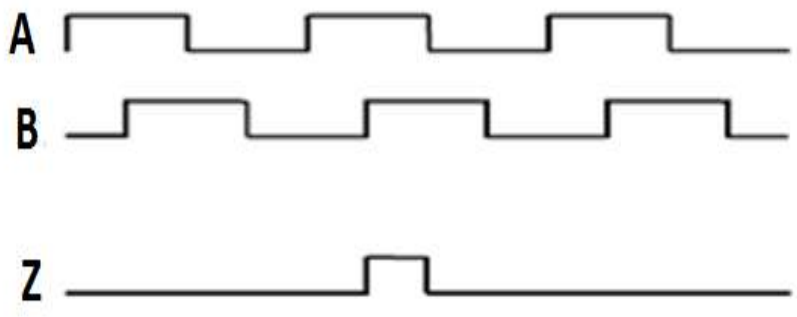


Figure 2.8: A, B and Z encoder pulses

Observe that the A and B are out of phase, this helps in determining the direction. For example, if A is high followed by B pulse, gives the direction of motor to be clockwise then B pulse getting high followed by A pulse gives the opposite direction i.e. counter clockwise. The number of pulses is counted for the speed or to identify the position of the motor. As discussed earlier Z is to give the reference signal.

Chapter-3

Other Components needed

This chapter discusses about the important electrical components that are needed for controlling/ operating the CRS-PLUS A150 robot arm. The following components are described thoroughly.

3.1 NI cRIO Controller
3.2 Driver module for motor
3.3 Other controllers
3.4 Regulated Power supply

3.1 National Instrument Compact RIO Controller (NI cRIO-9704)

The NI cRIO-9074 is an efficient integrated system. It is a combination of real-time processor and a reconfigurable FPGA (field-programmable gate array).The chassis is available with eight slots which can be used for NI C series Digital I/O, Analogue I/O, Motor Drive module, DAQ systems which can be used for embedded, machine control and monitoring applications. The real-time processor is of industrial grade worked with a frequency of 400MHz and 2M gate FPGA. This system features 128 MB of DRAM for embedded operation and 512 MB of nonvolatile memory for data logging. NI cRIO-9074 is powered by the NI LabVIEW reconfigurable I/O (RIO) architecture.

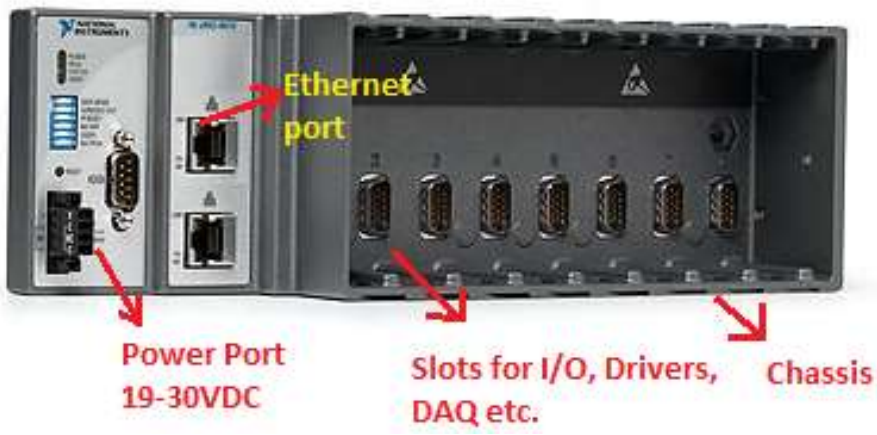


Figure 3.1: cRIO-9074(edited from NI website)

The cRIO-9074 features two Ethernet ports that you can use to conduct programmatic communication over the network and built-in Web (HTTP) and file (FTP)

servers as well as to add expansion and distributed I/O to the system. Picture of cRIO-9704 is shown in figure 3.1.[8].

To the cRIO-9704 six input modules are connected, the following of the details of the modules

1. Digital output module- NI 9476
2. Digital input module-NI 9425
3. Analogue output module- NI 9263
4. Analogue input module- NI9223
5. Analogue output module- NI 9263
6. Analogue input module- NI9223

3.1.1 Digital output module- NI 9476

NI 9476 is a 32 channel that operates on 24VDC; the output range is from 6V to 36V sourcing digital output with a maximum current of 0.25A.

Figure 3.2: NI 9476 and pinouts/front panel connections (NI 9476 manual)

This works at industrial logic level signals to wide range of relays actuators and motors. Figure 3.2[9] gives pin connection details of NI 9476.

The plan for this thesis is to write LabVIEW code to generate PWM signal from this module to control the Robot arm. But Note that the max current output is very low for CRS plus A150 Robot Arm, so we have to use an external driver for controlling CRS plus A150 Robot Arm.

3.1.2 Digital input module- NI 9425

NI 9425 is a sinking Digital input with 32-Channel. NI 9425 can read a signal in the range of 10VDC to 24 VDC. Proper signal conditioning is to be done before you expect to read any signal. Specifically in this thesis the quadrature encoder is to be read to control the CRS plus A150 Robotic Arm. The output of the encoder is 5VDC and if we connect it directly to the NI 9425 it can't read the signal. So, we need to amplify the signal before we provide the signal to the Ni 9425. Proper care should be taken while amplifying the signal as there is chance of amplifying the noise signal as well. Further more details are given in next chapters about signal conditioning.

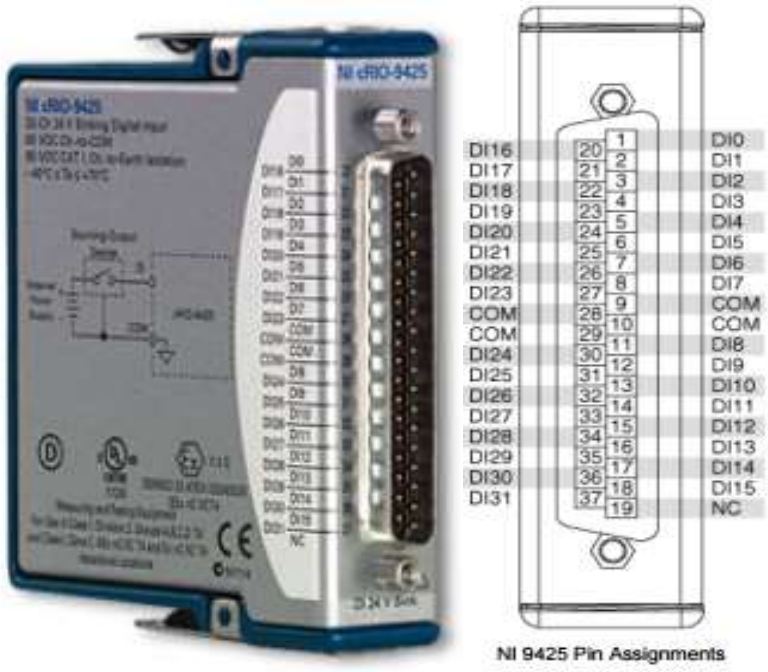


Figure 3.3: NI 9425 and Pinouts/Front Panel Connections(NI 9425 manual)

This report doesn't discuss about the Analogue modules as on cRIO, as we are not using them for now. But the technical manuals can be found in national instruments support website [11]. Once the modules are installed on to the cRIO we have to initialize cRIO and other software required to use cRIO, A detailed initialization procedure is given in chapter 4 of this report.

3.2 Driver module for motor

To control the motor direction of rotation, speed or position, we need a driver, as in this thesis we are using cRIO-9074 we can opt for a National instruments Driver module that is compatible with cRIO and the torque system Motors of the CRS Plus A150 robot Arm. But the issue is we need six driver modules and this is an expensive approach. The alternative is approach for an economical solution is to use H-bridge. H bridges are being used for a long time for controlling the motors. L298N is a known and reliable DC motor drive that is available of the desk and for a few bucks. We now have other issue here, that is the current and voltage rating of the motor. The motor at higher load/ speed requires a current more that 2Amps and voltage up to 30VDC or more. As we neither require handling high payloads nor high velocity, we fix the operation range from 7VDC to 12Vdc and max current to 2Amps.

3.2.1 L298N Driver module

L298N is Dual-channel, H-bridge motor driver module. It has high working efficiency, is capable of driving two DC motors simultaneously. And can provide an output of continuous 5VDC when you input more than 5VDC as driving voltage. It can proved a continuous current of 2A and can bear up to 3A MAX. It has inbuilt capacity filter, after flow protection diode, so it is more stable and reliable.

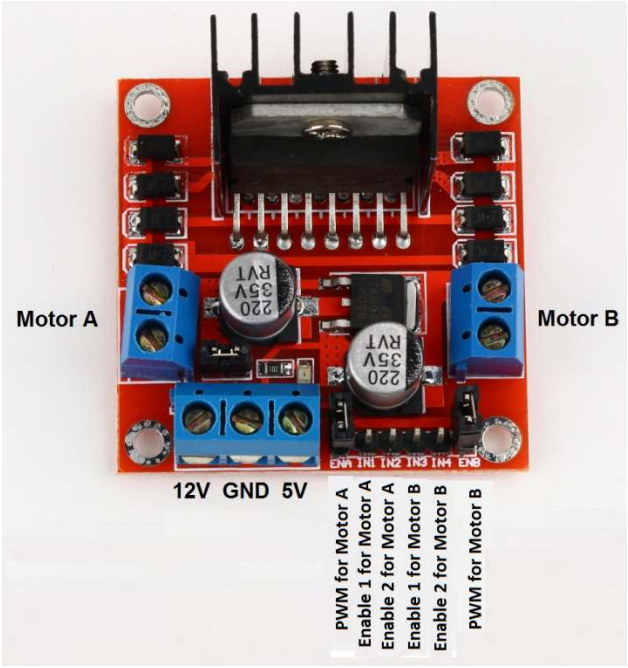


Figure 3.4 gives L298N H-bridge module

Three L298N Modules are to be used to control six-axis of the CRS plus A150 robot arm. Figure 3.4 gives L298N H-bridge module the necessary input and output connections details.

3.3 Other controllers

Further while doing experiments, there were issues generating PWM from digital output module NI 9476 using LabVIEW. The details are given in chapter five of this report. For this other controllers are tested and used to generate PWM. A brief introduction of following controllers is given here.

a. ARM controller board
b. Arduino

3.3.1 ARM controller board

After through research on generating PWM in LabVIEW, NXP 1768 board is selected to generate PWM signal to control the motor. It is an ARM Cortex-M3 based microcontroller that operates up to 120MHz CPU frequency. The ARM Cortex-M3 CPU works with a 3-stage pipeline that makes it a better operating microcontroller. The ARM Cortex-M3 CPU also has the ability to an internal pre-fetch unit that supports speculative branches. This board can interface through both Ethernet and USB. It can have the capability of motor control with PWM and Quadrature Encoder interface [12].Picture for NXP 1768 board and its pin configuration is shown in figure 3.5(taken from internet resource [13]). We can use pins p5 to pin p30 as digital I/O interfaces.

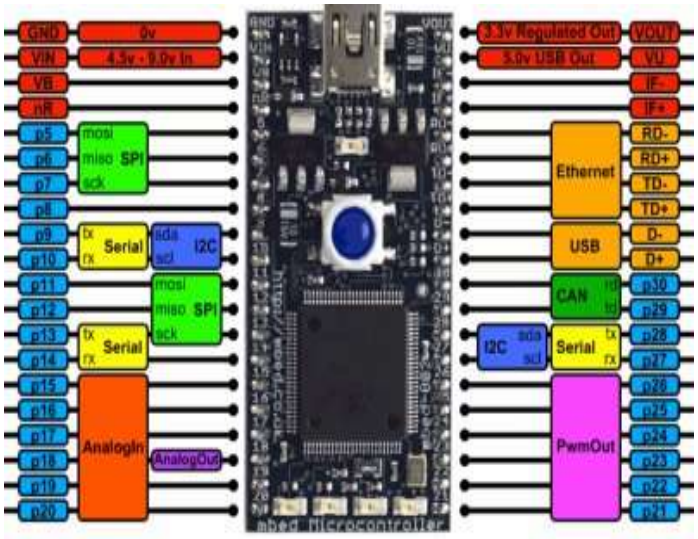


Figure 3.5: Arm mbed 1768 board (NXP 1768 manual)

This controller can an ability to generate 6 PWM signals and is specially allocated from pin P21 to pin P26. Sufficient for controlling CRS plus A150 robot arm. We can read the sensor from digital I/O pins. The tests done and code written on this controller are in chapter 5.

3.3.2 Arduino

Arduino is one of the most common microcontroller board that is used in academic projects. Arduino has onboard Analog I/O and Digital I/O. IT uses embedded programs which are called sketches. Arduino hardware and programming language are open source. It can interface with Pc through USB. There are multiple external plugin modules, adding them to can extend functionality of the Arduino. One more advantage is that the Arduino can also be interfaced with LabVIEW.

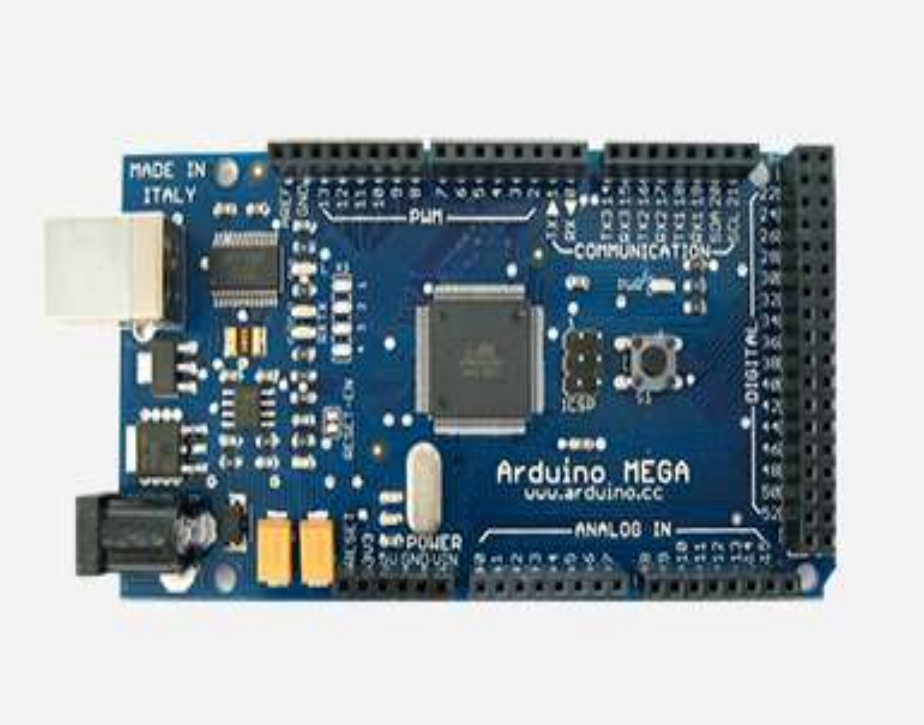


Figure 3.6: Arduino MEGA

Table 3.1 Arduino MEGA Specifications

Microcontroller	**ATmega1280**
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	39
	15 -PWM output (can also be used as Digital I/O)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	128 KB
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

LabVIEW must be version 9 or more to interface with Arduino. Moreover, feedback signals from encoder need not be amplified as Arduino can efficiently read pulses of 5VDC. An Arduino Uno is good enough to generate 6 PWM Signals but we need 12 also enable signals apart from reading encoder signals. So, Arduino MEGA is better as it has more number of I/O ports. Arduino MEGA has 54 Digital IO, 16 Analog IO. Other details are tabulated in table 3.1 and figure 3.6 [14] shows the Arduino MEGA.

3.4 Regulated Power supply

3.4.1 Introduction to regulated power supply

Regulated Power supply or power supply unit (PSU) is a device or system that supplies electrical energy. A power supply unit can be considered as source but it doesn't generate electric power. It just converts the electric energy to the level and type we need. For example Arduino needs 5VDC, Motors need 6 to 36VDC. Etc. but the voltage we have in our socket is 120VAC. So we need to step down the voltage to the required level and convert this in to direct current (DC), that doesn't have any ripples. For understanding the process take an example that 5VDC is needed. This whole process is divided into following steps

Step1:-First step is to convert high voltage to lower voltage level. In USA the mains are 120VAC, with frequency of 60Hz. In this step we have to bring this 120VAC to 12VAC.For this purpose we use a step down transformer, shown in figure 3.7. Step down transformer will step down the voltage at secondary and increases the current at secondary. There by maintaining constant power at both primary and secondary. Note that frequency is not affected. So we have a 12VAC with a frequency of 60HZ.

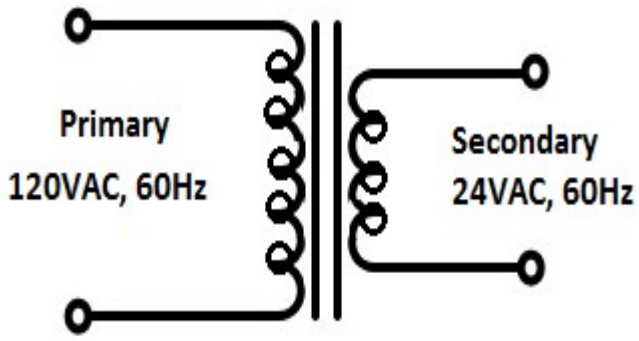


Figure 3.7: Step down transformer

Step2:- Now 12VAC is to be converted to DC Voltage. For the 12VAC is given to Bridge rectifier. Bridge rectifier converts AC to DC. But he output is not pure DC there is some AC Components and little ripples in to the output of the Bridge rectifier. These AC components are filtered by introducing a capacitor at output and then this voltage is to be regulated by a regulator LM7805 which gives a pure 5VDC. The circuit is shown in figure 3.8.

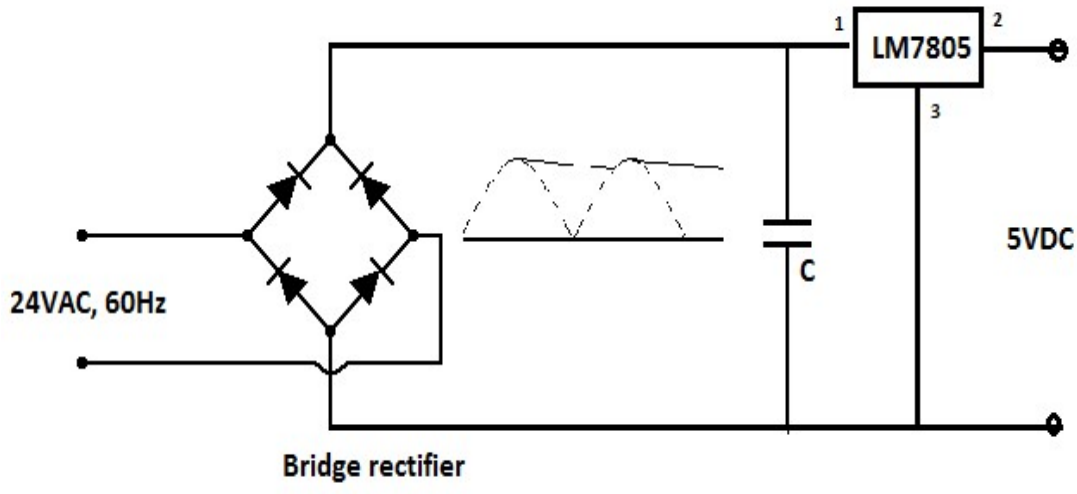


Figure 3.8:- Circuit to convert 24VAC to 5VDC

3.4.2 SMPS

These there are switching mode power supplies readily available. They are capable to tune the voltage to the required level. Not only the voltage even the current can be adjusted depending on the requirement. Figure3.9 shows the SMPS in the lab, just adjusting the nobs we can adjust the voltage. But at a time we can only get one output. 5Vdc and 12Vdc both are needed as discussed earlier. But fortunately the L298N will generate 5VDC which is used for energizing Quadrature encoders and Arduino.

Figure 3.9: SMPS at lab (MCERC lab 122)

Chapter-4

Software Installation and Configuration

4.1 Measurement & Automation Explorer

Measurement & Automation Explorer (MAX) provides access to your National Instruments CAN, DAQ, Field Point, GPIB, IMAQ, IVI, Modular Instruments, Motion, NI Switch Executive, VI Logger, VISA, and VXI devices. Like other NI software products, NI-DNET uses MAX as the centralized location for all configuration and tools.

4.1.1 MAX

Following are few aspects that can be achieved through MAX implementation:

- Configure your National Instruments hardware and software
- Back up or replicate configuration data
- Create and edit channels, tasks, interfaces, scales, and virtual instruments
- Execute system diagnostics and run test panels
- View devices and instruments connected to your system
- Update your National Instruments software

This document is needed to learn how to install MAX and use it to configure ones VXI system.

4.1.2 Prerequisites for Installation

- MAX automatically installs with the NI-VISA version 2.5/higher or NI-VXI version 3.0/higher.
- MAX is available for only Win-32 based Operating Systems.
- MAX cannot be downloaded by itself.
- It can be downloaded as a package with National Instruments drivers (NI-VISA, NI-DAQmx, etc.) and in the NI System Configuration package.

4.1.3 Configuring MAX

- Select **Measurement & Automation** under **National Instruments** in **Programs.**
  - **Start → Programs → National Instruments → Measurement & Automation**
- The program can also be launched by double clicking on the desktop icon, shortcut for the same.
- After the above step, MAX displays the below dialog box. This dialog box is used to configure MAX to search for new devices each time this is launched.
  - This dialog can also be configured to appear next time MAX is launched with the help of the check box (See in the dialog box Figure 4.1).

Figure 4.1: Dialogue box at before MAX launch

  - In the dialog box above, the first radio button is selected which means MAX is not configured to show updated system view when there are any changes.
  - Hence the system has to be refreshed after performing any changes.
  - If these preferences have to be changed, this dialog box can be accessed by selecting **User Preferences** under **Tools** option.

- In the **Configuration** section on the left hand side, for the **Devices and Interfaces** option under **My System**, click on the plus (+) sign to expand the options further down. From the resulting options, expand the **VXI System 0** option too in the same way by clicking on the plus sign (+) and then expand **Frame 0**. The MAX interface appears as in the below screenshot in Figure 4.2:

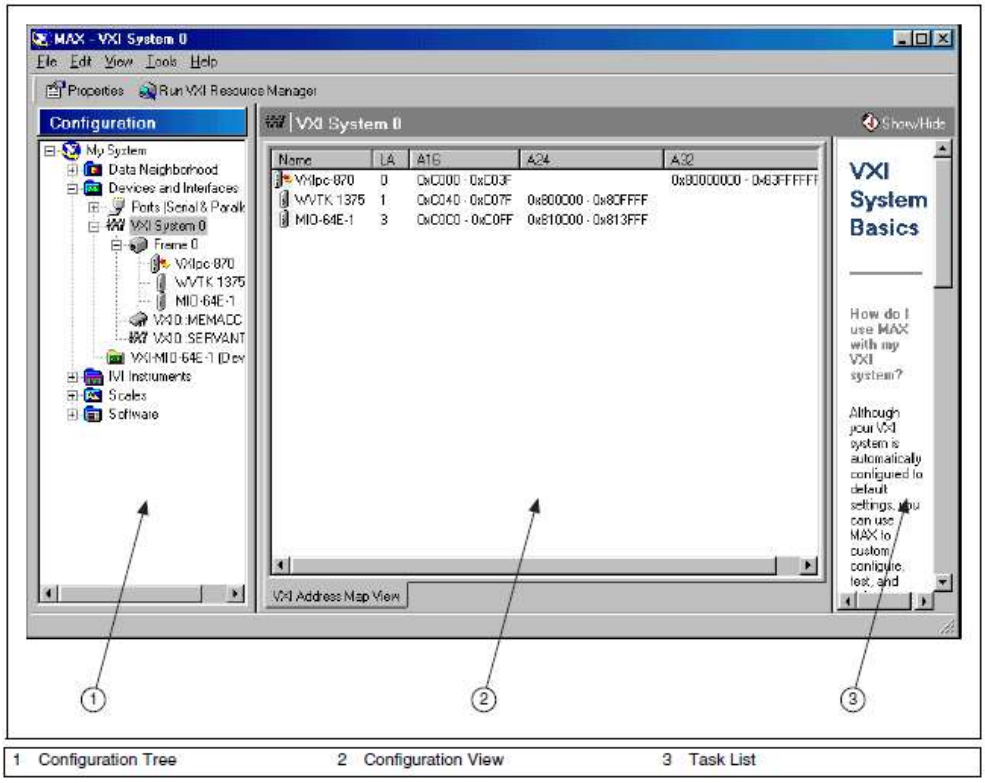


Figure 4.2: MAX launch configuration

Configuration Tree: The Configuration tree shows a hierarchical view of the instruments in the system. In the above figure, MAX shows below instruments:

- VXI System with one VXI chassis (Frame 0)
- A National Instruments VXIpc-870 embedded controller
- A Wavetek 1375 arbitrary waveform generator
- A National Instruments MIO-64E-1 multifunction data acquisition board

➢ The below MAX toolbar appears when the VXI system is selected in the configuration tree.

- The **Properties** from the toolbar displays properties of the selected VXI system.
- Run VXI Resource Manager runs VXI Resource Manager (Resman) for the selected VXI system.

- NI-VXI 3.0 or higher supports multiple VXI systems and both the properties and Resman apply to only the currently selected VXI system. To check properties and run Resman for all VXI systems, select **Tools → NI-VXI → VXI Resource Manager**.

System Properties:

- **Properties** option can be selected from the tools bar to view the properties for the selected VXI system. Properties can also be viewed by right clicking on a specific VXI system as shown in figure 4.3 and selecting **Properties** option from there.

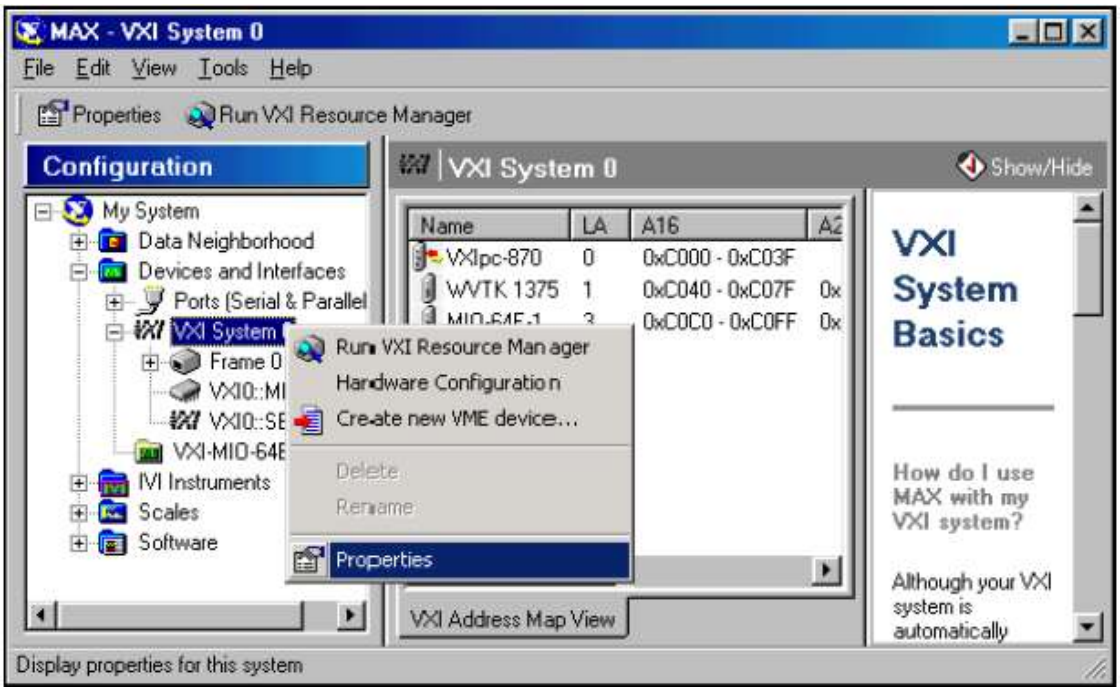


Figure 4.3: MAX system properties

- Additional options when right clicked on VXI system include:
  - **Run VXI source Manager** to launch Resman
  - **Hardware Configuration** to access hardware configuration for the system controller
  - **Create new VME device** to create a VME device as the name suggests.
- View the properties for the VXI system 0 by right clicking and selecting **Properties**. Below in the figure 4.4, is the Properties dialog box that shows up:

Figure 4.4: Setting MAX system general properties

- Following options can be viewed further under **General** tab:
  - Device type is displayed.
  - Device status is mentioned.
  - There is a **Troubleshoot** option to fetch online help when there is a problem with the system.
  - Device enabled status is shown at the bottom.
- Select Settings tab - the second tab, which is as shown in Figure 4.5 below:

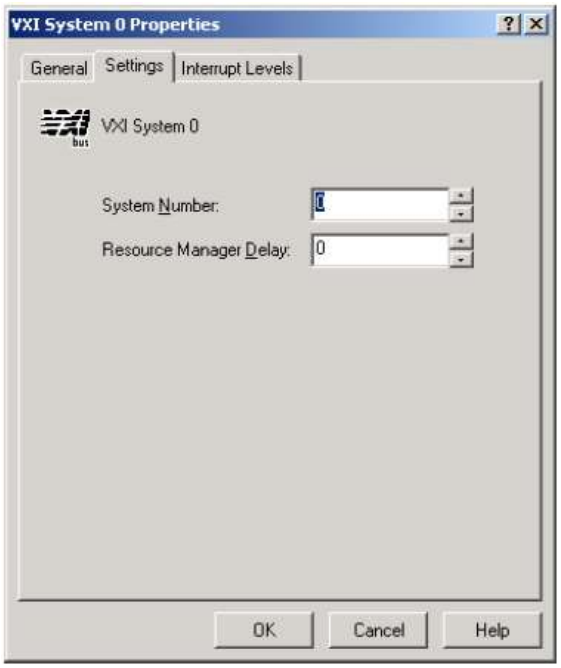


Figure 4.5: Setting MAX system properties

- Following points can be inferred from the above **Settings** dialog:
  - VXI system number and Resman delay for a specific VXI system can be changed.
- Select the **Interrupt Levels** tab which is as in figure 4.6 below:

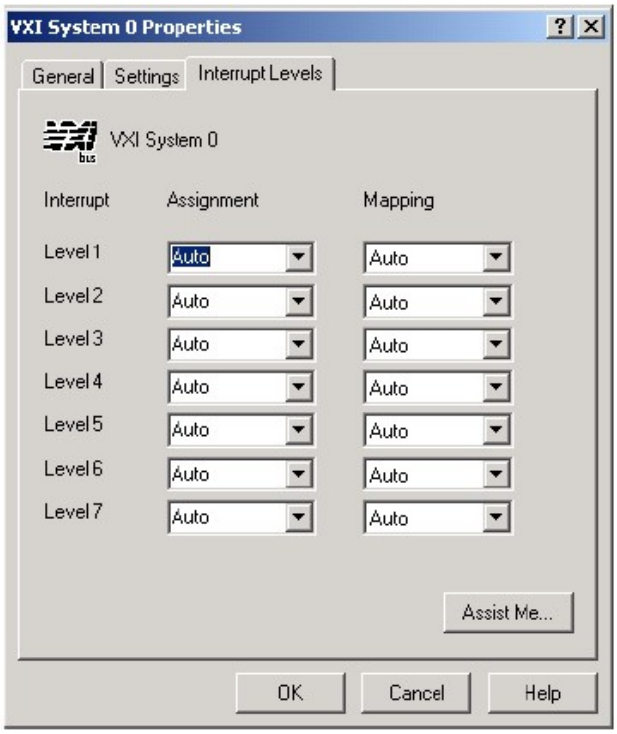


Figure 4.6: Setting MAX system properties interrupt level settings

- This tab can be used for the following:
  - Assign interrupts to specific controllers and set mappings for each interrupt level.
  - For the Assignment field, below 4 options can be selected from the drop down menu:
    - **Auto** – Automatically assign to a programmable handler.
    - Local – Assign interrupt level to a local controller.
    - Choose LA – Choose logical address for the controller that will handle the interrupt.
    - None – Do not assign the interrupt level to any controller.
  - For the Mapping field, we have following options in the dropdown:
    - Auto – Map automatically to root if root is the handler, else does not map.
    - Source – Map away from the root (Source the interrupt)
    - None – Do not map.
  - On clicking on **Assist Me**, **Interrupt Levels** dialog box appears as given in Figure 4.7 below:

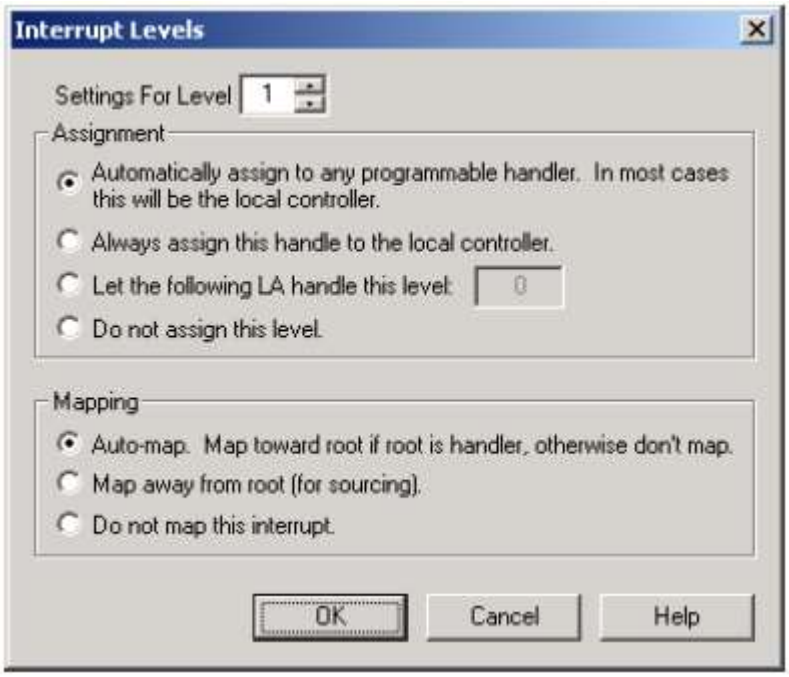


Figure 4.7: Setting MAX system interrupts levels properties

- This dialog box can be used for detailed descriptions of all settings and to set options separately for each interrupt.

Configuration View: Configuration view shows registers/address spaces requested by various devices in A16, A24 and A32 space. [For further details - 999]

- Various items in the configuration tree can be selected to review the information in the task list. The task list contents change according to the interface and device selected.

4.1.4 Device Properties in MAX

- Select Wavetek 1375 in the configuration tree. It provides information on the device in the **Attributes** tab of the configuration view shown as figure 4.8.
- Properties information for the Wavetek 1375 can be reviewed by right clicking on Wavetek 1375 and selecting **Properties**.

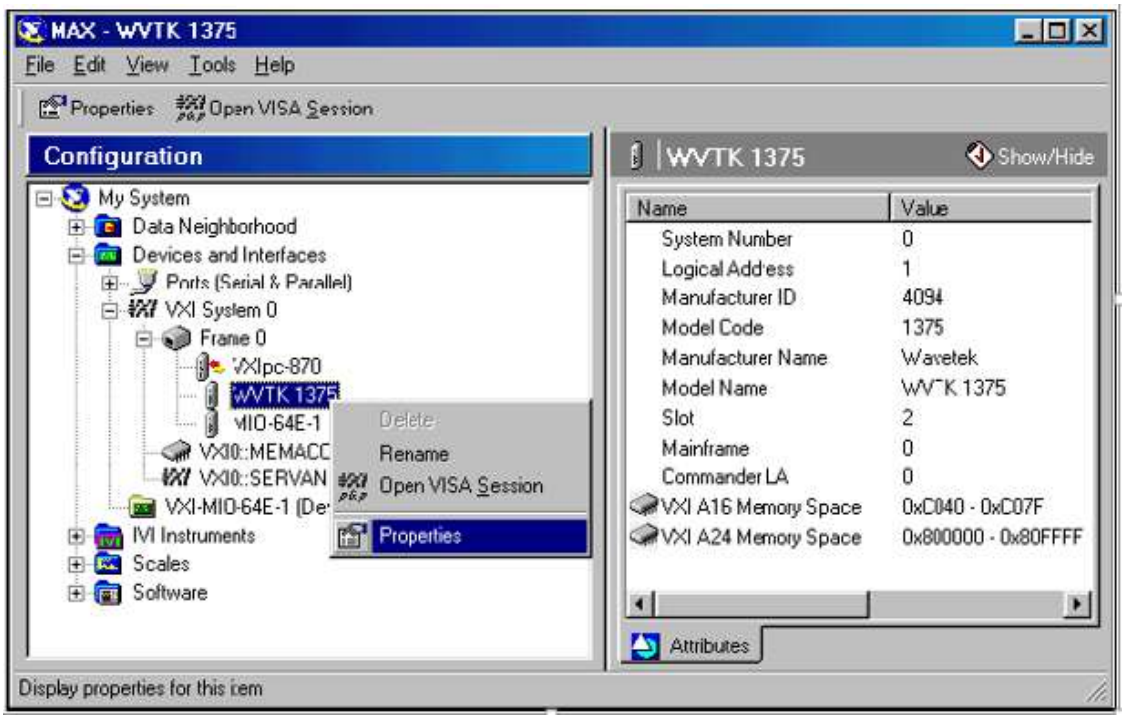


Figure 4.8: Setting MAX Wavetek 1375 properties

4.1.5 Configuring Devices in MAX

- VXI devices can be configured through the software. MAX can be used for that purpose.
- Right clicking on the VXIpc-870 controller in the configuration tree and selecting **Hardware Configuration** shows the below dialog box figure 4.9:

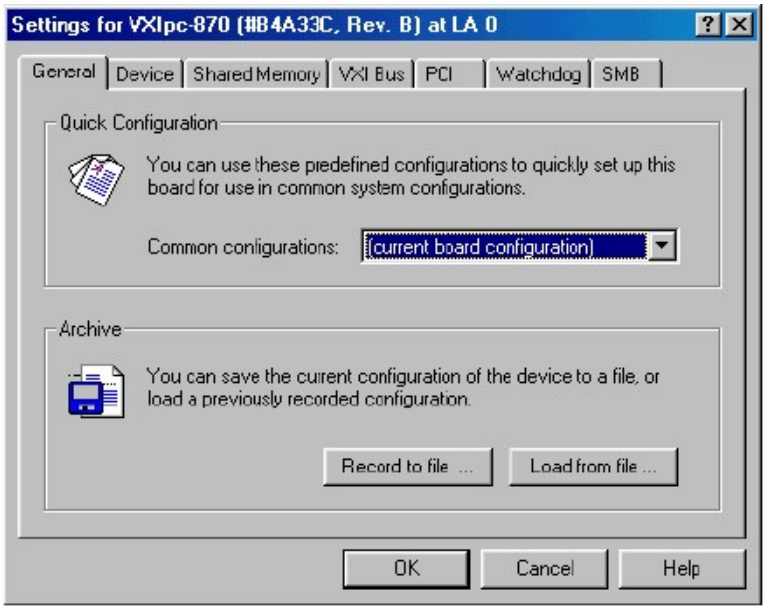


Figure 4.9: Setting for VXIpc-870

- Each tab starting from **General** to **SMB** allows to carry out specific activities/configure for the hardware related to specific controller. [For further details - 999]
- By selecting the options **Tools → NI-VXI → VXI Options**, MAX can be customized to meet the system needs through the VXI options displayed. Below is the screenshot in figure 4.10:

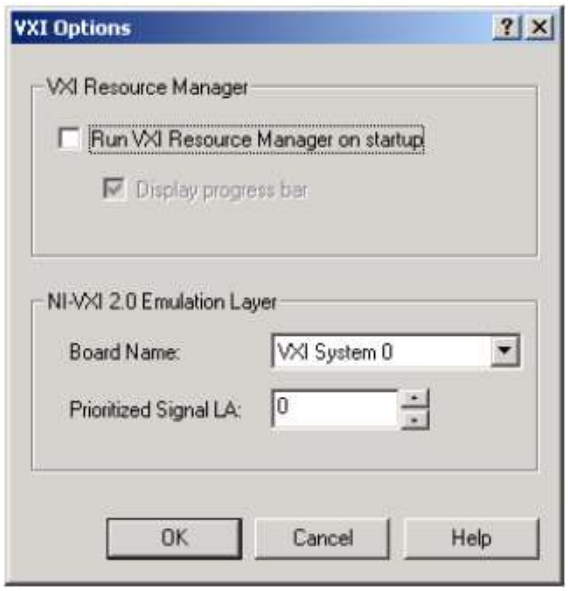


Figure 4.10: VXIpc Options

- With the help of the options **Tools → NI-VISA → VISA**, NI-VISA options can be set and driver can be set to run Resman on startup. Below is the screenshot in figure 4.11:

Figure 4.11: Setting Visa options

4.1.6 Help in MAX

- Online support from MAX helps with common questions that are collated from:
  - Several years of technical support calls
  - Helpful hints
  - References to application notes
  - Procedures to setup large variety of systems.
- If the MAX is residing on a network, there is a web link offered that links to the technical support home pages of National Instruments, NI-VISA and NI-VXI.
- Following can be accessed for help and information from the above links:
  - ***Online Problem Solving and Diagnostic Resources*** that include:
    - Knowledge Base
    - Ttroubleshooting wizards
    - Product Manuals
    - Hardware Reference Databases
    - Application Notes
  - ***Software Related Resources*** that include:
    - Instrument Driver Network

- Example Programs Database
- Software Library

➢ For further help, we can get it from Help → Help Topics → NI-VXI

4.1.7 Software Information

➢ Information about the system can be viewed by *Expand Software folder* in the Configuration Tree → Expand NI-VISA icon → Select visa32.dll.

➢ The below window in figure 4.12 appears with the Software information:

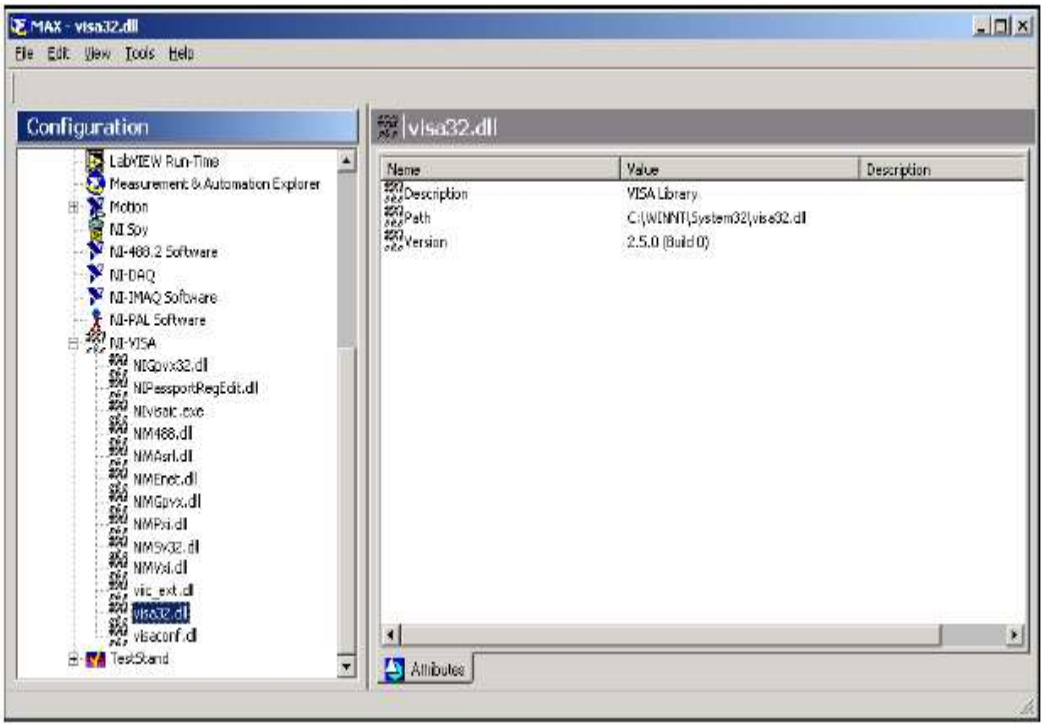


Figure 4.12: Setting Visa32.dll options

➢ Information on all the National Instruments versions of the drivers in the system is viewed here.

➢ This information aids during troubleshooting of the system to identify for any gaps.

4.2 LabVIEW Installation

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a system-design platform and development environment for a visual programming language from National Instruments.

4.2.1 Prerequisites:

- Following are the prerequisites to be met to install the **LabVIEW 2014 Development Environment** in the Windows Operating System based computer, that is given in table 4.1:

Table 4.1: LabVIEW 2014 Development Environment

Windows	Run – Time Engine	Development Environment
Processor	Pentium III/Celeron 866 MHz or equivalent	Pentium 4/M or equivalent
RAM	256 MB	1GB
Screen Resolution	1024 x 768 Pixels	1024 x 768 Pixels
Operating System	Windows 7/Vista (32-bit and 64-bit) Windows XP SP3 (32-bit) Windows Server 2003 R2 (32-bit) Windows Server 2008 R2 (64-bit)	Windows 7/Vista (32-bit and 64-bit) Windows XP SP3 (32-bit) Windows Server 2003 R2 (32-bit) Windows Server 2008 R2 (64-bit)
Disk Space	353 MB	3.67 GB (includes default drivers from the NI Device Drivers DVD)
Color Palette	Not Applicable	LabVIEW and the LabVIEW Help contain 16-bit color graphics. LabVIEW requires a minimum color palette setting of 16-bit color.
Temporary Files Directory	Not Applicable	LabVIEW uses a directory for storing temporary files. National Instruments recommends that you have several megabytes of disk space available for this temporary directory.
Adobe Reader	Not Applicable	You must have Adobe Reader installed to search PDF versions of all LabVIEW manuals.

- Following are the restrictions for installing or using LabVIEW on windows:
    - LabVIEW is not supported in the Windows 2000/NT/Me/98/95 Operating systems and in Windows XP 64 bit systems.
    - LabVIEW cannot be used or installed in Windows systems by logging in with a Guest account.

4.2.2 Installation Procedure:

- Insert the LabVIEW platform installation DVD and run it.
- Click on the **Install LabVIEW 2014** option.
- When prompted for a serial key, enter the serial key of the corresponding version of the LabVIEW and continue.
- Provide the folder where the LabVIEW is to be installed when prompted and continue further.
- Continue the installation by selecting the default options listed and at the end in the **License Agreement** page, be sure to check the license terms check box and click on Next.
- There is a prompt to install the hardware support which is the device drivers. This is needed if there is any external software being used or to be used. Else click decline and continue.
- This completes the installation.
- There is a prompt to restart the system. Continue with the restart and open the application to start using it.
- There may be a need for additional modules/toolkits to start using the particular version of LabVIEW installed.
- To check and confirm if the system meets minimum requirements to start using the LabVIEW, locate the module or toolkit readme in the LabVIEW Platform readme file on the installation DVD and clarify the same. The file is named as readme_platform.html.

4.3 Installing and Using NI cRIO-9074

Description cRIO (CompactRIO) is a real-time embedded industrial controller made by National Instruments for industrial control systems. The CompactRIO is a combination of a real-time controller, reconfigurable IO Modules (RIO), FPGA module and an Ethernet expansion chassis.

4.3.1 Prerequisites for Installation

- LabVIEW 8.6 version or the latest, not any earlier LabVIEW

- MAX (Measurement & Automation Explorer) software installed with LabVIEW 8.6
- NI cRIO-9074 chassis and its power cord
- cRIO software installation disk (NI-RIO)

4.3.2 Installing cRIO – Procedure

- Install LabView 8.6/ later versions and NI-RIO. MAX (Measurement & Automation Explorer) should be automatically installed with LabView 8.6/ later versions, and is the program you will use to initialize the cRIO. NI-RIO is a collection of drivers for the cRIO, and is located on a CD that comes with the cRIO.
- Open Windows Firewall and follow the below steps to add LabView 8.6/ later versions and MAX to Windows Firewall's list of exceptions as shown in figure 4.13.

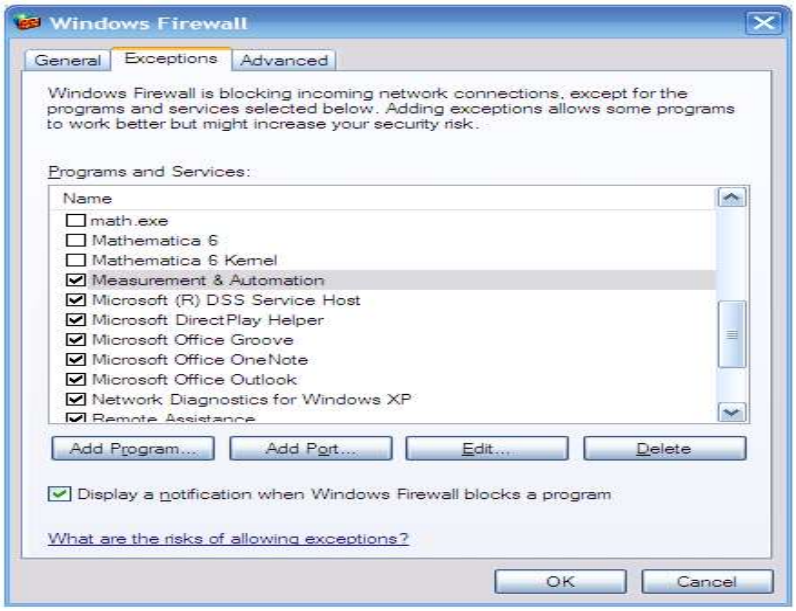


Figure 4.13: Firefox Setting

- Click on **Add Program** and from the resulting set of programs add either or both of the below :
  - National Instruments LabVIEW 8.2/ later versions
  - National Instruments LabVIEW 8.6/ later versions
- Once you get the below screen, it means that MAX is successfully added to Windows Firewall.
- Change your laptop's IP address to the fixed value **192.168.0.1.** For this:
  - Locate the internet connection properties.

○ Go to the Control Panel, open **Network Settings**, then right click on the network connection you are interested in [LAN] and choose **Properties.** The below screen given in figure 4.14 pops up :

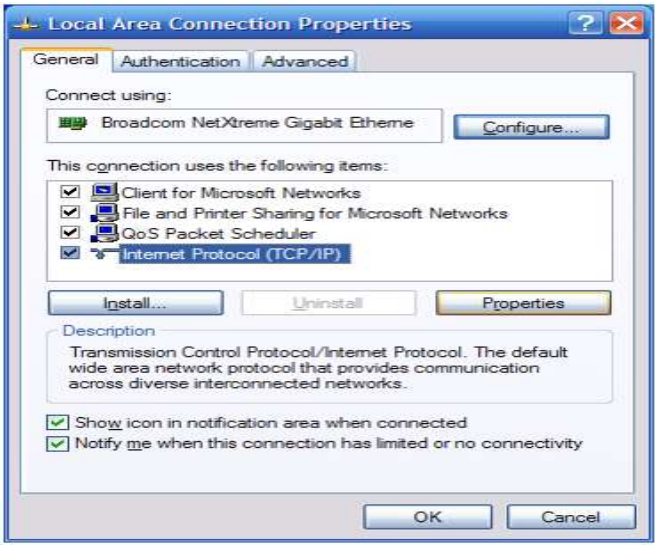


Figure 4.14: LAN settings

- Select **Internet Protocol (TCP/IP)** in the window above and click Properties.
- Choose **Use the following IP address** and provide the IP as 192.168.0.1.
- Once communicating with the cRIO is completed, change the setting back to **Obtain an IP address automatically** shown in figure 4.15.

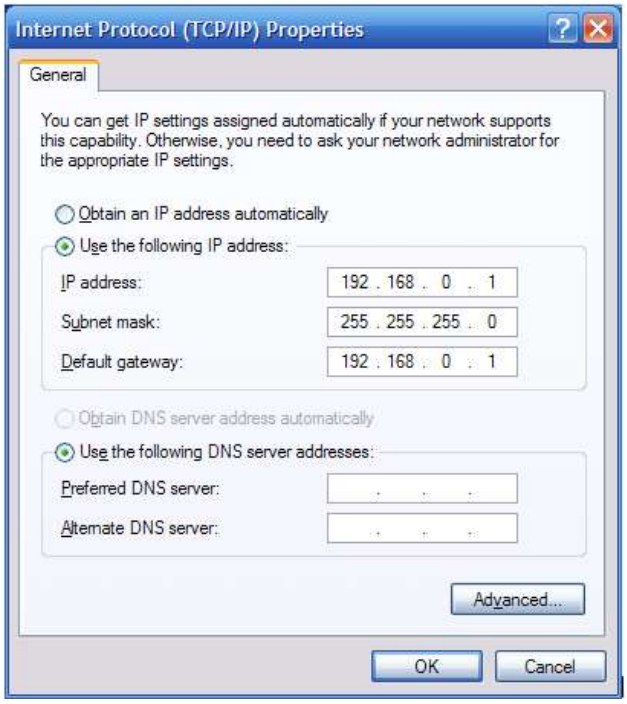


Figure 4.15: IP settings

- Connect the cRIO chassis to the system with an Ethernet cable and connect the power cord to the cRIO to power on the cRIO.
- Set the **IP RESET** switch on the cRIO to the **ON** position and press the **RESET** button on the cRIO; this resets the cRIO and assigns it a fixed IP address of 0.0.0.0.
- Once the cRIO is on, set the **IP RESET** switch back to the **OFF** position. If not, the cRIO will reset it's IP address every time it is powered on or reset.
- Open MAX and press **F5** to refresh the left display pane. MAX should find the cRIO and display it in the left display pane.
- Select the respective cRIO in the left pane of MAX and using the right pane, change the following settings:
  - cRIO IP = **192.168.02**
  - cRIO gateway = **192.168.0.1**
  - Subnet mask = **255.255.0.0**
- Click apply to save these changes. This will change the cRIO's IP address.

5 Experiments and Results

5.1 Identification of power cables

The power cable identification for CRSPLUS A150 is the first task to get robot to work. The identification and labeling of the power cables was done by Mr. Abdulrahman Alshankiti (Grad student of ISU, who worked on this thesis earlier). Later necessary changes to cable labeling are done for convenience. The main moto of writing this as a part of thesis report is to give the future students a clear idea about the cables and fro ease of axis, so, that they may not need to start research over the cables to the axis motors.

Unlike other Axis, Axis-1 has a separate cable. It is the only cable that has both power and encoder wires together. Generally encoder cables are not combined with power cables to protect the Encoder signals. Also a part of industrial standards, any sensor cables are generally shielded to protect encoder signals from interference from other signals or power transmission lines and to reduce the effect of noise from surrounding environment. For instance, a Japan based company by name Fanuc adopted optical communication for all the feedback signals in its machines including encoder signals from motor. For safety it is recommended power cables with different power levels are to be run with 90 degree angle between the cables. A current carrying conductor will get will have magnetic energy. Axis-1 cable is labeled with its axis number as shown in figure 5.1.

Figure 5.1: Axis-1

But the other Axis motor power connection wires are all together in a single cable. Identifying each power cable for an axis is an easy task, first a cable is selected then power of 12VDC is given directly to the cable. Now the ground cable from the 12VDC is

connected to the rest of unknown cables till motor rotates and then pair of cables can be identified. They are well labeled with their axis numbers/joint numbers as shown in figure 5.2.

Figure 5.2: Motor power cables

We need not do any calculations for the dimensions of the cable the manufacture should have already taken care of these. But when we connect them with through another wire to our drive or controller care must be taken for the dimensions of the cable being used. The following are the formula to calculate diameter/ size of the cable, for both single phase and three phase circuits.

Calculating Wire/Cable Size formula for single Phase Circuits

$$\text{Wire Circular mils} = \frac{2 \text{ x } \rho \text{ x I x L}}{\%\text{Allowable Voltage drop of source voltage}}$$

Calculating Wire/Cable Size formula for Three Phase Circuits

$$\textbf{Wire Circular mils} = \frac{\sqrt{3}\,\textbf{X2 x } \boldsymbol{\rho} \textbf{ x I x L}}{\%\textbf{Allowable Voltage drop of source voltage}}$$

Where;

ρ = Specific resistance of the metal used

D = half the total circuit length

I = Load Current in Amps

% Allowable Voltage drop of source voltage should not be more than 2.5% according to according to IEEE rule B-23.

Note: the Value of ρ = Specific resistance or resistivity of Conductor is used here for copper and aluminum is 11.2 and 17.4 respectively at 53° C (127° F).

5.2 Identification of encoder cables

Identification of encoder cables is not as simple as it looks and also very time consuming process. The complexity is each encoder has 5 cables to be identified as tabulated in table 5.1.

Table 5.1: Encoder pin description

SL No.	Encoder cable
1	+5V DC
2	GND
3	A
4	B
5	Z

Table 5.1: List of cables for each encoder

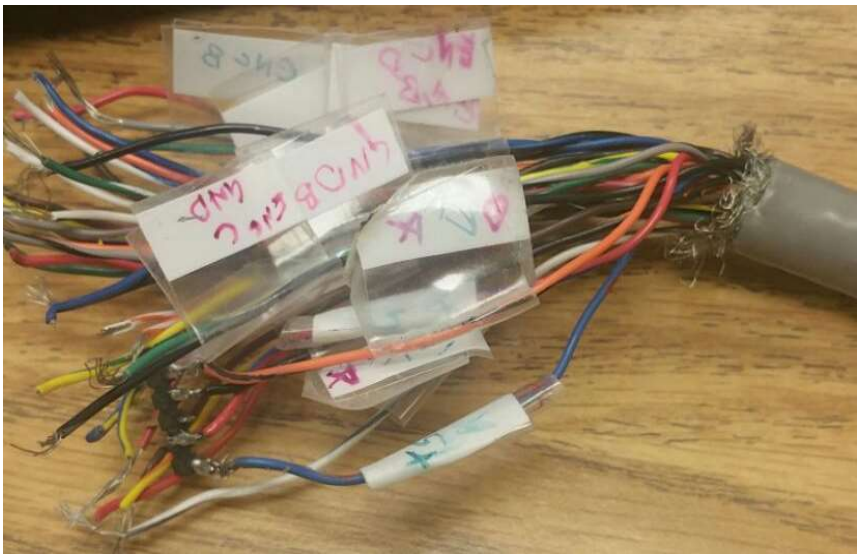


Figure 5.3: Encoder cable

The encoder may break if 5Vdc is given a wrong pin/wire. To avoid any mishap, a different approach is followed. First the 5Vdc and ground cable of each encoder is identified. This is done by opening the cap of the encoder, then identify the 5Vdc and GND wires(generally Red and Black wires)and then check the connectivity at the point of contact to encoder and end of encoder wire.(note that all cables connected to encoder is then connected to a cable with 57 wires of 0.5mm diameter. So we can't go with the color code of the cable. After find ding the power of each cable A and B are identified for each cable.

5.3 Signal conditioning

5.3.1 Some basics of signal conditioning

Signal condition is an essential aspect of today's industrial automation, measurement and control; that enables us to effectively measure, maintain or improve sensor signals and thus helps in efficiently controlling the system. Signal conditioning converts input signals into the signals acceptable to digitization hardware. Here are some examples where signal conditioning comes in aid:

- Isolation— to provide a protective barrier between sensors and other digital equipment or controller. In specific when handling with high voltage equipment.
- Noise reduction-To prevent noise produced due to ground loops, which decrease signal quality.
- Filtering—Filtering process reduce noise and remove unwanted signal frequencies arising from external sources.
- Amplification—Amplification is a process to increases a signal's amplitude or voltage level. This process increases measurement accuracy of small signals.
- Attenuation- Attenuation is to reduce signal's amplitude. It is equally important as amplification in systems that produce higher voltage sensor signals levels.
- Switching relays—one of the best examples and most used application of signal conducting is switching relays, to control a very high voltage Ac motor with a digital controller is almost impossible without Relays. Relays accept control signals as low as 5VDC and can control a 1000VAC three phase motors.

5.3.2 Encoder signal conditioning

The encoder generates a train of 5V pulses in synchronous to the rotation of the motor. But the NI cRIO can't read signals of 5V, hence can't read quadrature encoder feedback signals. To overcome this situation the feedback signal is to be amplified to a higher level. Operational amplifier (in short Op-amp) can be used to amplify the signal. Op-amp is an integrated circuit that is perfect for DC amplification applications.

Ideally it has infinite input impedance, zero output impedance, infinite bandwidth and zero offset voltage. Using Op-amp in non-inverting mode we can amplify this encoder signals the required level.

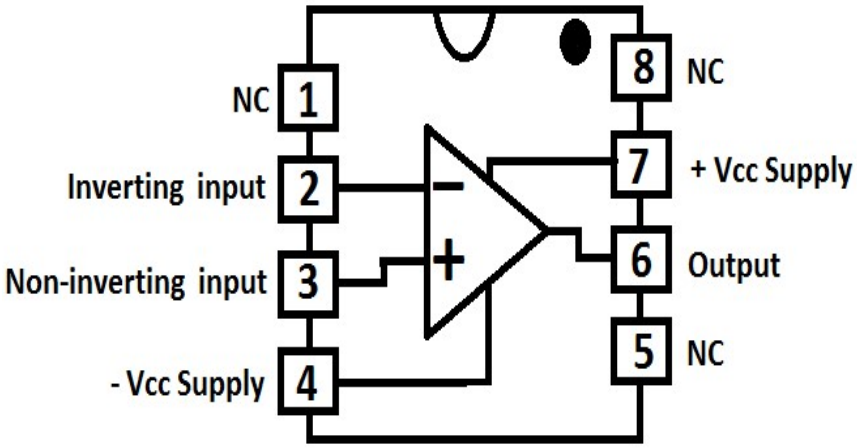


Figure 5.4: Operational amplifier

` For a non-inverting amplifier the input is given at non- inverting terminal that is pin 3 and the output is taken from the pin 6. Pin 2 is grounded with through R1 and feedback resistor is connected between pin 2 and pin 6.

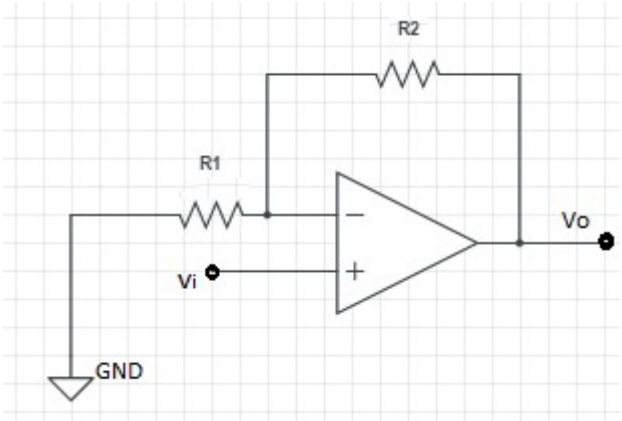


Figure 5.5: Non-inverting Amplifier circuit diagram

The output of the non-inverting amplifier is given by $V_0 = \left(1 + \frac{R_2}{R_1}\right) V_i$ .....Eq 5.1

The input signal is 5V the aim is to double the voltage. This can be achieved by taking $R_1 = R_2$. So let$R_1 = R_2 = 1\text{K}\Omega$.

$$V_0 = (1 + 1)V_i$$

$$V_0 = (2)5$$

$$V_0 = 10\text{V}$$

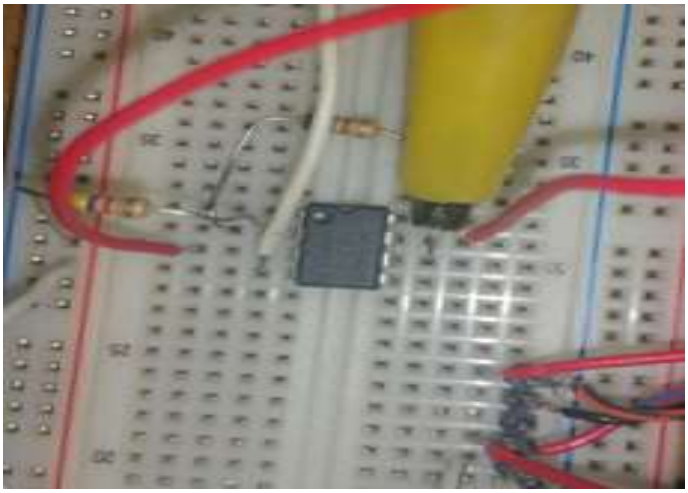


Figure 5.6: Non-inverting Amplifier connections

5.3.3 Other signal conditioning

The operational voltage for motor of CRS PLUS A150 robot arm is 6VDC to 36VDC. But for this thesis we are operating in the range of 6VDC to 12VDC. So the SMPS is set to 12VDC. But the encoder needs 5VDC for energizing it. There are multiple ways to get a constant 5 VDC to supply power to Encoder.

5.3.3.1 Voltage Divider circuit

A simple voltage divider circuit can be used to reduce the voltage from 12VDC to 5VDC. Voltage divider circuit is shown below in figure 5.7

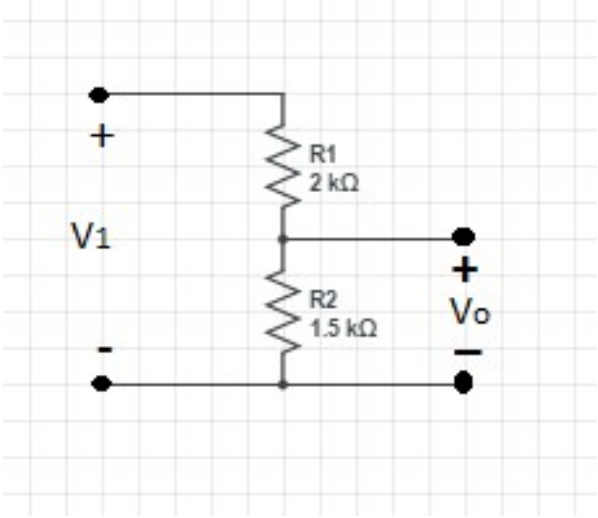


Figure 5.7: Voltage divider circuit

The output voltage Vo is given by the equation $V_{out} = \frac{V_1 R_2}{(R_1+R_2)}$……..Eq 5.2.

$$V_{out} = \frac{12 * 1.5K}{(2K + 1.5K)} = 5V$$

5.3.3.2 Voltage Regulator

The voltage regulator is an IC that will give a constant voltage as output. Generally voltage has three pins. Pin 1 is source voltage, Pin 2 is GND and Pin 3 is output voltage. L78S05CV is a voltage regulator that will give a constant 5VDC output. The input for this regulator cam varies from 5 VDC to18VDC. Irrespective of input the output voltage is always maintained constant.

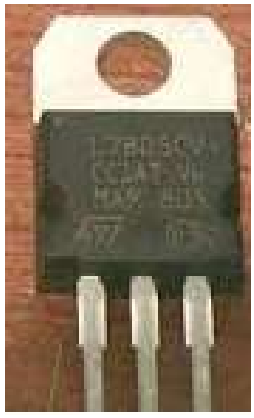


Figure 5.8: Voltage Regulator L78S05CV

5.4 LabVIEW Experiments

Once the LabVIEW is set up and cRIO is successfully connected to the system, we can create projects and write the required code. Go to start on your windows desktop screen and search for LabVIEW, Select the desired LabVIEW version in case if you have more than one LabVIEW versions installed on your system. You can rum both the versions simultaneously with two different codes or by using different instruments at the same time.

Figure 5.9: Starting LabVIEW

Once you double click on LabVIEW, it will initializes plugins and opens LabVIEW as shown in figure 5.1. If one wants to open the previously opened or written code, can go to open existing and select the code needed. Or can create a new project by selecting the needed from the options below **Create Project** soft button.

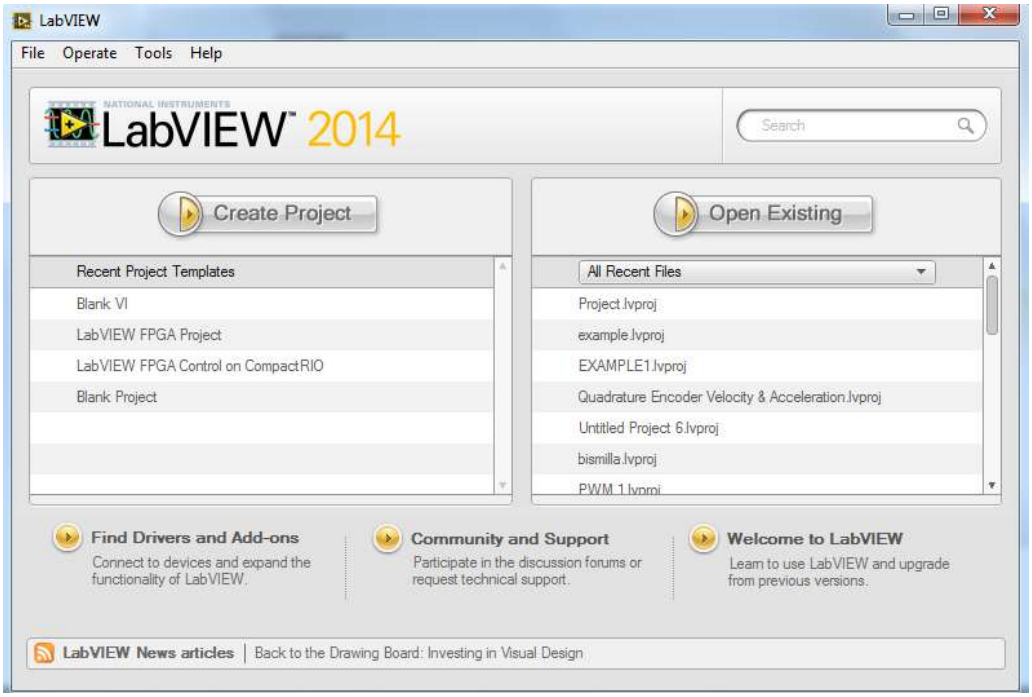


Figure 5.10: Open LabVIEW

Select the LabVIEW FPGA Compact RIO Embedded System or LabVIEW FPGA Project and press next

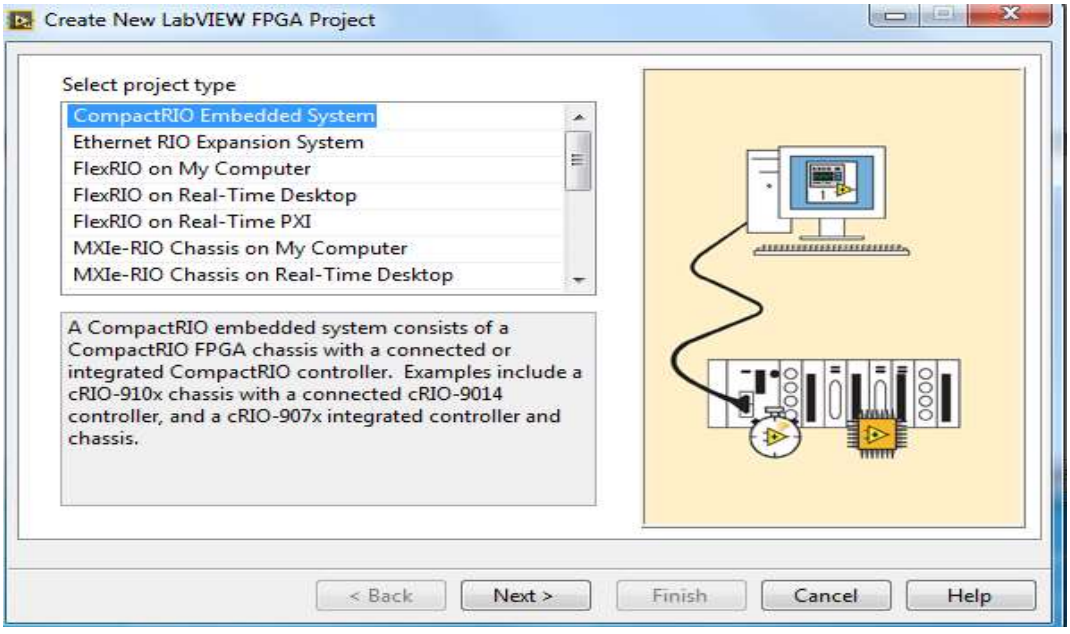


Figure 5.11: Creating new LabVIEW FPGA Project.

Select Discover existing system and click on next. This will enable the computer to detect the NI cRIO that is already physically connected to the computer.

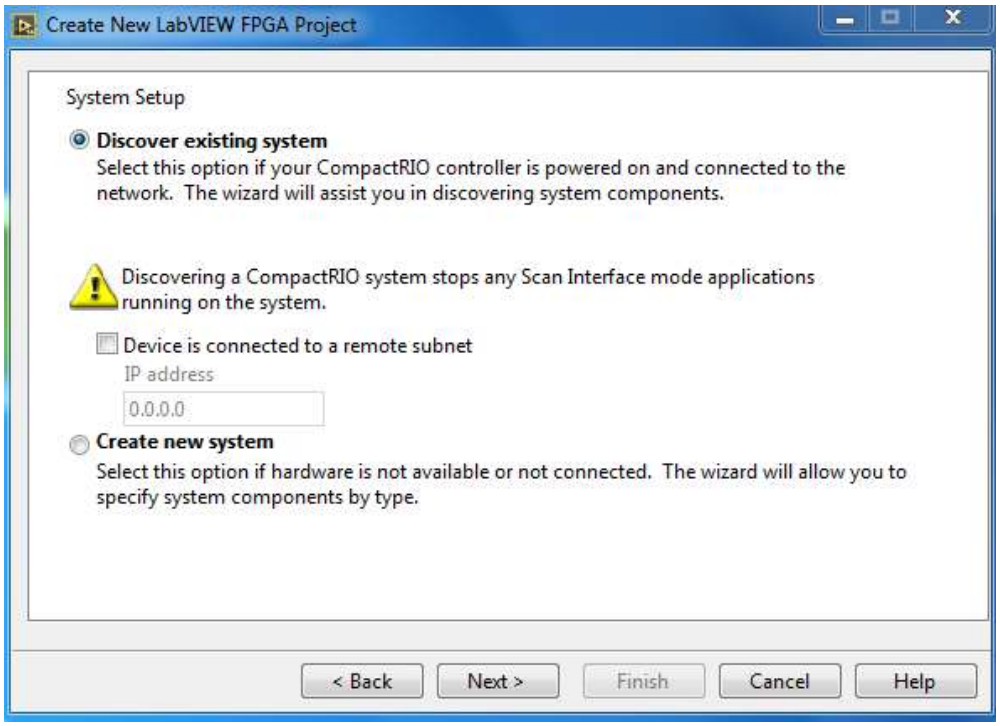


Figure 5.12: Discover existing system.

Once the system is detected it prompts to select the device that we want to establish connection. We only have one cRIO connected to computer so select the NI cRIO9074 and lick next button on the screen.

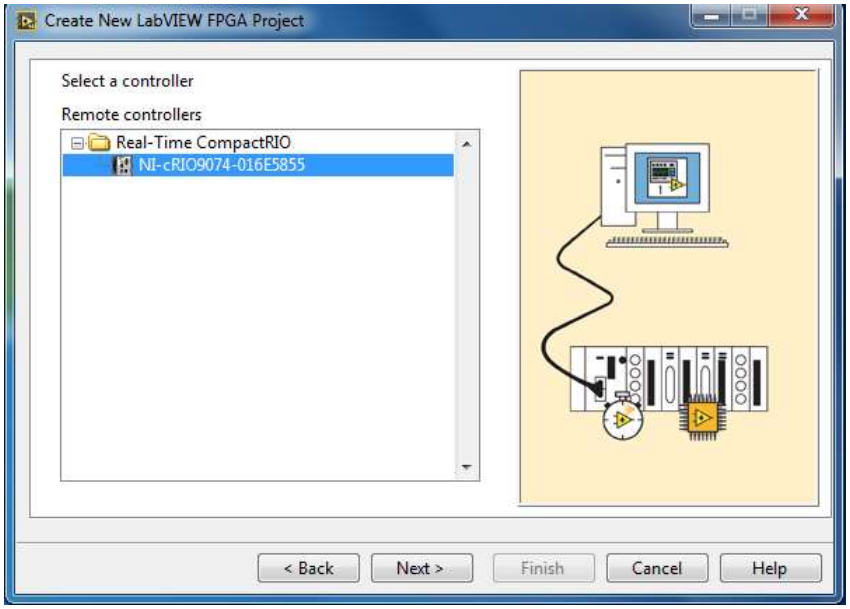


Figure 5.13: Selecting NI CRIO as controller

Now the connection is established successfully, click on finish and project will be readily available.

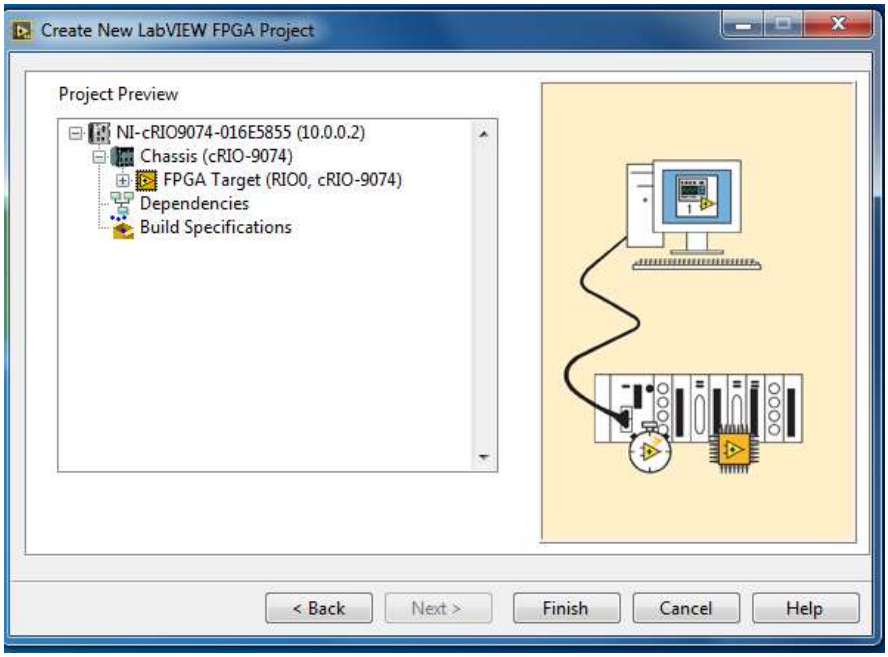


Figure 5.14: LabVIEW FPGA Project Preview

The figure 5.15 shows the new project that is created; verify if the modules on cRIO are identified correctly in the project window both FPGA and normal LabVIEW code can be written. Care must be taken, before writing code as code must be written in FPGA Target if inputs or outputs on the modules of cRIO are used. Once the code is written; it has to be saved and compiled before running the code. Code written in a project is saved to project.

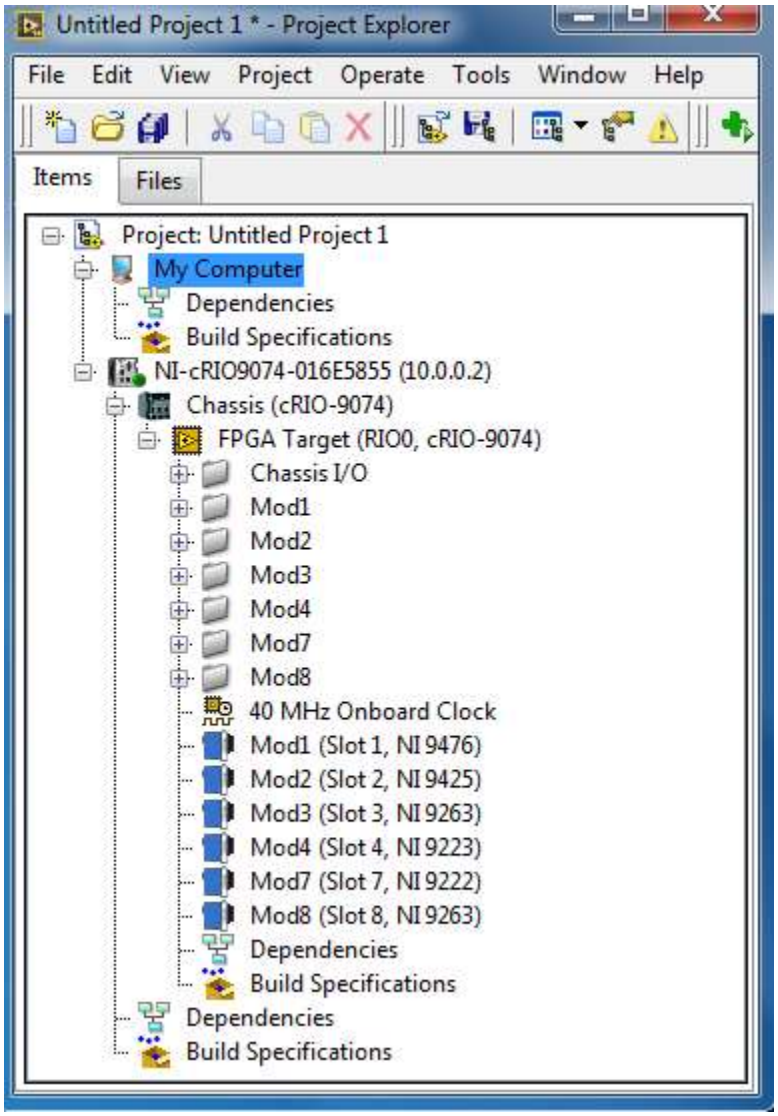


Figure 5.15: Project Explorer window

Go to file and select a new Vi or by the short key Ctrl+n. to create a new FPGA target VI, click on FPGA Target on the project explorer window and select new VI from there.

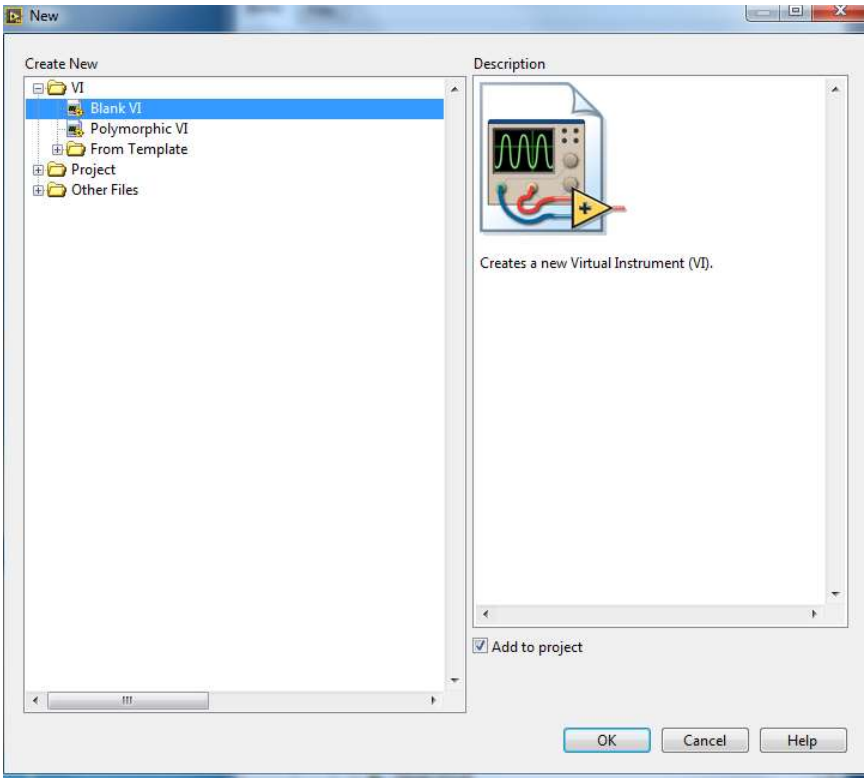


Figure 5.16: Creating new Vi file.

Once the new VI is selected, two windows called front panel and Block diagram will pop up as shown in figure 5.17. LabVIEW code is written in Block Diagram window and all the controls and indicators of the code will be on front panel.

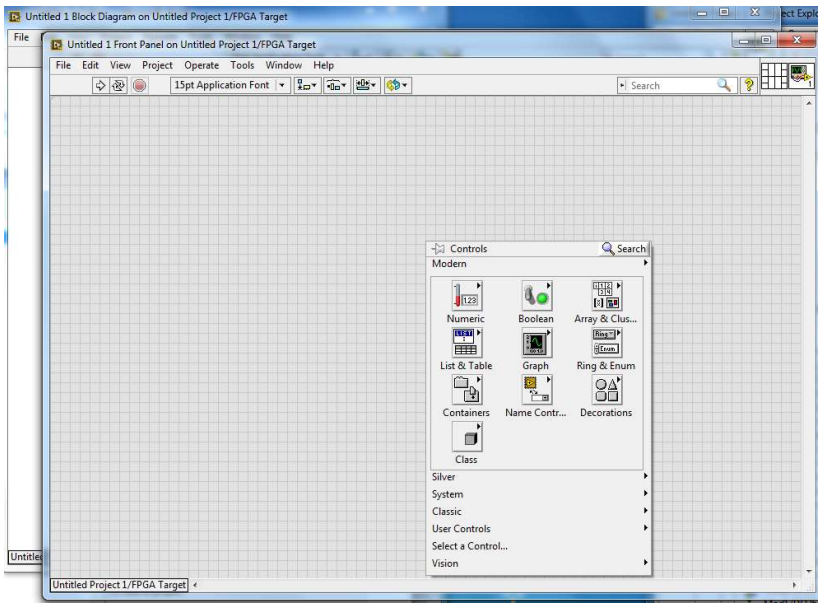


Figure 5.17: LabVIEW Front panel and Block diagram.

In figure 5.18 an example code is written. The code is to on the virtual LED when the virtual pushbutton is pressed.

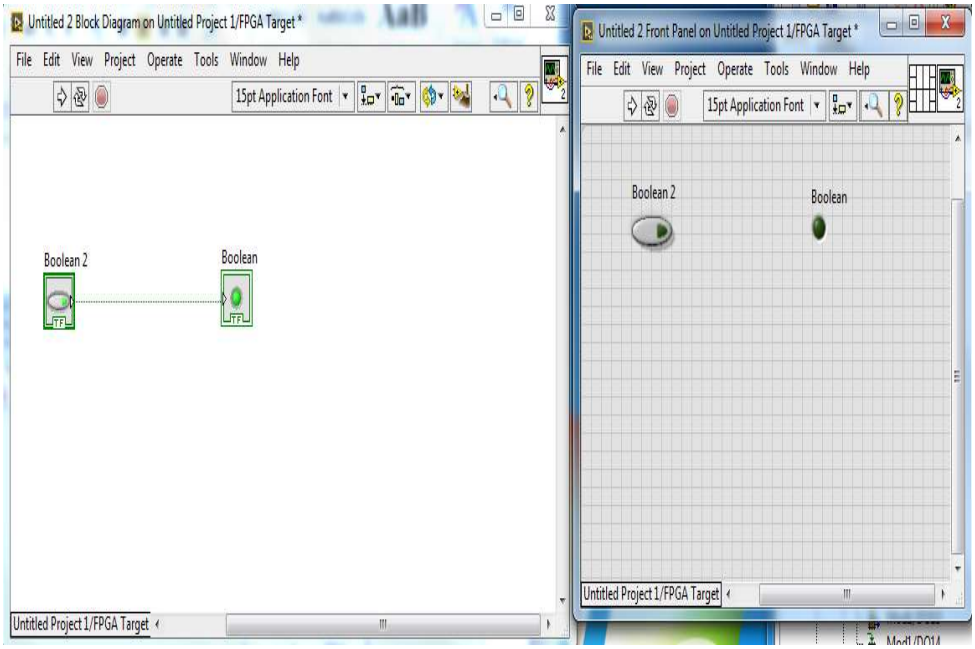


Figure 5.18: Example to LabVIEW code.

One the code is done, before the code is run, both project and the code must be saved.

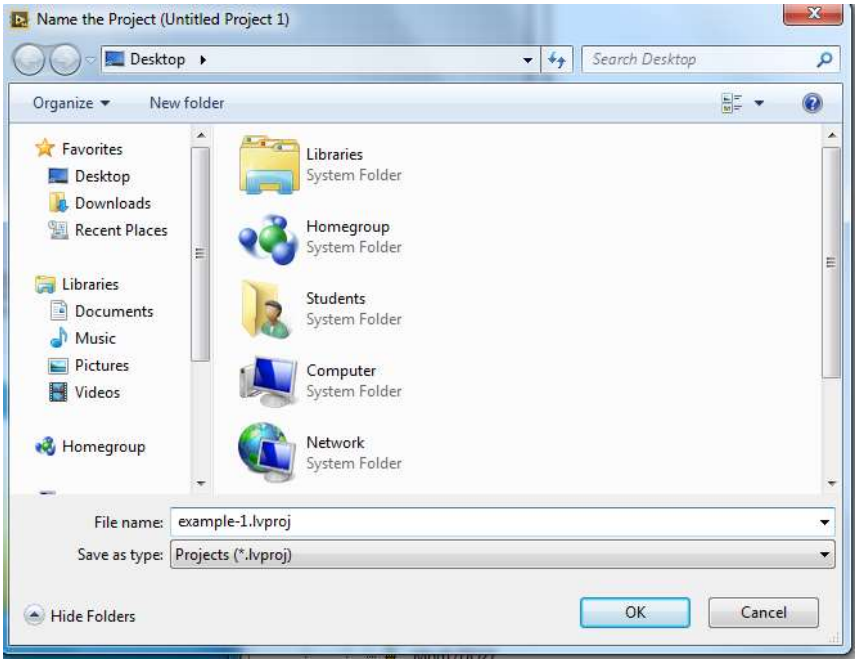


Figure 5.19: Save LabVIEW project

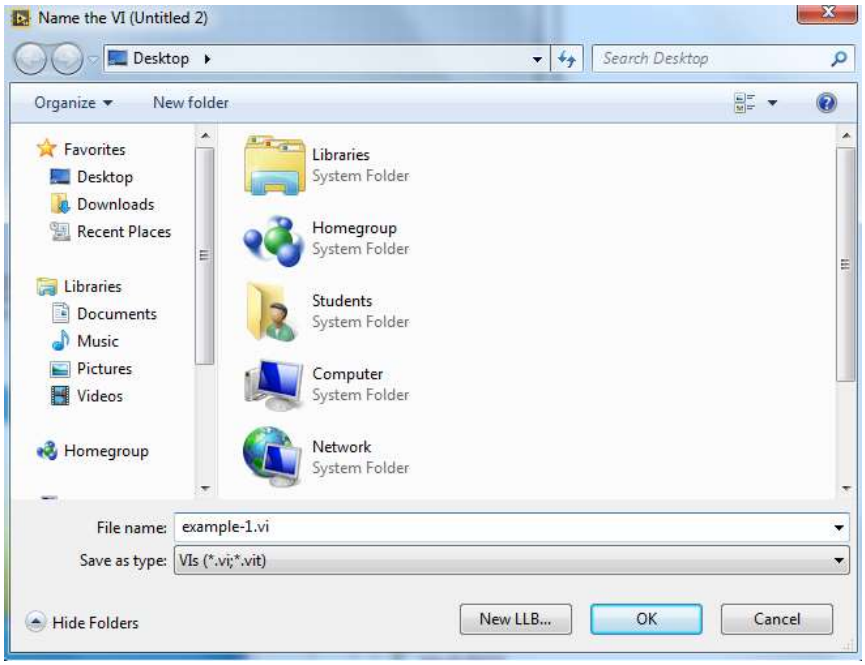


Figure 5.20: Saving LabVIEW Code

The saved code is then complied. Once the compilation is done the code is ready to run.

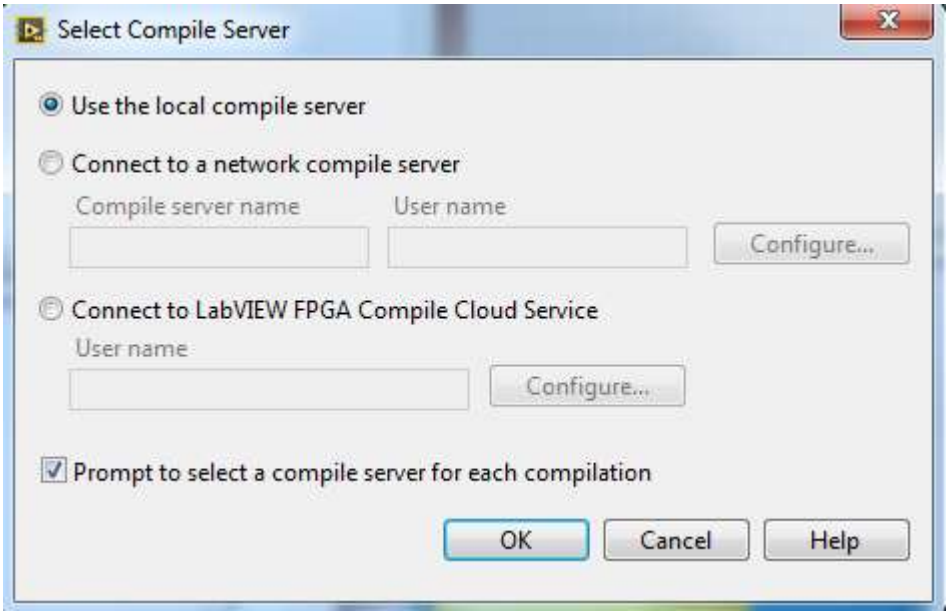


Figure 5.21: LabVIEW Compile

In the figure5.22 the code is compiled sucessifully and is running, even if a minor change done to the code later, the code must be compiled again

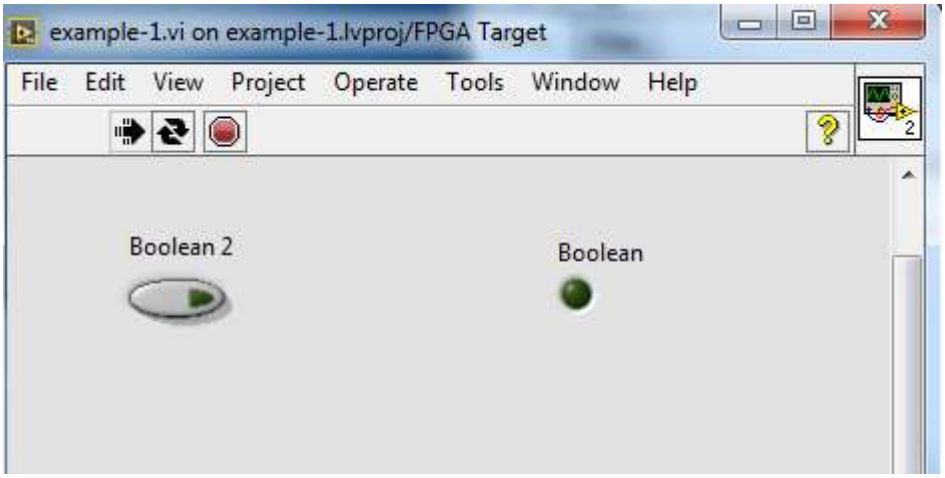


Figure 5.22: LabVIEW ready to run

In figure 5.22 the LED is off as the Switch is off. Where as in figure 5.23 when the switch is togelled the LED is on.

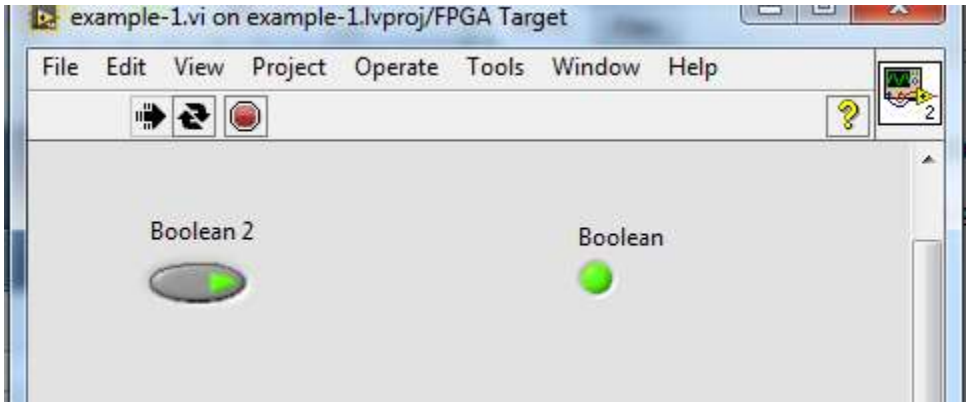


Figure 5.23: Reading virtual pushbutton and energize virtual LED

Code to test if the cRio is ready and connected successfully. Care must be taken while connecting the wires. The details of connections are given the manual also in the chapter 3 of this report. Note that there must be a common ground otherwise the cRIO will not be able to read the Digital input signals.

In the code in figure 5.25, DI23 and DO23are selected and when you give an input of more than 5VDC is given to DI23 the virtual LED should switch to ON state. LED will continue to be in ON state as long as the input is given to DI23. Similarly, when the

virtual pushbutton is toggled we should get an output at DO23. A stop button present in the code is used to break the loop.

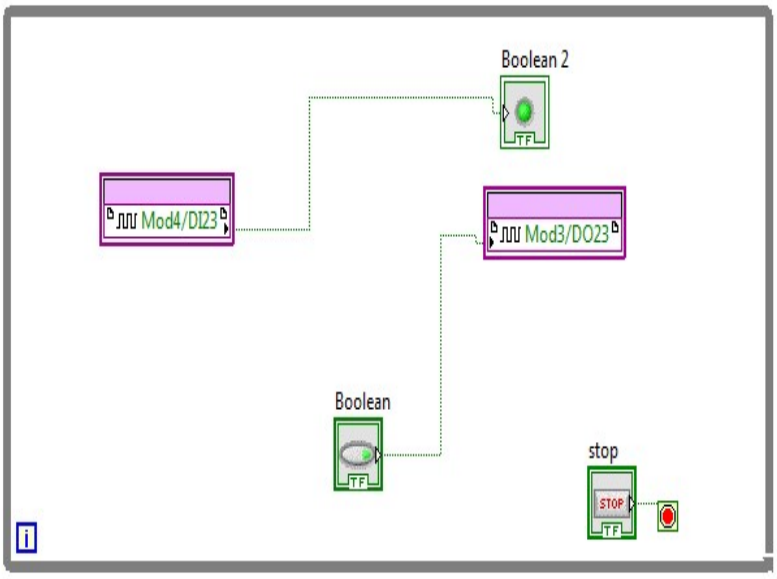


Figure 5.24: Code to read I/O from Crio

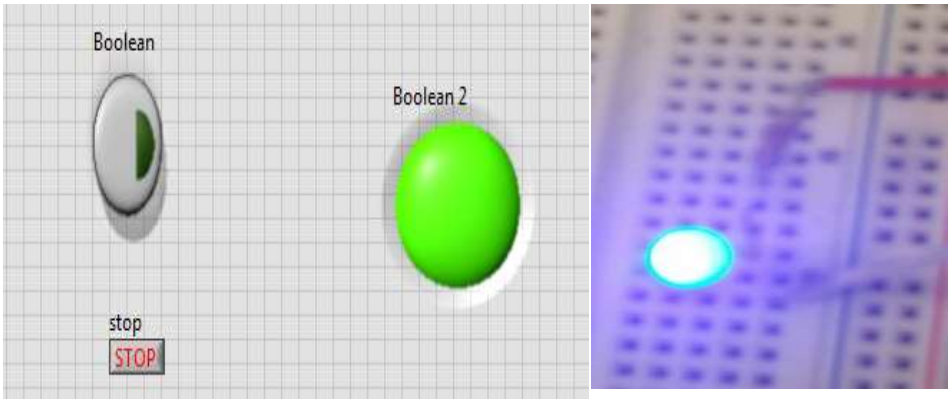


Figure 5.25: Reading cRIO I/O

5.5 Reading encoder signals with LabVIEW

LabVIEW code is written to read the encoder signal from NI ELVIS II. The NI Elvis is a training kit for students to learn. Elvis II has multiple functionalities; it can read digital and analogue I/O. Generate a constant 5VDC and 15VDC. It has an onboard DMM, function generator and so on.

A code to measure the encoder signals with LabVIEW is shown in figure 5.26. There are two functions i.e. waveform graph and waveform chart. The functionality of both the blocks is same but the X- axis scale is set to different values. The output of waveform graph is shown in figure 5.26 and output for waveform chart is in figure 5.27

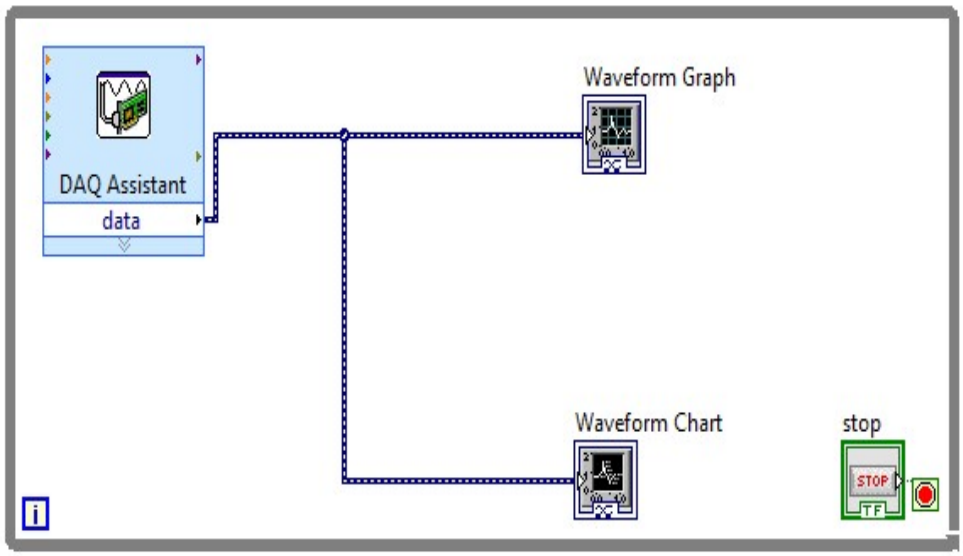


Figure 5.26: Reading encoder signals.

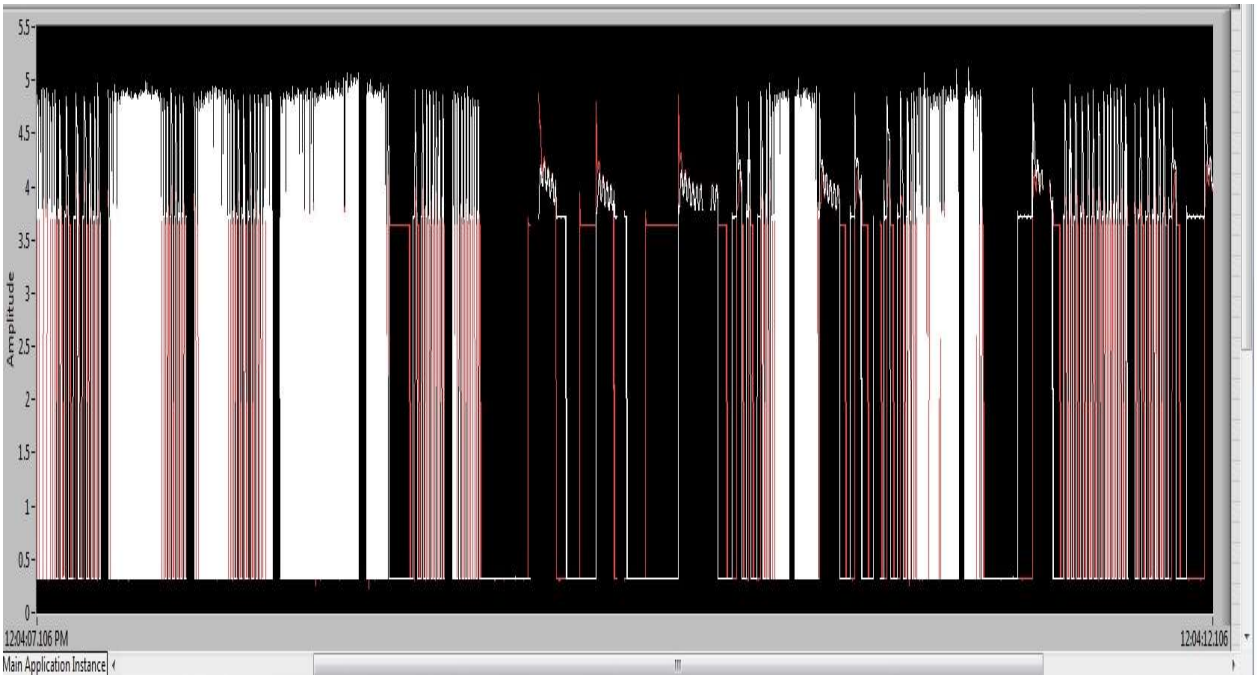


Figure 5.27: Encoder Output for A and B

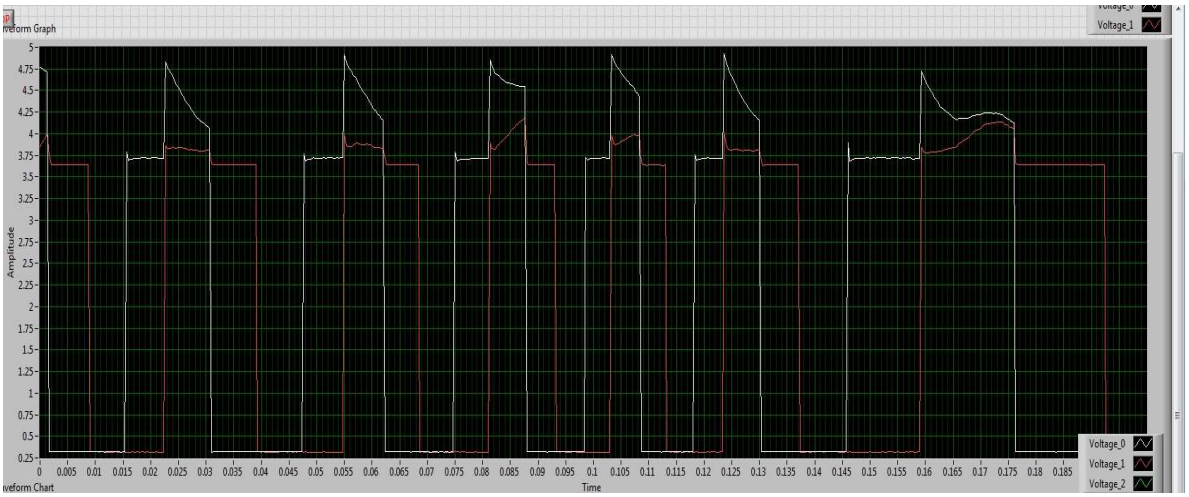


Figure 5.28: Encoder Output for A and B

Code to read encoder pulses is shown in figure 5.29, the code reads the encoder pulses, identifies the direction of motion of the motor and the counts the positon on the encoder. The encoder gives 1000 pulses per one revolution we can count the number of pulses and by using the gear ratio related motor and the joint we can precisely measure the position of the joint. As the encoder is an incremental type, it doesn't remember the previous position. There is a position reset button available in the code. By resetting the position we can set any position of joint to zero.

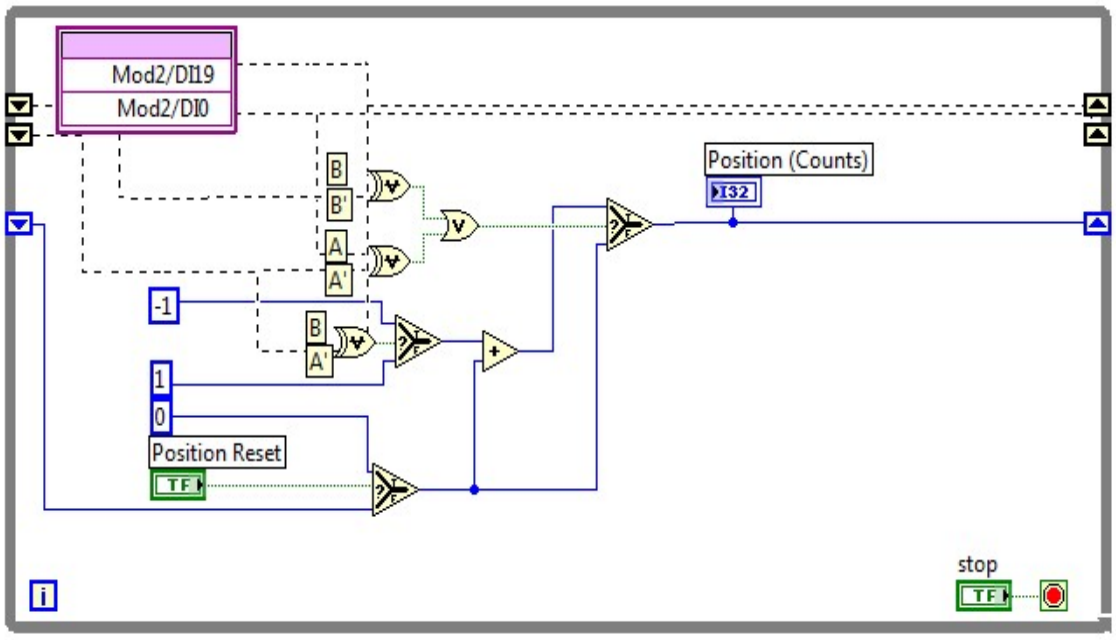


Figure 5.29: Counting Encoder Output at A and B pulses

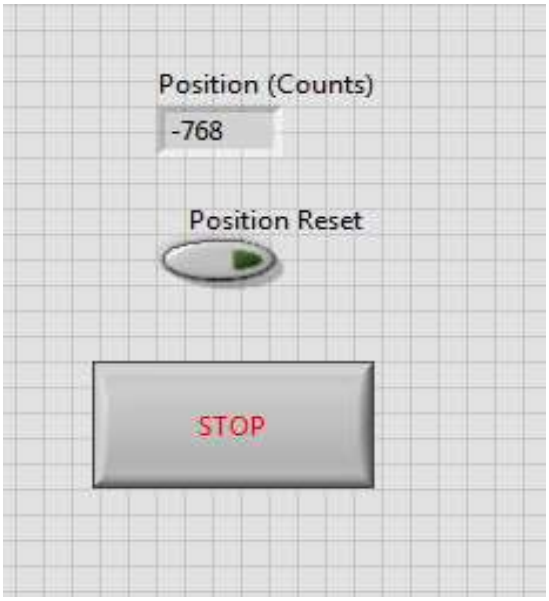


Figure 5.30: Counting Encoder position

5.6 Controlling robot with NI CRIO

The bot requires 6 to 36VDc and a current from 0 to 2AMS, how cRIO is not capable of handling this range of currents. So we have to use a L298N as a drive, for higher currents. A control system design is to be made to adopt an approach for controlling the CRS PLUS robotic arm. To control a robotic arm, that is to move it to one point to other in the work space, a smooth motion and a precise motion is needed. Movement of the arm is combined effect of movement of each axis. Controlling each of the motor will have a cumulative effect on the arm. To make a smoot trajectory, the initial and final velocity needs to be zero. Also we need to have initial and final position. So, we have

$\Theta(0) = \text{initial position}$ …….Eq 5.3

$\Theta(t_f)$=final position (t_f is final time) …….Eq 5.4

Differentiation of the above initial and final position, we have initial and final velocities.

$\Theta'(0) = 0$ …….Eq 5.5

$\Theta'(t_f) = 0$ …….Eq 5.6

The cubic polynomial [15] is taken with four coefficients that satisfy equations from 5.3 to 5.4.

$$\Theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \text{……} \quad \text{Eq 5.7}$$

So the joint velocity is differentiation of the equation 5.7

$$\Theta'(t) = a_1 + 2a_2 t + 3a_3 t^2 \text{..... Eq 5.8}$$

So the joint acceleration is differentiation of the equation 5.8

$$\Theta''(t) = 2a_2 + 6a_3 t \text{..... Eq 5.9}$$

Using Equation 5.7, 5.8 and 5.9 one can control position, velocity or acceleration of one joint.

For instance let us drive the motor to position +100, in a time frame of 10 Sec. To calculate the equation from velocity, substitute t and position in equation 5.7 and 5.8

$$\Theta(0) = a_0 + a_1(0) + a_2(0)^2 + a_3(0)^3 = 0$$

$$\Rightarrow a_0 = 0$$

$$\Theta(10) = a_1(10) + a_2(10)^2 + a_3(10)^3 = 100$$

$$\Rightarrow a_1 + a_2 10 + a_3 100 = 10$$

$$\Theta'(0) = a_1 + 2a_2(0) + 3a_3(0) = 0$$

$$\Rightarrow a_1 = 0$$

$$\Theta'(10) = 20a_2 + 3a_3(10)^2 = 0$$

$$\Rightarrow 2a_2 + 30a_3 = 0$$

After solving we have

$$a_0 = 0;$$

$$a_1 = 0;$$

$$a_2 = 3;$$

$$a_3 = \frac{-1}{5}$$

So the equation for velocity control will be

$$\Theta'(t) = 6t - \frac{3}{5}t^2$$

LabVIEW code is written for this velocity equation.

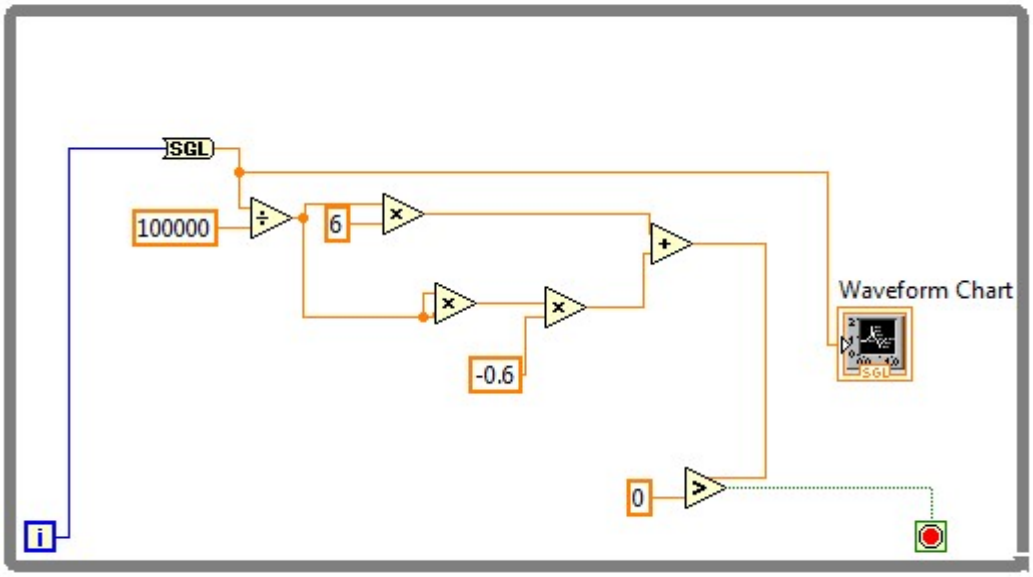


Figure 5.31: Velocity control in LabVIEW code.

The output of this code can be given to control the motor through PWM. LabVIEW code for PWM was not successful so, Arduino is chosen as alternative approach for controlling the CRS plus A150 robotic Arm was chosen. Arduino has a capability to interface with LabVIEW and hence the idea of controlling the robot with ROSS can be achieved.

5.7 Reading encoder signals Arduino

Arduino coding also called sketch is very similar to C programming language. As it is an open source language there is lot that can be done with Arduino.

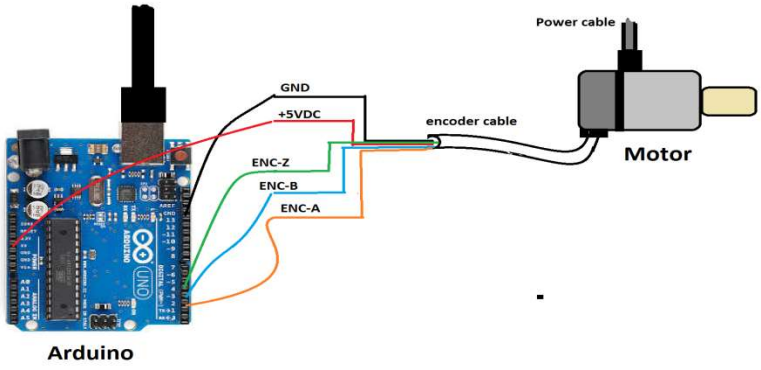


Figure 5.32 shows the circuit connections to read encoder signals

Coming to reading encoder, the most important parameters are reading pulses from A, B and identify the direction of the motor and finally increment or decrement position depending on pulses and direction of rotation. Figure 5.32 shows the circuit connections needed. Here we need not do any additional signal conditioning as the Arduino is capable of reading 5V pulses from A and B pins of encoder.

A simple code for to read encoder and output is given in Appendix.

5.8 Controlling robot with NI Arduino

Once the encoder readings were precisely measured, the next step is to control the robot. For this the Arduino cannot produce the voltage and current required for the motor. So L298n is to be used. We can generate required PWM signal and control the robot motion very precisely. The various codes experimented and the results are given in Appendix.

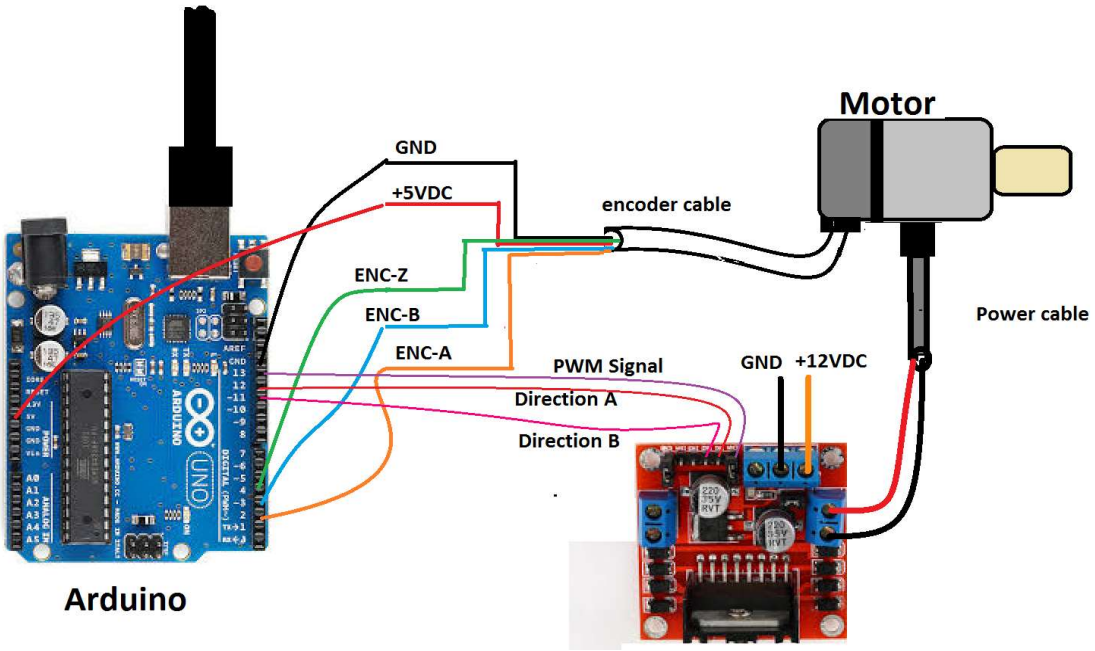


Figure 5.33 connections for controlling motor with Arduino

Figure 5.33, shows the connections between motor, Arduino and L298N that required to control a motor using Arduino. Note that common ground is to be maintained.

6 Conclusion and future work

6.1 Conclusion

In this thesis, CRS A150 Robot Arm is controlled using Arduino. Additionally, to increase the efficiency two Arduinos are connected in I2C (serial communication) and CRS A150 Robot Arm is controlled in master slave combination. Arduino processor is comparably slower in process than NI cRIO but is capable enough to command the robot. The robot is tested with both position control and velocity control successfully.

6.2 Future work

The following are to be achieved to improve and use this CRS A150 robot arm.

1. Serial communication can be extended to six Arduinos to control six axis simultaneously.

2. LabVIEW and Arduino can be interfaced for better control of the robot.

3. ROS and Arduino can be interfaced for more robust and efficient control of CRS A150 Robot ARM

Appendix

//Arduino program to read an input from key board

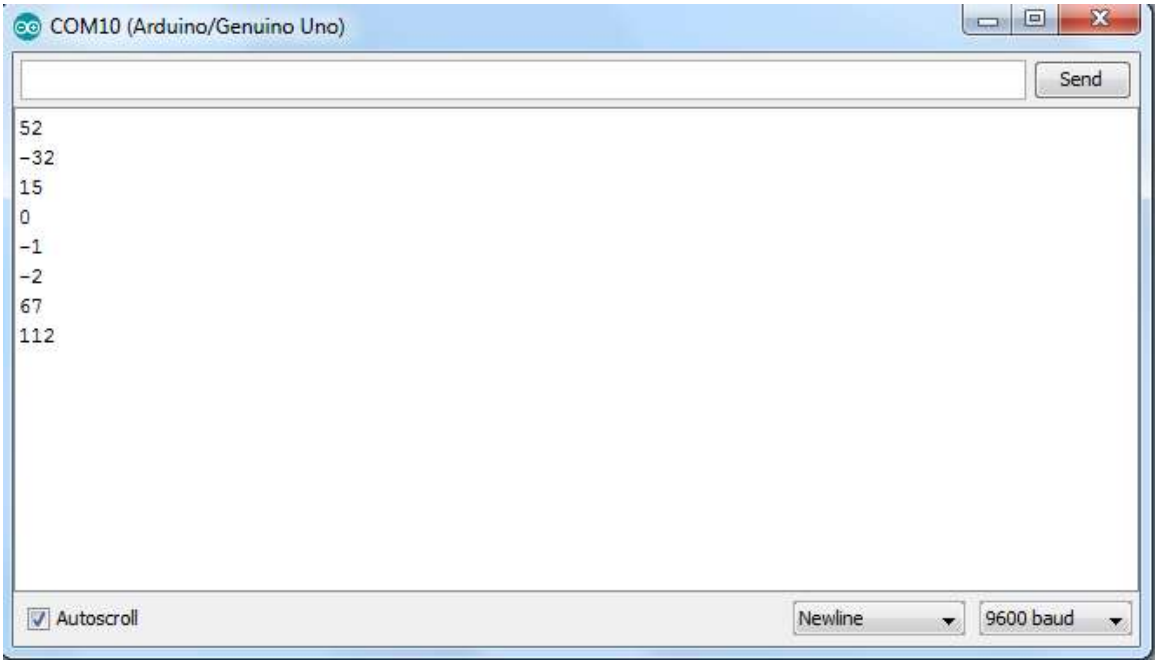
```
long i=0;// input variable

void setup()
{
  Serial.begin(9600);
  Serial.setTimeout(10);
}
void loop()
{
  while (Serial.available()==0){}
  i= Serial.parseInt();

Serial.println(i);

}
```

Output



//Arduino program to read an encoder signal

```
int val;
 int encoder0PinA = 10;
 int encoder0PinB = 11;
 int encoder0Pos = 0;
 int encoder0PinALast = LOW;
 int n = LOW;

 void setup() {
   pinMode (encoder0PinA,INPUT);
   pinMode (encoder0PinB,INPUT);
   Serial.begin (9600);
 }

 void loop() {
   n = digitalRead(encoder0PinA);
   if ((encoder0PinALast == LOW) && (n == HIGH)) {
     if (digitalRead(encoder0PinB) == LOW) {
       encoder0Pos--;
     } else {
       encoder0Pos++;
     }
     Serial.print (encoder0Pos);
     Serial.print ("\n");
   }
   encoder0PinALast = n;
 }
```

Output:


COM10 (Arduino/Genuino Uno)
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
-14
-15
-16
-17
-18
-19
-20
-21
-22
-23
-24
-25
-26

```
//Arduino program to run a motor in open loop control

#define InA1          11              // INA motor pin
#define InB1          12              // INB motor pin
#define PWM1           13               // PWM motor pin


void setup() {
 pinMode(InA1, OUTPUT);
 pinMode(InB1, OUTPUT);
 pinMode(PWM1, OUTPUT);
 }

void loop() {
 motorForward(250);                    //(25%=64; 50%=127; 100%=255)
 delay(2000);

 motorStop();
 delay(5000);

 motorBackward(250);
 delay(2000);

 motorStop();
 delay(5000);

}

void motorForward(int PWM_val)
{
 analogWrite(PWM1, PWM_val);
 digitalWrite(InA1, LOW);
 digitalWrite(InB1, HIGH);
 }

void motorBackward(int PWM_val)  {
 analogWrite(PWM1, PWM_val);
 digitalWrite(InA1, HIGH);
 digitalWrite(InB1, LOW);

}
 void motorStop()  {
 analogWrite(PWM1, 0);
 digitalWrite(InA1, LOW);
 digitalWrite(InB1, LOW);
}
```

//Arduino program to run a motor with encoder feedback

```
long i=0;// input variable FROM CODE 1

 int encoder0PinA = 2;// FROM CODE 2
 int encoder0PinB = 3;// FROM CODE 2
 int encoder0Pos = 0;// FROM CODE 2
 int encoder0PinALast = LOW;// FROM CODE 2
 int n = LOW;

#define InA1            11              // INA motor pin
#define InB1            12              // INB motor pin
#define PWM1             13               // PWM motor pin

void setup()
{
  pinMode (encoder0PinA,INPUT);// FROM CODE 2
  pinMode (encoder0PinB,INPUT);// FROM CODE 2

  pinMode(InA1, OUTPUT);// FROM CODE 3
  pinMode(InB1, OUTPUT);// FROM CODE 3
  pinMode(PWM1, OUTPUT);// FROM CODE 3

  Serial.begin(9600);// FROM CODE 1
  Serial.setTimeout(10);// FROM CODE 1
}
void loop()
{
  while (Serial.available()==0){}
  i= Serial.parseInt();
  Serial.println(i);

  n = digitalRead(encoder0PinA);
   if ((encoder0PinALast == LOW) && (n == HIGH)) {
     if (digitalRead(encoder0PinB) == LOW) {
       encoder0Pos--;
     } else {
       encoder0Pos++;
     }
     Serial.print (encoder0Pos);
     Serial.print ("\n");
   }
   encoder0PinALast = n;
```

```
  if(i>=0){
  while (n<=i)
  {
   motorForward(250);
   }
   motorStop();
  }
  else
  {
   while (n>=i)
   {
     motorBackward(250);
     }
     motorStop();
}
```

//Arduino program to control the position for 4th axis

```
long j=0;
long i=0;
 int encoder0PinA = 2;
 int encoder0PinB = 3;
 int encoder0Pos = 0;
 int encoder0PinALast = LOW;
 int n = LOW;
 #define InA1           11
#define InB1           12
#define PWM1       13
 void setup()
{
  pinMode (encoder0PinA,INPUT);
  pinMode (encoder0PinB,INPUT);
   pinMode(InA1, OUTPUT);
  pinMode(InB1, OUTPUT);
  pinMode(PWM1, OUTPUT);
   Serial.begin(9600);
  Serial.setTimeout(10);
}
void loop()
{
  Serial.println("Let's begin!");
  while (Serial.available()==0){}
  j= Serial.parseInt();
  Serial.println(j);
  i=1.6*j;
   while(i>=0){
   n = digitalRead(encoder0PinA);
   if ((encoder0PinALast == LOW) && (n == HIGH)) {
     if (digitalRead(encoder0PinB) == LOW) {
       encoder0Pos--;
     } else {
       encoder0Pos++;
     }
     Serial.print (encoder0Pos);
```

```
    Serial.print ("\n");
  }
    encoder0PinALast = n;
      motorForward(250);
   if (encoder0Pos>=j){
   motorStop();
   }   }
while(i<=0){
  n = digitalRead(encoder0PinA);
  if ((encoder0PinALast == LOW) && (n == HIGH)) {
    if (digitalRead(encoder0PinB) == LOW) {
      encoder0Pos--;
    } else {
      encoder0Pos++;
    }
    Serial.print (encoder0Pos);
    Serial.print ("\n");
  }
    encoder0PinALast = n;

   motorBackward(250);
   if (encoder0Pos<=j){
   motorStop();
   }
  }
}
void motorForward(int PWM_val)
{
 analogWrite(PWM1, PWM_val);
 digitalWrite(InA1, LOW);
 digitalWrite(InB1, HIGH);
 }

void motorBackward(int PWM_val)  {
 analogWrite(PWM1, PWM_val);
 digitalWrite(InA1, HIGH);
 digitalWrite(InB1, LOW);
```

```
}

void motorStop()  {
 analogWrite(PWM1, 0);
 digitalWrite(InA1, LOW);
 digitalWrite(InB1, LOW);
}
```

//Arduino serial communication With LED

//Master code

//i2c Master Code(UNO)

```
#include <Wire.h>
void setup()
{
  Serial.begin(9600);
    Wire.begin();
}
void loop()
{
  while(Serial.available())
  {
    char c = Serial.read();
       if(c == 'H')
    {
      Wire.beginTransmission(5);
      Wire.write('H');
      Wire.endTransmission();
    }
    else if(c == 'L')
    {
      Wire.beginTransmission(5);
      Wire.write('L');
      Wire.endTransmission();
    }
  }
}
```

```
//Slave code

//i2c Slave Code(Leonardo)

#include <Wire.h>
void setup()
{
  Wire.begin(5);
  Wire.onReceive(receiveEvent);
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
}
void loop()
{
}
void receiveEvent(int howMany)
{
  while(Wire.available())
  {
    char c = Wire.read();
    if(c == 'H')
    {
      digitalWrite(13,HIGH);
    }
    else if(c == 'L')
    {
      digitalWrite(13,LOW);
    }
  }
}
```

//Arduino I2c Code for robot control

//Master

```
#include <Wire.h>
int q=0;
long a=0;//for angle
long b,i;
 int encoder0PinA = 2;
 int encoder0PinB = 3;
 int encoder0Pos = 0;
 int encoder0PinALast = LOW;
 int n = LOW;
#define InA1          11
#define InB1          12
#define PWM1       13
void setup()
{
  pinMode (encoder0PinA,INPUT);
  pinMode (encoder0PinB,INPUT);
  pinMode(InA1, OUTPUT);
  pinMode(InB1, OUTPUT);
  pinMode(PWM1, OUTPUT);
   Serial.begin(9600);
Wire.begin();
}
```

```
void loop()
{
  while(q==0){
  Serial.println("first axis");
  while (Serial.available()==q){}
 i= Serial.parseInt();
  Serial.println(i);
Wire.beginTransmission(5);
     Wire.write(i);
     Wire.endTransmission();
q++;
    exit;
    }
Serial.println("axis 2");
 delay (1000);
  while (Serial.available()==0){}
  a= Serial.parseInt();
  Serial.println(a);
  b=1.6*i;
   while(b>=0){
   n = digitalRead(encoder0PinA);
   if ((encoder0PinALast == LOW) && (n == HIGH)) {
    if (digitalRead(encoder0PinB) == LOW) {
      encoder0Pos--;
    } else {
      encoder0Pos++;
    }
    Serial.print (encoder0Pos);
    Serial.print ("\n");
   }
    encoder0PinALast = n;
      motorForward(150);
```

```
   if (encoder0Pos>=a){
   motorStop();
   }
  }
while(b<=0){
  n = digitalRead(encoder0PinA);
  if ((encoder0PinALast == LOW) && (n == HIGH)) {
    if (digitalRead(encoder0PinB) == LOW) {
      encoder0Pos--;
    } else {
      encoder0Pos++;
    }
    Serial.print (encoder0Pos);
    Serial.print ("\n");
  }
    encoder0PinALast = n;
      motorBackward(150);
   if (encoder0Pos<=b){
   motorStop();
   }
  }
}
void motorForward(int PWM_val)
{
 analogWrite(PWM1, PWM_val);
 digitalWrite(InA1, LOW);
 digitalWrite(InB1, HIGH);
 }
 void motorBackward(int PWM_val)  {
 analogWrite(PWM1, PWM_val);
 digitalWrite(InA1, HIGH);
 digitalWrite(InB1, LOW);
  }
 void motorStop()  {
 analogWrite(PWM1, 0);
 digitalWrite(InA1, LOW);
 digitalWrite(InB1, LOW);
}
```

//Slave

```
 #include <Wire.h>
long i=0;
 int encoder0PinA = 2;
 int encoder0PinB = 3;
 int encoder0Pos = 0;
 int encoder0PinALast = LOW;
 int n = LOW;
#define InA1          11
#define InB1          12
#define PWM1       13
void setup()
{
  Wire.begin(5);
  Wire.onReceive(receiveEvent);
   pinMode (encoder0PinA,INPUT);
  pinMode (encoder0PinB,INPUT);
  pinMode(InA1, OUTPUT);
pinMode(InB1, OUTPUT);
  pinMode(PWM1, OUTPUT);
    Serial.begin(9600);
  }
void loop()
{ }
void receiveEvent(int howMany)
{
  while(Wire.available())
  {
   long i = Wire.read();
   while(i>=0){
   n = digitalRead(encoder0PinA);
   if ((encoder0PinALast == LOW) && (n == HIGH)) {
     if (digitalRead(encoder0PinB) == LOW) {
       encoder0Pos--;
     } else {
       encoder0Pos++;
     }
     Serial.print (encoder0Pos);
```

```
    Serial.print ("\n");
   }
   encoder0PinALast = n;
   motorForward(250);
   digitalWrite(8, HIGH); //for led not important
   if (encoder0Pos>=i){
    motorStop();
   }
}
 while(i<=0){
   n = digitalRead(encoder0PinA);
   if ((encoder0PinALast == LOW) && (n == HIGH)) {
     if (digitalRead(encoder0PinB) == LOW) {
       encoder0Pos--;
     } else {
       encoder0Pos++;
     }
     Serial.print (encoder0Pos);
     Serial.print ("\n");
   }
     encoder0PinALast = n;

    motorBackward(150);
    if (encoder0Pos<=i){
    motorStop();
    }
   }
   }
}
void motorForward(int PWM_val)
{
 analogWrite(PWM1, PWM_val);
 digitalWrite(InA1, LOW);
 digitalWrite(InB1, HIGH);
 }
 void motorBackward(int PWM_val)  {
 analogWrite(PWM1, PWM_val);
 digitalWrite(InA1, HIGH);
```

```
 digitalWrite(InB1, LOW);
  }
 void motorStop()  {
 analogWrite(PWM1, 0);
 digitalWrite(InA1, LOW);
 digitalWrite(InB1, LOW);
}
```

REFFERENCE

[1] http://www.digital-circuitry.com/MyLAB_Robotics_CRS-M1.htm

[2] http://www.torquesystems.com

[3]. Muhammad Mubeen, "Brushless DC Motor Primer," Motion Tech Trends, July, 2008.

[4]. Derek Liu, "Brushless DC Motors Made Easy," Freescale, 2008.

[5]. Padmaraja Yedamale, "Hands-on Workshop: Motor Control Part 4 -Brushless DC (BLDC) Motor Fundamentals," Microchip AN885, 2003.

[6] http://nptel.ac.in/courses/112103174/module2/lec3/3.html

[7] http://specimens.iri.isu.edu/etd/GetFile.aspx?exid=97&etid=1&mtid=1

[8] http://sine.ni.com/psp/app/doc/p/id/psp-822/lang/en

[9] http://sine.ni.com/nips/cds/view/p/lang/en/nid/208824

[10] http://sine.ni.com/nips/cds/view/p/lang/en/nid/208815

[11] http://www.ni.com/en-us/support.html

[12] http://www.nxp.com/documents/user_manual/UM10360.pdf

[13] https://developer.mbed.org/platforms/mbed-LPC1768/

[14] https://www.arduino.cc/en/Main/arduinoBoardMega

[15] http://dallaskasaboski.blogspot.com/2012/01/fencing-aikido-robots-chuck-and-friends.html

[16] http://dallaskasaboski.blogspot.com/2012/01/fencing-aikido-robots-chuck-and-friends.html

[17] Arseneau. S.c., R.J. Nicholls, M.Farooq, and A.Hopkinson. "Robotic mimicking control system." *Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuit and Systems. MWSCAS 2001.*