

ROBOTIC ARM: A LEARNING PLATFORM

Ishrak Hasnain

Bachelor of Engineering (Honours)

With a major in Mechatronics Engineering



Department of Engineering

Macquarie University

Date: 05th September 2016

Supervisor: Professor Subhas Mukhopadhyay



ACKNOWLEDGMENTS

I would firstly like to acknowledge my family who has supported me throughout the entire process of my undergraduate degree. With the motivation and support received from my academic supervisor Professor Subhas Mukhopodhyay, I was able to learn new things, and challenge my abilities to move forward and develop myself. In this process I have also met and worked with a number of people who had helped and guided me as well. I take this opportunity to thank each and every person who helped me accomplish this and believed in me. Thank you.



STATEMENT OF CANDIDATE

I, Ishrak Hasnain, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the Department of Mechatronic Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment at any other academic institution.

Student's Name: Ishrak Hasnain

Student's Signature: Ishrak Hasnain (electronic)

Date: 05 April, 2016



ABSTRACT

A good learning platform is an essential part of a student's learning outcome. In this document, study and research conducted on a Crust Crawler AX-12A Smart Robotic Arm is stated. This includes the details of the hardware used, the assembly procedure, software used and Experimental procedure. The project was conducted to develop a learning platform with these laboratory instructions to help students learn how to control a robotic arm. The key component used in this project is the AX-12A Dynamixel motor which is a smart servo motor with its own microcontroller and memory. Nine Laboratory experiments were developed based on LabVIEW which would allow the students to learn from basic controls to performing a repetitive or conditional operation. Limitations and issues faced during the process is also shared.

Contents

Acknowledgements

Abstract

Table of Contents

List of Figures

List of Tables

CHAPTER 1	INTRODUCTION.....	11
1.1	PROJECT GOAL	11
CHAPTER 2	BACKGROUND	12
2.1	SERVO MOTOR.....	12
2.2	COMPONENTS OF ROBOTIC ARM	12
2.3	SERIAL AND PARALLEL COMMUNICATION.....	13
CHAPTER 3	SAFETY.....	14
CHAPTER 4	HARDWARE	15
4.1	DYNAMIXEL SERVO AX-12A	15
4.2	USB2DYNAMIXEL.....	15
4.3	POWER SUPPLY	17
4.4	CRUST CRAWLER AX-12A SMART ROBOTIC ARM FRAME.....	17
CHAPTER 5	CRUST CRAWLER ROBOTIC ARM ASSEMBLY	18
CHAPTER 6	SOFTWARE.....	35
6.1	ROBOTIS ROBOPLUS 1.0.....	35
6.2	LABVIEW	37
6.2.1	<i>VI</i>	37
6.2.2	<i>Front Panel</i>	37
6.2.3	<i>Block Diagram</i>	37
6.2.4	<i>Control Palette</i>	37
6.2.5	<i>Function Palette</i>	38
6.2.6	<i>Toolbar</i>	38
6.2.7	<i>Help</i>	39
CHAPTER 7	LAB EXPERIMENT PROCEDURE	39
7.1	GETTING STARTED WITH DYNAMIXEL	39
7.2	STABLISHING CONNECTION VIA THE USB COM PORTS	45
7.3	CONTROLLING A DYNAMIXEL SERVO MOTOR	48

7.4	USING INPUT AND OUTPUT INSTRUMENTS	52
7.5	CONTROLLING THE ROBOTIC ARM.....	56
7.6	CONTROLLING GRIPPER USING TORQUE.....	58
7.7	POSITION FEEDBACK.....	61
7.8	USING DATA FROM A FILE	62
7.9	PERFORMING REPETTIVE OR CONDITIONAL OPERATION.....	64
CHAPTER 8	DISCUSSION	66
CHAPTER 9	CONCLUSION	68
REFERENCES		69

Chapter 1 Introduction

Robotic arm is being widely used in the manufacturing industry and is gradually taking over intensive and repetitive activities which previously used to be conducted by human beings. This is helping reduce the health impact suffered by a person due to these activities.

1.1 Project Goal

The goal of this project is to study and experiment on a Crust Crawler AX-12A/AX-18A Smart Robotic Arm which has recently been purchased by the university. The components included in the robotic arm are: AX-12A Dynamixel Servos, USB2Dynamixel connector and power supply. LabVIEW will be used as the software to operate the robotic arm . During the process, the complications and the limitations of the components should be investigated and presented accordingly. Finally using the investigations and development of this project, learning resources such as safety manual, installation procedure, and laboratory procedures need to be produced for the undergraduate students studying the unit ELEC326 Mechatronics Systems.

Chapter 2 Background

2.1 Servo Motor

Servo motors consists of a motor which is integrated with its own position sensors, control circuit. The control circuit of the servo read the current position of the motor and rotates or holds the desired position based on the input received, and works in a closed loop. The advantage of a servo motor is that its capable of holding its torque at high rpms which means that it can perform heavy works faster. The motor controller only provides the required amount of current for the servo to hold its position or to move to its location. If a heavier load in applied to the motor, it increases the current to the motor and performs its task. Due to the complexity of the design of the servos, they have a higher cost.

2.2 Components of Robotic arm

Robotic arms are very complex and technical to construct and understand. Despite that, its components can be broken down to small and understandable chunks.

Typically, a robotic arm requires mainly five components [1]. These five components are:

Controller

The controller is designated to connect to all the sensors and Servo motors in the system. It sends and receives data and power to and from the components to enable them to perform the operation.

Manipulator

The manipulator is referred to as the arm of the robot. This is the links which would be moved with the help of the servo motors in the system to change the position of the robotic arm.

End effector

Also, known as the robot's hand, the end effector is a device which is attached to the wrist of the manipulator. This enables it to perform tasks such as lifting, grasping, manoeuvring, transporting, along many other operations.

Power supply

The power supply, would supply the required amount of energy to drive the servo motors and controllers of the robotic arm.

A means for programming

This is used for the purpose of training or instructing the components of the robotic arm to perform the required task. This can be achieved by writing programs that would compute the inputs received from the sensors and the controllers and drive the servo motors accordingly to achieve the required output.

2.3 Serial and Parallel Communication

The serial and parallel are the two types of communication modes that establishes communication link between electronic devices. The parallel communication link consists of more than one channel and therefore can transmit and receive data at the same time. This also provides the ability to view data that are transmitted in real time unlike serial communication links. In case of the serial communication link, data is transmitted and received using a single channel, however, to achieve a higher data rate, the transmission can be clocked up.

Chapter 3 Safety

- Bolt the mounting tabs of the robotic arm down to a secured and stable base before starting any operation.
- Secure all components of the robotic arm and ensure that there are no loose screws and parts before starting operation.
- Inspect all the wires and wiring connections. Zip tie any loose or hanging wire to avoid damaging the wires. Damaged wires can short the metal body and electrocute.
- Ensure to always maintain a safe distance of at least a meter from the robotic arm during operation and also while the power is on [2].
- Turn off power immediately in the following cases:
 - Mechanical failure
 - Electrical failure
 - Electrocution
 - Malfunction (Software or Hardware)
 - Unusual Vibration or Noise
 - Burning Smell, Smoke or Fire
- Tie up long hairs and remove all loose articles before handling the robotic arm. In case of a tangle or parts of body being stuck, turn off power immediately to release the torque of the servos.
- In case of a servo overheating, thermal shutdown will be initiated by the controller of the servo to protect the motor. Do not immediately reset the motor. Shut down the power and allow the motor to cool down for at least a few minutes to avoid permanent damage to the servo motor [2].

Chapter 4 Hardware

4.1 Dynamixel Servo AX-12A

Dynamixel servos model number AX-12A are smart servos developed by ROBOTIS which are capable of working in a network. The network can be created by connecting up to 254 servos consecutively in a series and provide them with unique IDs between 0 to 253. The data communication speed between the servos and the controller can range between 7343bps to 1Mbps depending on the data required to be transmitted and also the device used for communication. Some smart features of Dynamixel servos are that they can provide feedbacks such as position, temperature, load, and input voltage. Dynamixel also consist safety features such as Overload Error, Range Error, Overheating Error, Angle Limit Error, Input Voltage Error, etc. [3]. When such errors are detected, Dynamixel servos can automatically cut off power, and hold the limits, to protect itself [4]. The servos have a stall torque of 1.5N.m at 12V and 1.5A, and the no-load speed of the motor is 59rpm at 12V.

4.2 USB2DYNAMIXEL

The USB2Dynamixel is device which is used to establish a communication between the computer and the Dynamixel servos via the USB port of the computer. The device is capable of establishing connection via the follow connectors: 3 Pin (3P), 4 Pin (4P), and serial port connector. Any disturbance in voltage produce an unstable connection between the equipment and the computer, therefore a proper grounding of all components are required [5]. The parts of the USB2Dynamixel are shown in the figure 1 below.

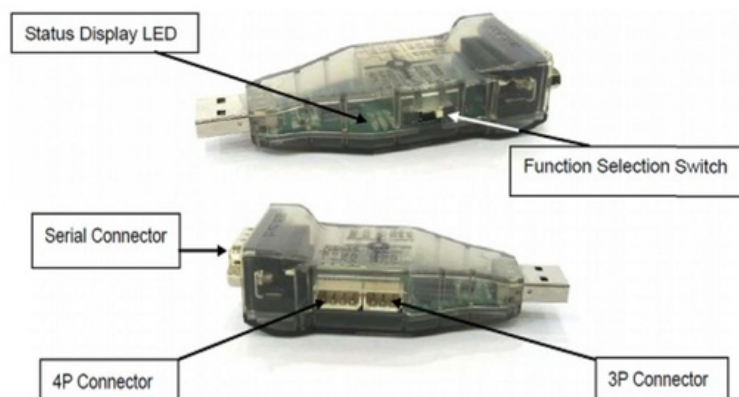


Figure 1

The function selection switch as seen in figure 1 is used to select between the three communication modes available in this device [5].

1. Selecting TTL, establishes communication between the USB port and the 3-pin(3p) port. This port is usually used to connect AX series Dynamixel servos.
2. Selecting RS484, establishes communication between the USB port and the 4-pin(4p) port. This port is usually used to connect DX, RX, and EX series Dynamixel servos.
3. Selecting RS232, establishes communication between the USB port and controllers using serial cable. This port is usually used to connect to controllers such as CM-5, CM-510 or wireless devices such as ZigBee to generate a wireless communication.

In the following figures 2 and figure 3 the functions of the connecting pins of a USB2Dynamixel can be seen.

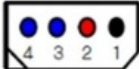

4 Pin Cable			3 Pin Cable		
Pin No.	Signal	Pin Figure	Pin No.	Signal	Pin Figure
1	GND		1	GND	
2	NOT Connected		2	NOT Connected	
3	DATA + (RS-485)		3	DATA (TTL)	
4	DATA - (RS-485)				

Figure 2

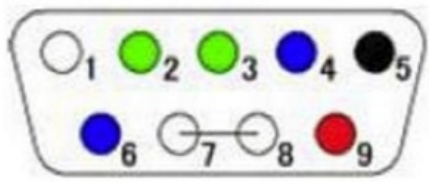
Pin No.	Signal	Pin Figure
1	Data (TTL)	
2	RXD (RS-232)	
3	TXD (RS-232)	
4	D+ (RS-485)	
5	GND	
6	D- (RS-485)	
7	Short with No. 8	
8	Short with No. 7	
9	USB power (5V)	

Figure 3

4.3 Power Supply

The power supply used to operate the robotic arm is a AC/DC Switching adaptor manufactured by Mean Well Enterprise co. Ltd. (Model no. GST60A12). The input and output rating of the adaptor are:

Input: 100-240VAC, 50/60Hz, 1.4A

Output: 12VDC, 5.0A, 60W Max

4.4 Crust Crawler AX-12A Smart Robotic Arm Frame

The frame of the robotic arm has been constructed using aluminium and has a hard-anodized finish which protects the frame from scratches and corruptions. The overall length of the standard configuration arm is around 56.51cm and weight about 1kg without any upgrades [3]. The initial base angle of frame which needs to be fixed in the standard arm configuration can be set in 45° degree intervals. The rotating turntable of the base is supported with 4 Precision carbon steel ball bearings to distribute the weight of the robotic arm and also provide a smooth and unrestrictive rotation of the base. The standard gripper is driven by a 60-tooth heavy duty gear train connected to a single AX-12A servo which can open the gripper wide up to 6cm. The gripper also consists of a mount for sensor which can be adjusted to provide better position for the sensors [3].

Chapter 5 Crust Crawler Robotic Arm Assembly

The assembly process presented below was constructed by referring to the AX12A/AX18A Smart Robotic Arm - Kit Assembly Guide published by the manufacturer Crust Crawler [2]. The process has been simplified and sequences have been changed to ensure a faster and hassle free installation. Difficulties faced during the installation has been considered and the process has been changed accordingly.

1. Start up the Robotis “RoboPlus Manager” software and open the Dynamixel Wizard tab from its home page (this software can be downloaded from <http://support.robotis.com>) [6].
2. Connect one servo at a time to the controller and change/allocate them with unique ID (1-7 for the 7 servos required) and label them accordingly.

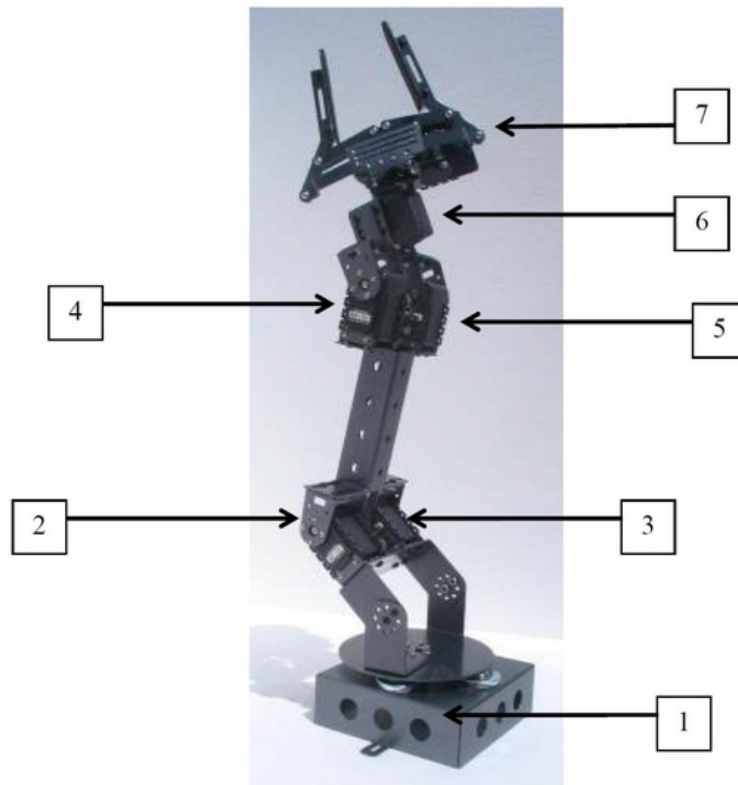


Figure 4

3. During assembly position the servos at the designated positions as shown in the figure 4.
4. Check all the servos to ensure that the Servo horn indicator and the vertical marker on top of the servo is aligned (as shown in figure 5).
5. Note: If misaligned, the servos could result in the operating against each other when connected in parallel [2].

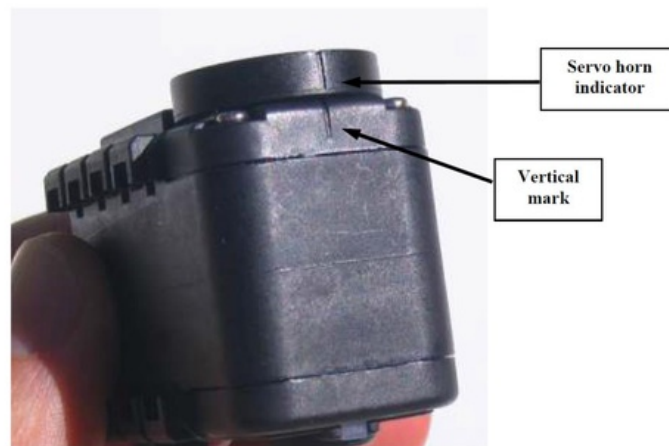


Figure 5

6. Take the servo addressed 1 and place two '#2' nuts from the servo box in the nut slots second from the bottom as shown in figure 7.

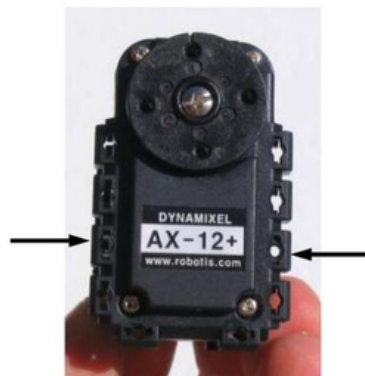


Figure 6

7. Place the servo on the base frame and install two 5mm screws with #2 washers as shown in figure 7.



Figure 7

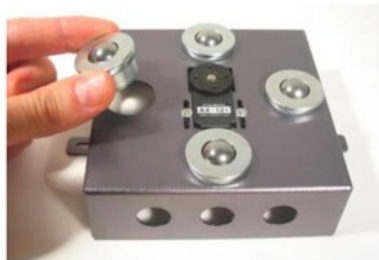


Figure 8

8. In the four 1" circular slot in the base, place 4 carbon steel ball bearings as shown in figure 8.
9. Take four 12mm screw and four washers from the hardware kit and install the lower turntable brace to the turntable as shown in figure 9.

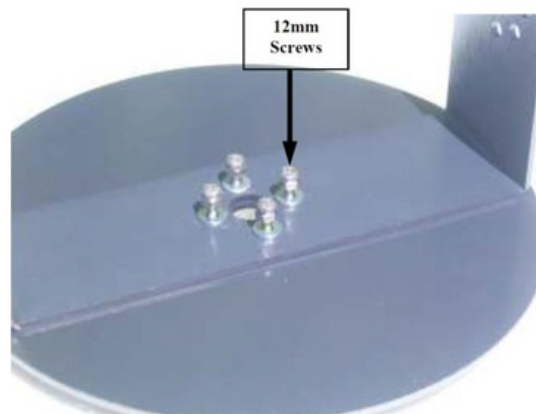


Figure 9

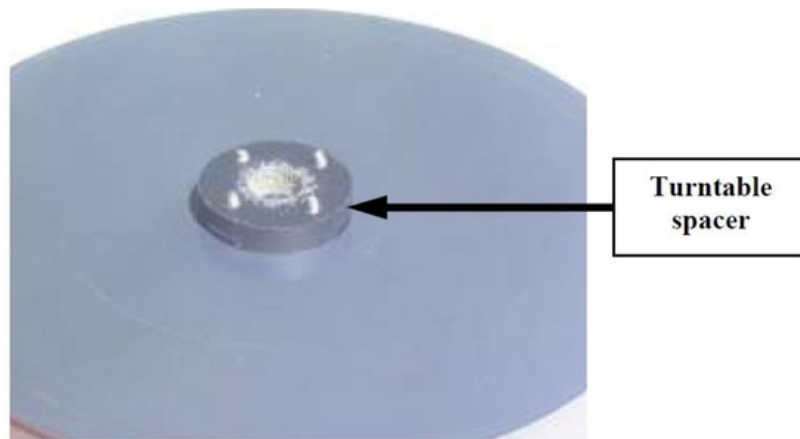


Figure 10

10. Install the turntable spacer under the turn table by aligning the 4 holes to the 12mm screws and keep screwing until they pass through the spacer onto the other side.
11. Install the turntable onto the four screw holes of the servo. Do not Overtighten and align the base as shown in figure 11. After tightening the screws, the turntable should be turned back and forth to verify that it moves freely and is evenly resting on the bearings.

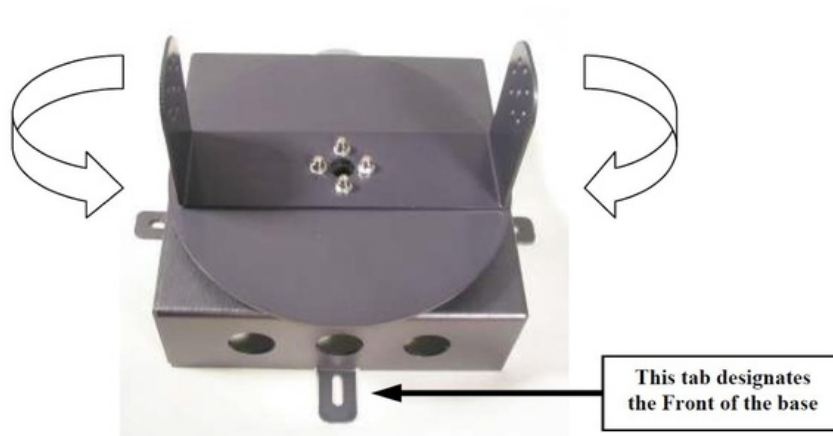


Figure 11

Step 12 - 13 needs to be performed on servos addressed 2,3,4, and 5.

12. Place four #2 nuts on the corner slots of the servos base as shown in figure 12. Install the black Base Servo Bracket that has been provided in the box with four 5mm screws.



Figure 12



Figure 13

13. On the Black Short rotary servo bracket, place the pivot spacer and the pivot as shown in figure 14 and screw them onto the servos using four 5mm screws on the front and the servo set screw at the back as shown in figure 15. Make sure that the bracket is installed pointing upwards and also ensure that the servo horn indicator is correctly aligned with the vertical mark as shown in step 4 of the assembly.

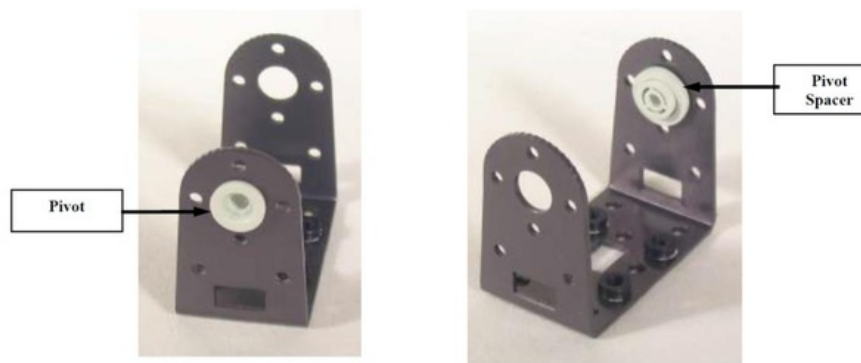


Figure 14

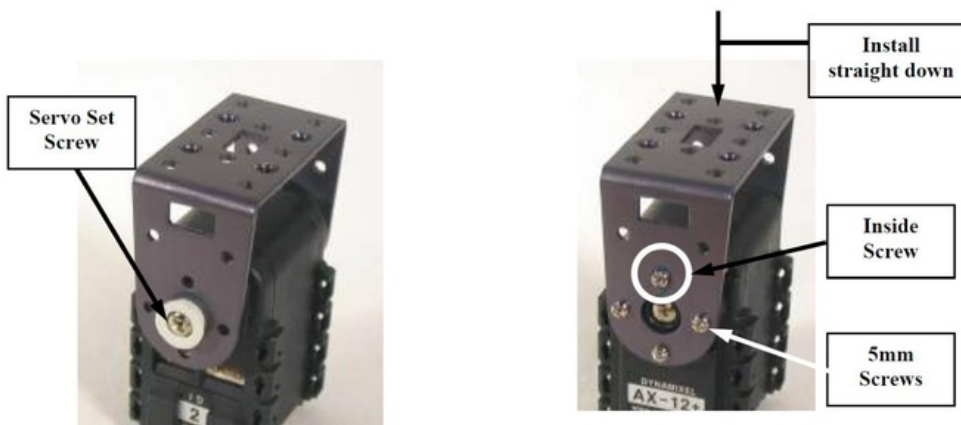


Figure 15

14. The Assembled servos after completing step 11 and 12, can be seen in figure 16.



Figure 16

15. Use four #4, 1/4" screws to attach each servo (addressed 2 and 3) on to the upper brace of the turn table. Ensure that both the servos are facing the same direction.
16. Use eight #4, 1/4" screws to attach the main beam onto the top of the servo addressed 2 and 3. Centre the main beam before tightening the screws.(figure 17)



Figure 17

17. Use eight #4, 1/4" screws to attach the servos addressed 4 and 5 on top of the main beam. Ensure that both the servos are facing the same direction.

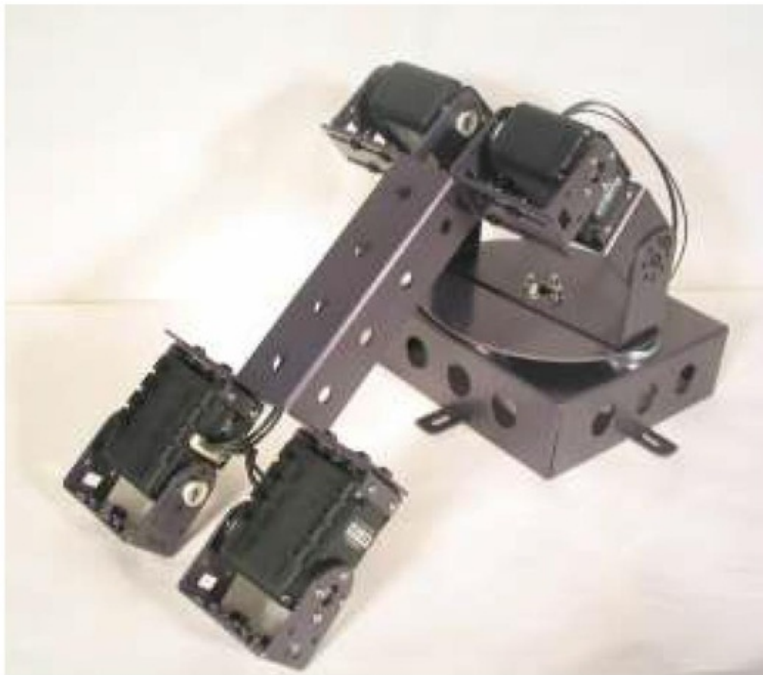


Figure 18

18. Use eight #4, $\frac{1}{4}$ " screws to attach the 90 degree braces on top of the servos addressed 4 and 5. The screws should not yet be tightened.

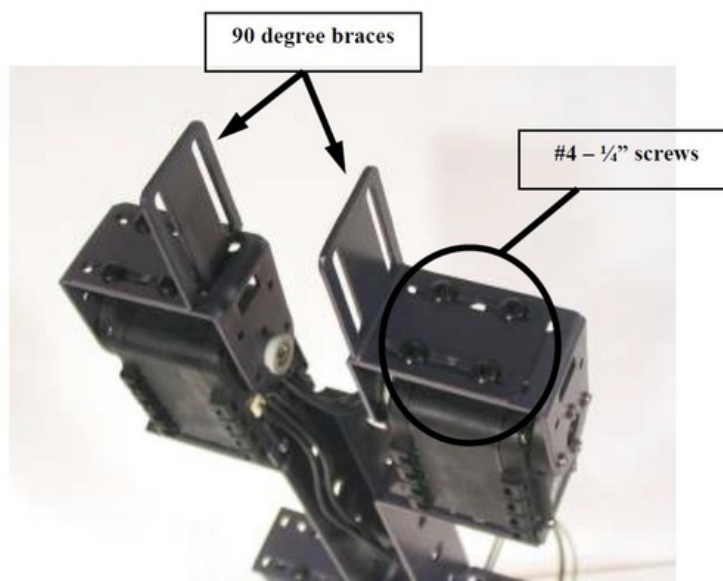


Figure 19

19. On servo addressed 6, four #2 nuts need to be placed on the bottom slots and the 3rd slot from the bottom (figure 20). Repeat the same process on the other side of the servo as well.



Figure 20

20. Attach the servo base bracket on each side of the servo using eight 5mm screws (figure 21).



Figure 21

21. Use eight #4, 1/4" screws to attach the servo addressed 6 onto the 90 degree brackets. Tighten the screws on the side of the servo first and then align the servo to the centre (figure 22). Tighten the screws on top of the servo addressed 4 and 5 (figure 23).

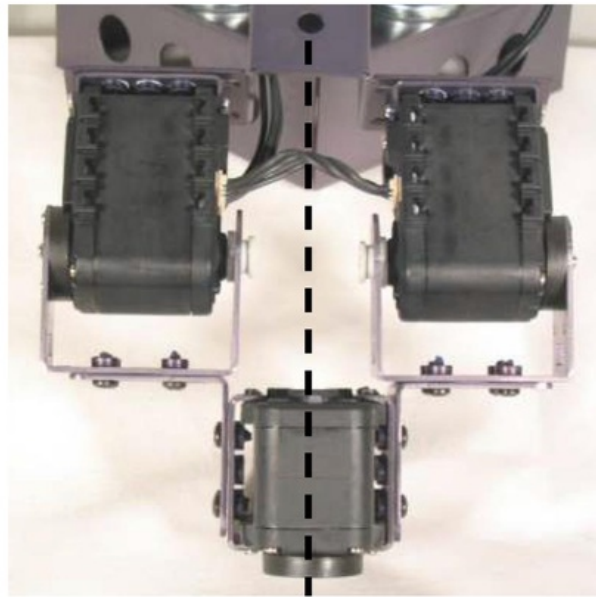


Figure 22

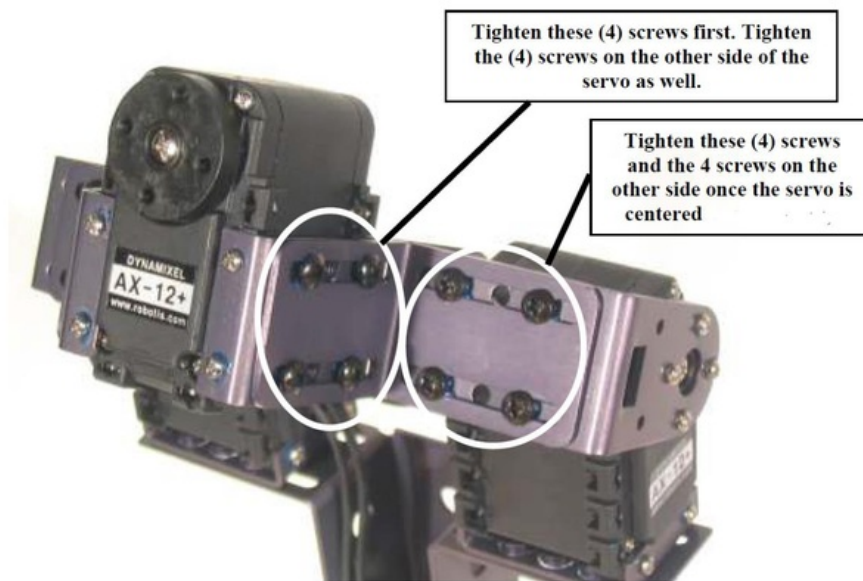


Figure 23

22. Use four 5mm screws to attach the bracket for the gripper onto the servo addressed 6 as seen in figure 24.

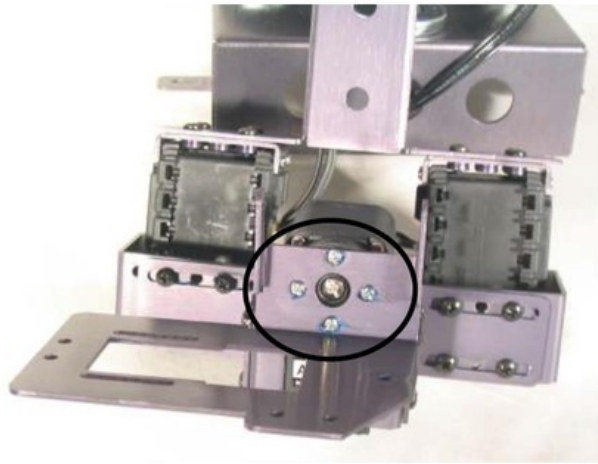


Figure 24

23. Use two 5mm screws and two washers to attach the servo addressed 7 onto the bracket of the gripper.



Figure 25

24. On the **odd spaced** holes of the plastic gear, screw in two #2, 5/16" screws as shown on figure 26. The 5/16" screw can be found on the bag of screws labelled as Hardware set.

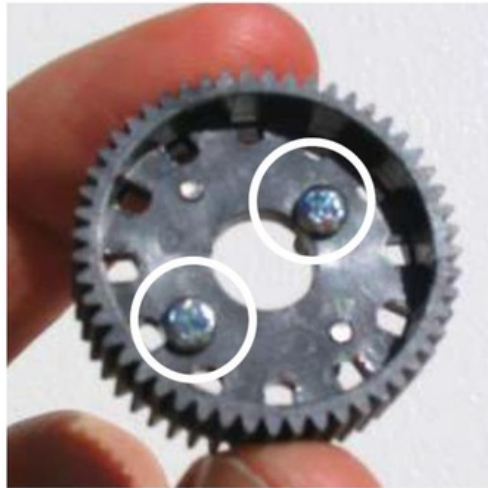


Figure 26

25. Screw in two 10mm screws through the opposite direction of the even holes as shown in figure 27.

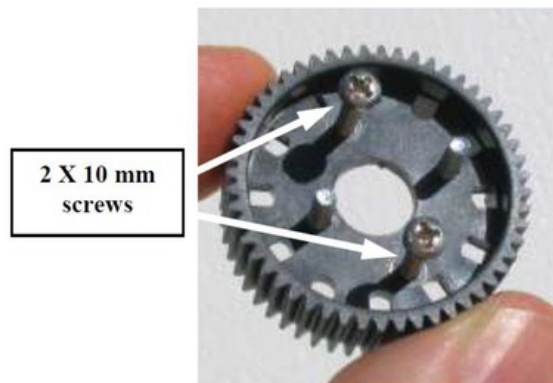


Figure 27

26. Install the black spacer by aligning the holes with the odd spaced screw heads as shown in figure 28 and 29.



Figure 44

Large holes

Figure 28



Figure 29

27. Assemble the gear onto the motor of the gripper as shown in figure 30. Do not overtighten.

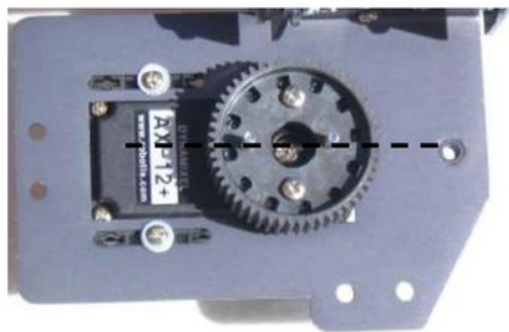


Figure 30

28. Use a #4 5/8" screw to install the pre-set gear as shown in figure 31. Use a #4 – 1/8" nylon spacer between the frame and the gear. Ensure the proper alignment of the two gears.



Figure 31

29. Install the gripper brace onto the gears using two #2 lock washers and nuts as shown in figure 32.



Figure 32

30. Use the components listed in figure 33 in the same sequence and install it on the holes shown in figure 34.

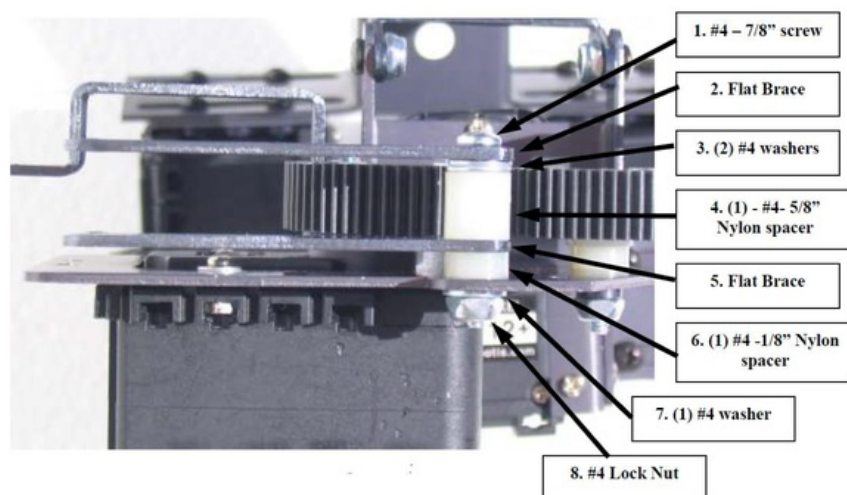


Figure 33

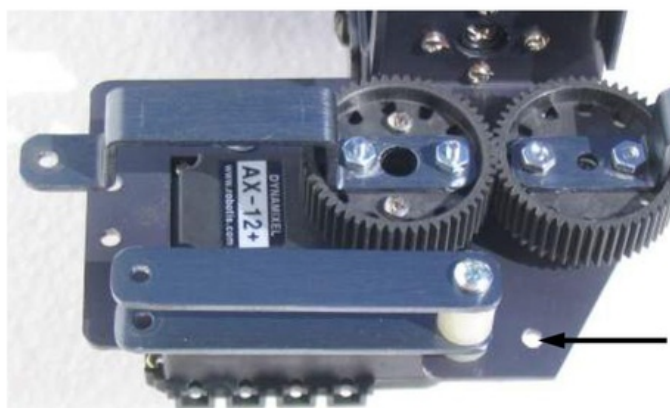


Figure 34

31. Screw the gripper paddles on together using four #4-1/4" screws and paste them onto the rubber gripper pad as shown in figure 35.



Figure 35

32. Install the gripper assembly onto the flat braces using the components listed in figure 36.

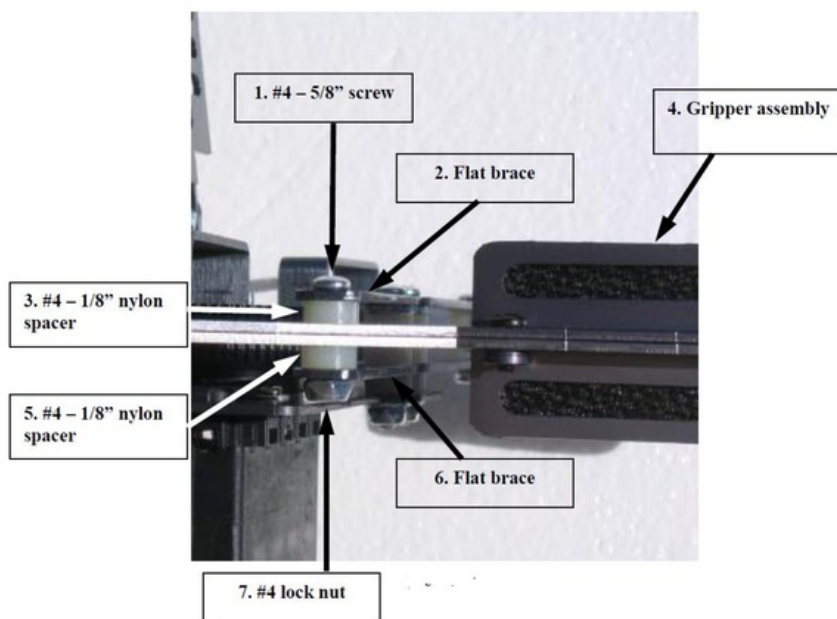


Figure 36

33. The final view of the assembled gripper is shown on figure 37.



Figure 37

Chapter 6 Software

6.1 Robotis Roboplus 1.0

Roboplus 1.0 is a software developed by ROBOTIS which can be used to connect, configure, and control ROBOTIS components such as controllers, wireless and communication devices, Dynamixel motors, sensors etc. The roboplus software consist of multiple programs such as the Roboplus Manager, Roboplus Motion, and Dynamixel Wizard. Roboplus Manager is used to configure the Robotis controllers and communication devices such as ZigBee. The Roboplus Motion function can be used to model the movement of the robots manufactured by Robotis. It consists of preloaded 3D models of the robots and provides an animated view of the robot's motion. Dynamixel wizard is the easiest way to connect to the Dynamixel motors and can be used to update firmware, configure, and calibrate [7]. The program from Roboplus 1.0 used for this project is the Dynamixel wizard.

Dynamixel wizard is initially used to set the motor ID's of each individual motor one by one. This needs to be done and the motors need to be labelled before the assembly of the robotic arm is started. After the assembly, the motors can be wired up in series and then connected and searched with Dynamixel wizard, to view all the motors connected in the system. Once connected using Dynamixel wizard, the EEPROM (electrically erasable programmable read-only memory) of each motor can be accessed and modified. The EEPROM consists of all the configurable values of the motor such as: Motor ID, Baud rate, Angle Limits, Temperature Limits, Voltage Limits, Torque Limits, etc. These values are stored in a location of the memory with addresses between 1 to 18 of the EEPROM and the stays set even if the power is turned off. The space in the memory between addresses 24 to 49, is used by the RAM and reset to an initial value every time power is turned off. The RAM hold values of the all the current and operating status of the motor. Some values available in the RAM are: Goal Position, Goal Speed, Present Position, Present Speed, Present Load/ Torque, Present voltage, Present temperature, etc. Using Dynamixel wizard, all the parameters of all the motors can be viewed, and set at the same time, which makes it very convenient and allows the system to operate within safe limits [4]. A detailed list of the addresses of all the configurable and controllable parameters in the EEPROM and the RAM can be found on the control table in figure 38.

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
E E P R O M	0 (0X00)	Model Number(L)	Lowest byte of model number	R	12 (0X0C)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	0 (0X00)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	1 (0X01)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	3 (0X03)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	70 (0X46)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	140 (0X8E)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36(0x24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36(0x24)
R A M	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	CW Compliance Margin	CW Compliance margin	RW	1 (0X01)
	27 (0X1B)	CCW Compliance Margin	CCW Compliance margin	RW	1 (0X01)
	28 (0X1C)	CW Compliance Slope	CW Compliance slope	RW	32 (0X20)
	29 (0X1D)	CCW Compliance Slope	CCW Compliance slope	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed (Moving Velocity)	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed (Moving Velocity)	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit (Goal Torque)	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit (Goal Torque)	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position (Present Velocity)	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position (Present Velocity)	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	32 (0X20)
	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)

Figure 38 – Control Table of AX-12A Dynamixel Servo Motor [4]

6.2 LabVIEW

6.2.1 VI

Program developed in LabVIEW is called VI which stands for Virtual Instruments. This is due to the graphical user interface of the LabVIEW which allows users to replicate physical instruments such as switches, knobs, indicator lights, oscilloscopes, etc. and can be used to analyse and control the system [8]. A VI consists of two user interface a Front Panel and a Block Diagram which will be discussed further in the sections below.

6.2.2 Front Panel

One of the user interfaces of a VI is the front panel. The front panel is constructed using indicator and controls, which are also known as output terminals and interactive inputs. An entire set of different functions and instruments including the controls and indicators are available on the control palette of the VI. Some of these functions include push buttons, knobs, dials, which work as input/control mechanisms and LED lights, Gauges, and graphs, which work as output/indication mechanism. The functions or instrument used on the front panel can intake data and provide it to the functions on the block diagram of the VI. The instruments or functions which are used to indicate/output data on the front panel, does the opposite and retrieves and displays data from the functions on the block diagram [9].

6.2.3 Block Diagram

The block diagram is just the code used to control the VI. In LabVIEW, the difference is that the block diagram consists of object and is connected resembling a flow chart instead of just having lines of code like other programming languages [8]. The components placed on the front panel appear as icons on the block diagram. Using wiring connections, data can be sent and retrieved from the front panel and used to operate the program.

6.2.4 Control Palette

The Controls palette, consists of several components which acts as indicators and controls and are used to assemble the front panel of the VI. When an item from the control palette is placed onto the front panel, its icon with its inputs and outputs already appear on the block diagram.

Under the Controls palette, there are three sub palettes which are the most commonly used. These are as follows:

- Numeric sub palette

Under the numeric sub palette, there are two components which are used the most and are the simplest way to input and display numeric value. Those are – “Numeric indicator” and “Numeric control”. Other components such as the knobs, gauges and sliders are also present and can be used to create a more graphical interface.

- Boolean sub palette

The Boolean sub palette contains Boolean controls and indicators. Some of the available Boolean inputs are Push Button, Toggle Switch, and Stop button, and outputs are LED lights.

- String & Path sub palette

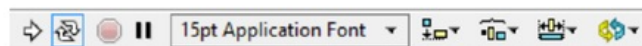
This sub palette includes the Combo Box, the String Controls, and many other functions.




6.2.5 Function Palette



The Functions palette is the palette that could only be found on the block diagram. This palette includes the functions and VIs which would assist in the construction of the block diagram.

6.2.6 Toolbar


The LabVIEW toolbar has many buttons, each of them dedicated to perform a particular function. The behaviour of each have been described below.



- The Run button  would be used to run the VI for just a single iteration. However, this would only be possible if the arrow is white and solid.
- When block diagram is being edited or created, if any part of the diagram seems to contain an error, the Run button would be appearing as broken . The list of errors can be found on the Error list window which shows all the warnings and errors.
- To continuously run the VI, the Run continuously button  have be need to be pressed. Once pressed, the VI will continuously run until it is stoped.

- While the VI is operating, a stop button  will appear red and can be used to Abort Execution.
- To pause running a VI, press the Pause button . When this is done, the LabVIEW will highlight that part of the block diagram where the execution has been halted. As a result, the user will know exactly where to start continuing working on the VI.

6.2.7 Help

The Help button , which is also known as the Context Help window, will help the user attaining more detailed information regarding LabVIEW objects. If help is required, select a particular object, and click the help button to view all the details of that object.

Chapter 7 Lab Experiment Procedure

7.1 Getting Started with Dynamixel

Robotis, the manufacturer of Dynamixel have their own software “Roboplus 1.0” which allows to connect to the internal EEPROM of the motors and configure the motors to serve the required purpose. This software should already be installed on the university computers, however, if it is required to be used on a personal computer, it can be downloaded for free from the Robotis website (the latest version of robotis Roboplus 1.0 can be downloaded from the following link -

http://en.robotis.com/BlueAD/board.php?&bbs_id=downloads&key=&keyword=&sort=&cate=SOFTWARE&sort=down).

Roboplus 1.0 has a program in it called the Dynamixel Wizard which needs to be used to configure the motors. Using the Dynamixel Wizard connect to each motor, and set them up with a unique Motor ID (this should have already been done during the assembly process of the robotic arm). Any limits, modes and alarms that are pre-set on the motor will be visible here and can be changed to meet your requirement.

Dynamixel motors can be controlled using many different software such as: Matlab, Arduino Language (C/C++), LabVIEW, etc. The exercises in this document is based on the software

LabVIEW which was developed by National Instruments. LabVIEW should already be installed on the university computers, and a free license for the university students are available to use on the university network. To use LabVIEW in a personal computer, outside the university network, a student license can be obtained from National Instruments for a period of six months. If LabVIEW is installed in a personal computer, the same version as the university should be installed or else the programs may not work when transferred (the instructions to download the latest version of LabVIEW can be downloaded from the following link - <http://www.ni.com/tutorial/13413/en/>).

The first task required to establish connection to the dynamixel motors using the USB2DYNAMIXEL is the NI-VISA Driver. The driver should already be installed in the university computers. However, to use a personal computer, the driver can be downloaded from the website of National Instruments using a student license (the latest version of the driver can be downloaded from the following link - <https://www.ni.com/nisearch/app/main/p/bot/no/ap/tech/lang/en/pg/1/sn/catnav:du.n8:3.25.123.1640.ssnv:ndr/>).

A Certified LabVIEW Plug and Play Instrument Driver, is already available for Dynamixel motors at the National Instruments website. The driver can be downloaded from the website of National Instruments (the latest version of the driver can be downloaded from the following link - http://sine.ni.com/apps/utf8/niid_web_display.download_page?p_id_guid=86E2A39663C26077E04400144FB7D21D).

Once downloaded, the Dynamixel Motor file needs to be unzipped and then saved on

C Drive > Program Files > National Instruments > LabVIEW 20** > instar.lib > Dynamixel Motor

Or

C Drive > Program Files (x86) > National Instruments > LabVIEW 20** > instar.lib > Dynamixel Motor

Note: Program Files or Program Files (x86) depends on which version of LabVIEW is installed (32-bit or 64-bit).

The driver comes with an initialization function which allows LabVIEW to establish the connection with the Dynamixel motors and also comes with a library of function palettes. The function palettes consist of functions which allow to configure, operate, and monitor the performance of the motors. Some of the functions that are available and would be essential for operating the motors are listed below:

Table 1 Dynamixel Initialize Functions





Function	Icon	Inputs	Outputs	Description
Dynamixel Initialize		<ol style="list-style-type: none"> VISA Source Name Baud Rate Forced Baud Rate Error In Timeout 	<ol style="list-style-type: none"> VISA Source Name Out Error Out 	Initializes Dynamixel, establishes connection to the COM ports, and sets Baud rate
Dynamixel Close		<ol style="list-style-type: none"> VISA Source Name Error In 	<ol style="list-style-type: none"> Error Out 	Closes Dynamixel

Table 2 Dynamixel Configuration Functions

Function	Icon	Inputs	Outputs	Description
Configure Safety Setting		<ol style="list-style-type: none"> VISA Source Name Motor ID Alarm Settings Safety Limits Error In 	<ol style="list-style-type: none"> VISA Source Name Out Error Out 	Allows to configure the alarm settings of the Dynamixel motor and to set the safety limits such as Temperature, Voltage and maximum torque limits.
Configure Angle Limit		<ol style="list-style-type: none"> VISA Source Name Motor ID 	<ol style="list-style-type: none"> VISA Source Name Out Error Out 	Allows to configure the angle limit of the motor. In joint mode, the angle limit is set at 0-300





		3. Clockwise Angle Limit 4. Counter-Clockwise Angle Limit 5. Error In		degrees by default. This can be changed using this sub VI. If limit is set to 0-0 the motor changes to wheel mode.
Configure Active		1. VISA Source Name 2. Motor ID 3. Active Compliance Setting 4. Error In	1. VISA Source Name Out 2. Error Out	Allows to configure the slope and margin of the motor acceleration.

Table 3 Motor Operation Functions

Function	Icon	Inputs	Outputs	Description
Toggle Torque		1. VISA Source Name 2. Motor ID 3. Torque off 4. Error In	1. VISA Source Name Out 2. Error Out	Turn the torque of the motor on and off.
Move Motor		1. VISA Source Name Motor ID Goal Position (0 – 300 degree in joint mode) Goal Speed (RPM) Error In	1. VISA Source Name Out 2. Error Out	Operates the motor to move to its goal position at the goal speed designated.
Query Movement		1. VISA Source Name 2. Motor ID 3. Error In	1. VISA Source Name Out 2. Motor Moving	Checks and displays if the motor is in motion or not








			3. Error Out	
Read Motor Position		1. VISA Source Name 2. Motor ID 3. Error In	1. VISA Source Name Out 2. Error Out	Checks and displays the current position of the motor
Read Motor Velocity		1. VISA Source Name 2. Motor ID 3. Error In	1. VISA Source Name Out 2. Error Out	Checks and displays the current velocity of the motor in rpm. It displays value 1-114 for anti-clockwise movement and 115-228 for clockwise movement (for clockwise, the output need to be subtracted by 114 to receive the actual output).
Read Applied Torque		1. VISA Source Name 2. Motor ID 3. Error In	1. VISA Source Name Out 2. Error Out	Checks and displays Motor Load/ Torque (%)

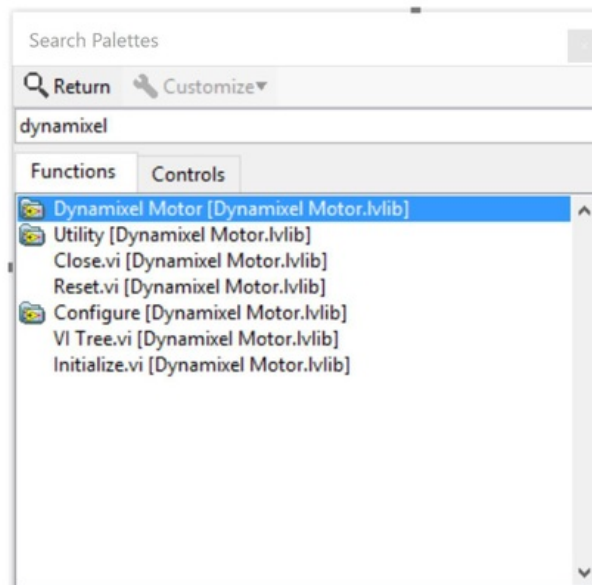
Table 4 Dynamixel Utility Functions

Function	Icon	Inputs	Outputs	Description
Scan for Motors		1. VISA Source Name 2. Start ID (0) 3. End ID (253) 4. Error In	1. VISA Source Name Out 2. Motor ID's Found 3. Error Out	Scans for all the motors connected in the system and displays them in a array.
Change Motor ID		1. VISA Source Name	1. VISA Source Name Out 2. Error Out	Changes current motor ID to a new motor ID.

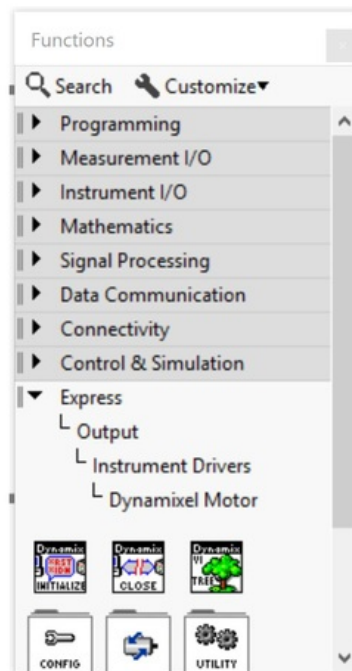
		2. Current Motor ID 3. New Motor ID 4. Error In		
Query Temperature		1. VISA Source Name 2. Motor ID 3. Error In	1. VISA Source Name Out 2. Temperature (C) 3. Error Out	Checks and displays the temperature of the selected motor.
Query Voltage		1. VISA Source Name 2. Motor ID 3. Error In	1. VISA Source Name Out 2. Voltage (V) 3. Error Out	Checks and displays the voltage acting on the selected motor.

7.2 Stablishing Connection via the USB COM Ports

1. Open a new blank VI from the file tab.
2. The Blank VI should open a front panel and a block diagram window. (if block diagram does not open, click anywhere on the front panel and press ctrl and e together)
3. Right-Click anywhere on the Block Diagram window to open the function palette
4. Open up structure, and select a while loop.
5. Place the while loop on the Block Diagram.
6. Right-Click anywhere on the Block Diagram window to open the function palette
7. On the top right corner of the function palette, click on search, and search for Dynamixel.



8. Click on “Dynamixel Motor [Dynamixel Motor.lvlib]” to open the Dynamixel Function palette



a.

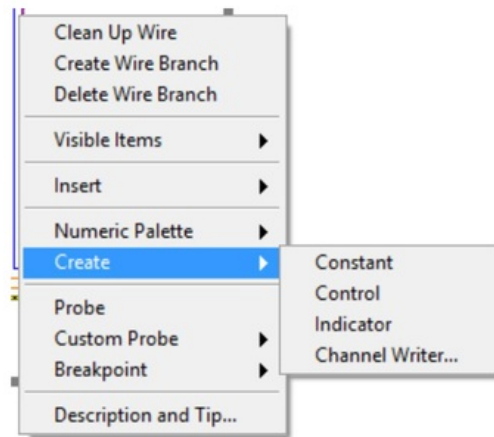
9. Place Dynamixel Initialize and Dynamixel Close function on either side of the loop, and connect the “VISA Source name out” and “error out” of Dynamixel initialize to the “VISA source name in” and “error in” of Dynamixel close, through the loop.
 - Right-Click on the input VISA resource name →create →control
 - Right-Click on the input Baud rate →create →control
 - Right-Click on the input error in →create →constant
 - Right-Click on the Conditional Terminal of the while loop →create →constant

Note: The control inputs will now be visible on the front panel and can be controlled as desired. These steps will initiate the Dynamixel driver and allow connection to the Dynamixel motors via the selected USB communication ports of the computer.

7.3 Controlling a Dynamixel Servo Motor

At first we need to establish communication with the Dynamixel motors through the USB2Dynamixel device. To do so, please follow the steps in section 7.2 – Establishing connection via USB COM ports.

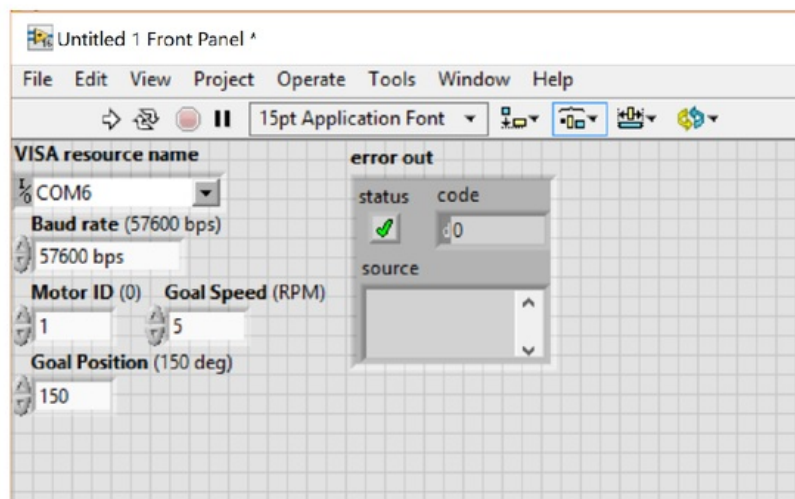
1. Right-Click anywhere on the Block Diagram window to open the function palette
2. On the top right corner of the function palette, click on search, and search for Dynamixel.
3. Click on “Dynamixel Motor [Dynamixel Motor.lvlib]” to open the Dynamixel Function palette
4. Open the Motor Operation sub palette to find all the input and output functions available for the Dynamixel motors.
5. Place the function “Move Motor” from the palette into the while loop on the block diagram.
6. Connect the VISA source name input of the “Move Motor” function to the VISA source name of the “Dynamixel Initialize” passing through the loop. This directs the functions to the communication port used to connect to the Dynamixel motors, via the USB2Dynamixel device.
7. Set the control parameters to control the motion of the motor.
 - Right-Click on the input Motor ID →create →control (or constant if you want to have a fixed motor ID)
 - Right-Click on the input “Goal Position” →create →control
 - Right-Click on the input “Goal Speed” →create →control (or constant if you do not want to a changeable motor speed)



Note: The Goal Speed should have an input greater than “0”. Having the value “0” or no inputs, the goal speed will be set to maximum by default.

- Right-Click on the input “error in” →create →constant
- Right-Click on the output “error out” →create →Indicator

Rearrange the blocks to have a clear view of the icons or click on the Clean Up Diagram function on the tool bar for LabVIEW to automatically rearrange and clean up the blocks.



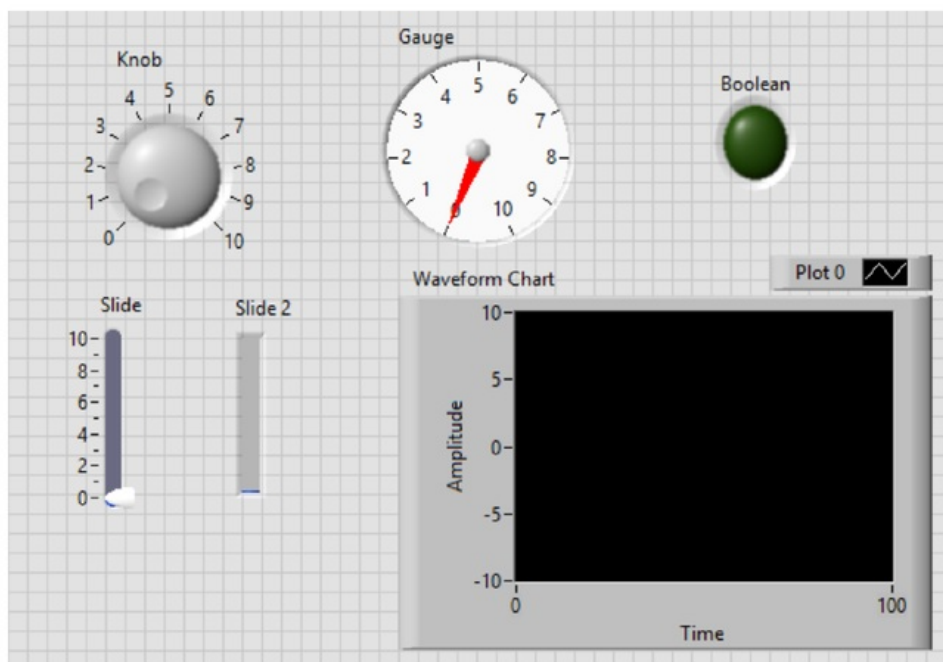
14. Verify your connection to the Robotic Arm, and ensure that the arm is properly secured to the table top and is at a clear distance from you.
15. Click on “Run” to start operation.
16. Change the Goal Position and Goal Speed from the front panel to designate a new location and speed for the axis and verify the correct operation of the system.

7.4 Using Input and Output Instruments

The advantage of using LabVIEW is its user friendly graphical user interface which consists of many pre-set control and indication instruments. Some pre-set instruments available are, Knob, Gauges, Buttons, LED, Graph generators etc. In this section, how some of these instruments can be used for the robotic arm will be shown. This section is a continuation of the section 7.3.

1. Right click anywhere on the front panel (this will open the control palette).
2. For this example, the following instrument will be used from the control palette:
 - Numeric → Knob (for Goal Position Control)
 - Numeric → Gauge (for present position indicator)
 - Numeric → Vertical Pointer Slide (for Goal Speed Control)
 - Numeric → Vertical Progress Bar (for present Speed indicator)
 - Boolean → Round LED (for motor on indicator)
 - Graph → Waveform Chart (for plotting the velocity time graph)

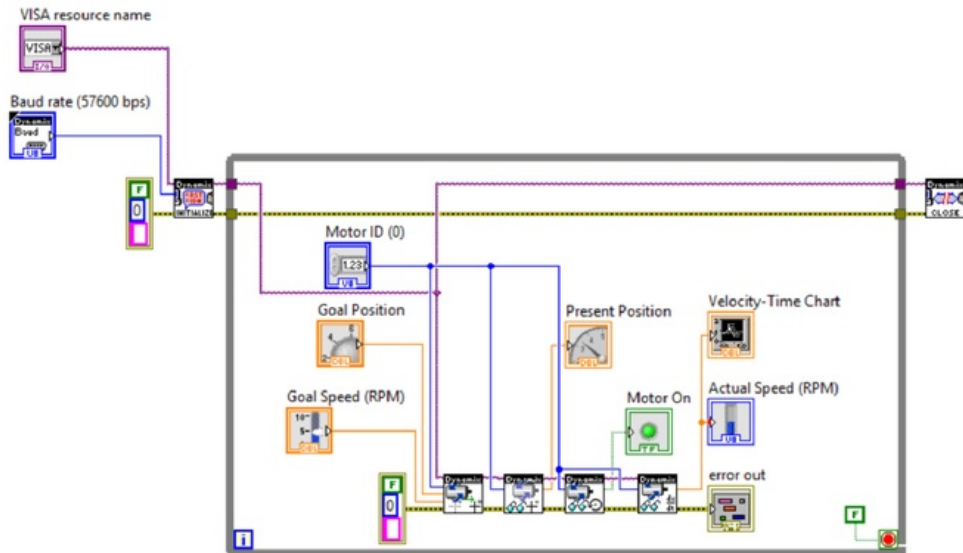
(Feel free to browse through and select any appropriate instrument that you would like to use for your VI.)



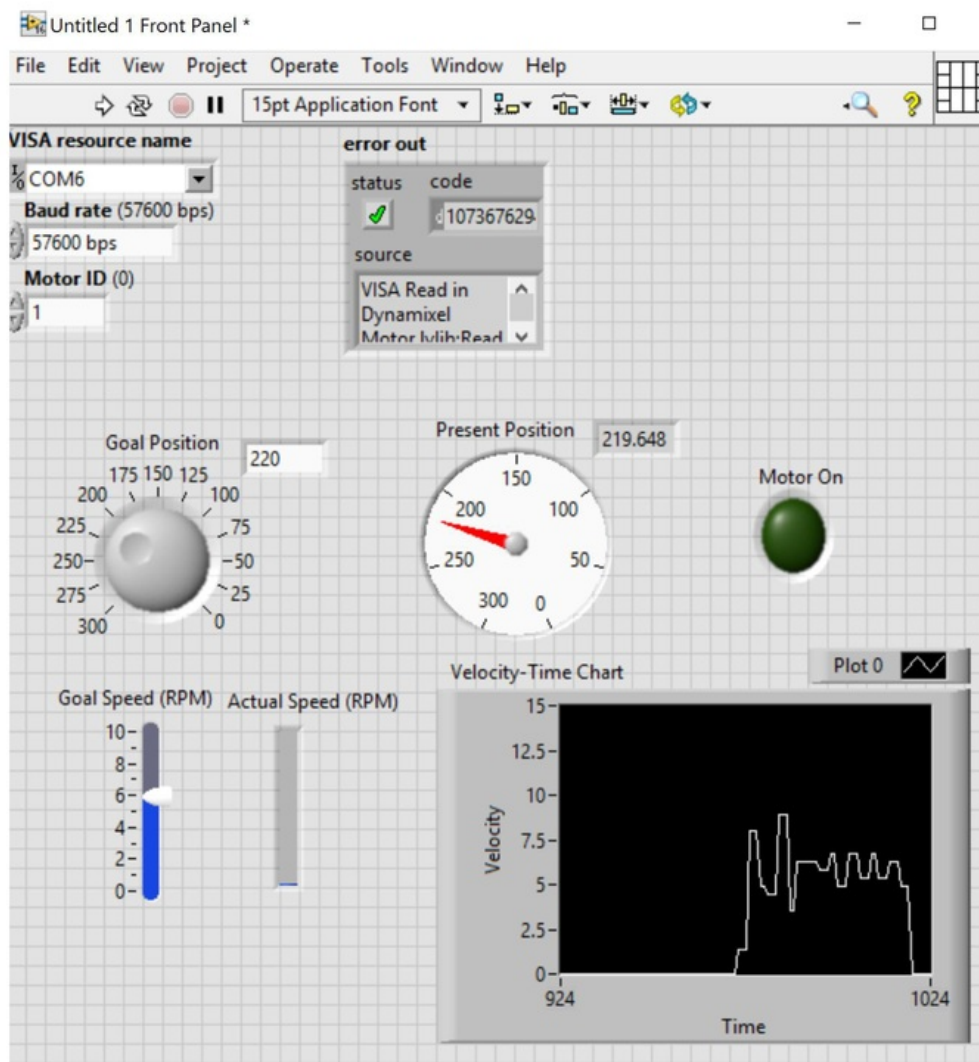
3. Select the knob. Right click on the knob to open Properties from the menu.

- Appearance Tab → Label the knob and customize the appearance as you like
 - Scale Tab → Select appropriate scale that suits best for your operation. (For example, in this program, you can set a range of minimum “0” and Maximum “300”)
4. Similarly configure the other instruments to suit the purpose of controlling the functions of the motor. Depending on your requirement, you can choose to set your instruments to any configuration or appearances as you like.
 5. The instruments opened and setup on the front panel will also be visible on the block diagram. You can also locate the instrument icon on the block diagram by selecting the instrument on the Front panel → Right click → Find Terminal.
 6. In the block diagram, initialise Dynamixel, select the COM port to which the USB2Dynamixel is connected, and set up the while loop as shown in section 7.2
 7. The instruments now need to be connected to their respective inputs and outputs.
 8. The instruments selected above would require the following Dynamixel functions:
 - Move Motor
 - Read Motor Position
 - Query Movement
 - Read Motor Velocity
 9. Right-Click anywhere on the Block Diagram window to open the function palette
 10. On the top right corner of the function palette, click on search, and search for Dynamixel.
 11. Click on “Dynamixel Motor [Dynamixel Motor.lvlib]” to open the Dynamixel Function palette
 12. Open the Motor Operation sub palette to find all the input and output functions available for the Dynamixel motors.
 13. Place the functions in step 8 from the palette into the while loop on the block diagram.
 14. Connect the knob and the Vertical Pointer Slide to the goal position and goal speed of the Move Motor function.
 15. Connect the Gauge to the “Position” output of the “Read Motor Position” function.
 16. Connect the Vertical Progress Bar and the Waveform Chart to the “Velocity” output of the “Read Motor Position” function. (Outputs can be connected to multiple indicators)

17. Connect the Round LED to the “Motor Moving” output of the “Query Movement” function.
18. Connect the same Motor ID to all the functions.
19. Complete connecting the VISA Source name, error in, and error out in all the functions.



20. Verify your connection to the Robotic Arm, and ensure that the arm is properly secured to the table top and is at a clear distance from you.
21. Click on “Run” to start operation.
22. Change the Goal Position and Goal Speed from the front panel to designate a new location and speed for the motor and verify the correct operation of the system.



7.5 Controlling the Robotic Arm

The Crust Crawler AX-12A Smart Robotic arm has four axes of freedom and a gripper as an end effector. The arm is constructed using seven Dynamixel AX-12A servo motors where axis 2 and axis 3 is operated using two servos each, axis 1 and axis 4 is operated using one servo each, and the gripper is operated using one servo.

In the previous exercise, the method to control a single Dynamixel motor was shown. Similar method to the previous section has to be used to construct a VI (virtual instrument) which will be able to control all four axes of the robotic arm and also the gripper.

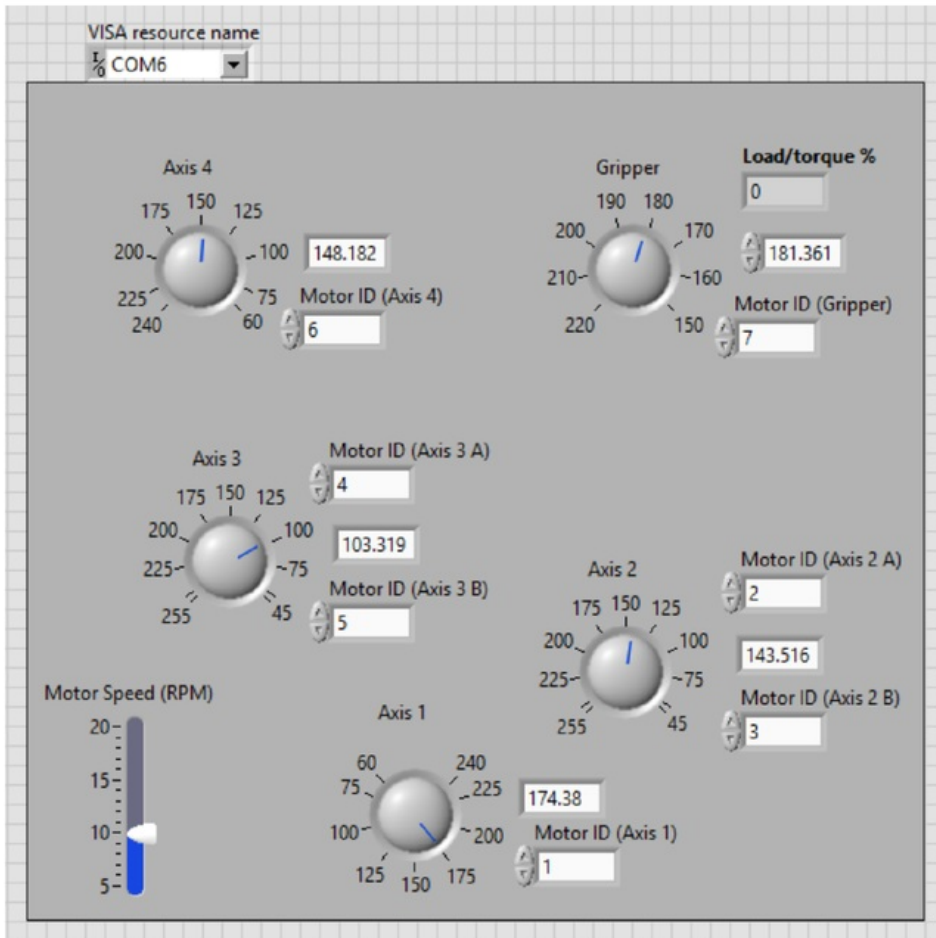
The steps required for this exercise are:

1. Check and ensure the Motor ID's of all the motors in the robotic arm using the Dynamixel Wizard program from the RoboPlus software.
2. Ensure that all the motors have their own unique ID between 0 and 253 and is not similar to any other motors connected to the system.
3. Open a new blank VI on LabVIEW
4. In the block diagram, initialise Dynamixel, select the COM port to which the USB2Dynamixel is connected, and set up the while loop as shown in section 7.2
5. Since Axis 1, Axis 4, and the gripper consists of one motor each, follow the steps from section 7.4 (Controlling a Dynamixel Motor) to create the control and indicator functions required to operate the motors.
6. A single instrument can be set for Goal Speed can be wired to all the motors. (to individually control the speed of motor, use separate control instrument)
7. Choose any appropriate instruments to control and indicate the inputs and outputs.
8. For Axis 2 and Axis 3, firstly ensure that the motors are assembled correctly and are facing the same direction in each axis.

(Note: Motors facing opposite direction would require opposite Goal positions as input values and therefore would be harder to program. The motors could also get damaged if they act against each other and are connected to the frame.)

9. Setup two "Move Motor" function for Axis 2, the same way it was setup for controlling a single motor. The only difference for this axis would be that the "Goal Position" and the "Goal Speed" for both the motors should be the same.
10. Repeat the same process for Axis 3 as well.
11. Set appropriate limits for all your instruments.

12. A sample view of the Front panel and the block diagram is shown below:



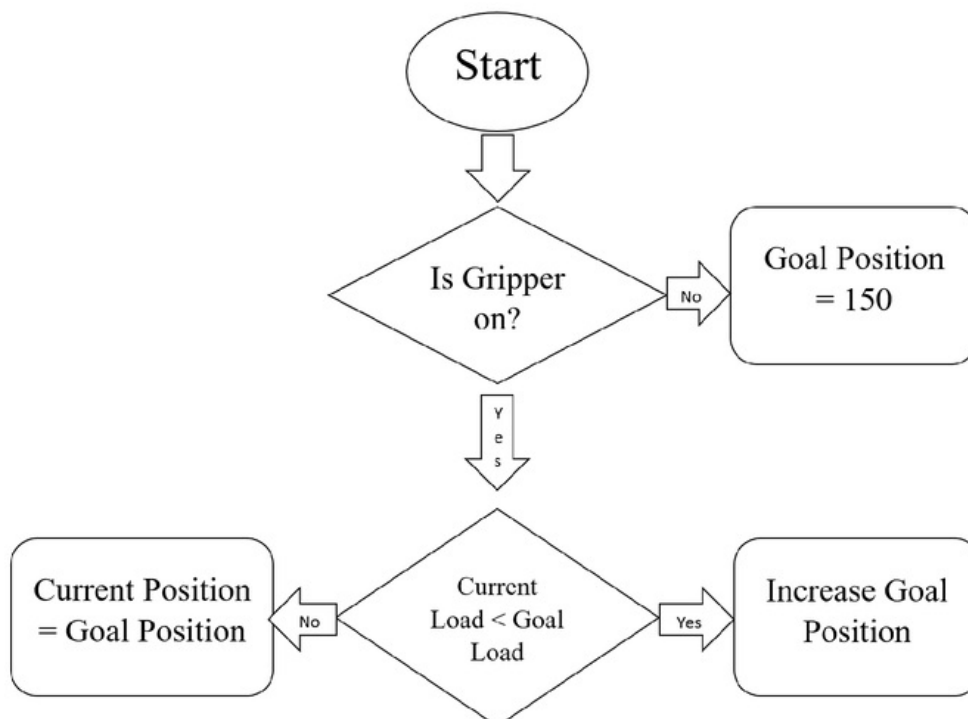
13. Operate all the axes of the robotic arm by changing their “Goal Position” and the “Goal Speed” and verify the correct operation of the system.

7.6 Controlling Gripper using torque

WARNING: The Servo motors are not designed to hold on to the Load/Torque for a long period of time. Gripping on for a long period of time will cause the motors to overheat and get damaged. In case of a servo overheating, thermal shutdown will be initiated by the controller of the servo to protect the motor. Do not immediately reset the motor. Shut down the power and allow the motor to cool down for at least a few minutes to avoid permanent damage to the servo motor [2].

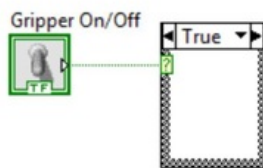
Note: The Motor Load/ Applied Torque value retrieved from the motor is just the current load and not measured by a torque sensor. Therefore, this can only be used as a reference and is not the actual load/ torque applied by the motor.

Controlling the gripper using the torque is a very useful and this would enable the gripper to grip on to any sized object within the grippers limits. Being able to control the applied load would also allow the freedom to grasp light and fragile objects with less force, and Heavy and rigid objects with more force. The process that can be used to accomplish this task is described on the flow chart below:



The following steps can be used as a guideline to achieve the expected outcome of this exercise:

1. Open a new VI. (This exercise can also be done on the previous VI developed to control the robotic arm)
2. Insert a Push Button or a Toggle switch from the Boolean sub palette of the controls palette onto the front panel. (This will be used to turn the gripper on and off)
3. Insert any appropriate instrument to input the goal load percentage of the motor.
4. In the block diagram, initialise Dynamixel, select the COM port to which the USB2Dynamixel is connected, and set up the while loop as shown in section 7.2
5. Now on the Block Diagram, Right Click anywhere to open the Function Palette and select Case Structure from the Structures sub palette. Place the case structure on the block diagram.
6. The case structure now requires a condition, which now for this exercise is to check if the gripper is turned on or off. This can be achieved by connecting the Push Button, or the Toggle Switch to the condition input of the case structure (the green question mark symbol). What this does now is that, whenever the switch is turned on, the true section of the case structure will be implemented, and when turned off, the false section of the case structure will be implemented.



7. According to the flow chart, the false section should therefore have the function to drive the motor to Goal Position 150 (which is the location at which the gripper opens fully).
8. The True section should now have another case structure to compare the Current Load and the Goal Load.
9. Right Click anywhere on the Block diagram to open the function palette and use appropriate comparators from the Comparison sub palette to compare the two loads and use them for the case structure.
10. Similarly, implement both the cases mentioned on the flow diagram onto the case structure.

11. Run the program to assess the performance and correct operation. Add additional conditions, or structures to further improve the operation performance.

Note: One error that should clearly be noticed is that the gripper would be twitching while gripping. Taking a better look at the load acting on the motor would show a rapid fluctuation in load due to gripping. This will cause the case to continuously toggle and therefore the gripper would want to grip and also let go at the same time. Setting up a range of value for the goal load (e.g. $\pm 10\%$ of the set value) using a In Range Comparator can be used as a solution for this problem.

7.7 Position Feedback

This exercise will go through the procedure of taking the feedback from a motor and use it to control another motor. Once developed for a single motor, the VI then needs to be extended to accommodate a complete robotic arm. For this, two robotic arm would be required; one with Motor IDs 1 to 7 and another one with Motor IDs 8 to 14.

1. Open a new blank VI.
2. In the block diagram, initialise Dynamixel, select the COM port to which the USB2Dynamixel is connected, and set up the while loop as shown in section 7.2
3. Now to drive a motor the required inputs are Motor ID, Goal Speed, and Goal Position. Therefore, the Goal Speed and the Goal Position values need to be retrieved from the driving (feedback) motor and implemented onto the driven motor. (The driving and driven motors should have their own unique motor ID)
4. The functions that would be required for this task are:
 - Move Motor
 - Read Motor Position
 - Read Motor Velocity

From the Dynamixel Motor sub palette of the function palette, select and place the functions onto the loop of the Block Diagram.

5. Connect the VISA source name, error in and error out with appropriate controls, indicators, and constants.
6. Read Motor Position and Read Motor Velocity should be from the driving motor, therefore, create a control for the Motor ID and connect it parallelly to both the functions. Label the Motor ID Control as "Axis 1 Feedback Motor ID".
7. Create another control for the Motor ID of the driven motor. Label the Motor ID control as "Axis 1 Driven Motor ID".

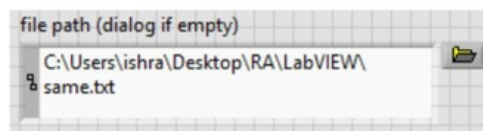
Note: The manual of the AX-12A Dynamixel motors state that the motor takes in a value between 0-114rpm for the Goal Speed. However, while the Speed is read from the feedback of the motors, the output value is 1-114 for rotations in Counter Clockwise direction and 115-228 for rotations in Clockwise direction. Connecting the Motor Velocity output directly to the input Goal Speed would cause an error when the motor is rotated in Clockwise direction.

8. Create a Case Structure from the Function Palette.

9. Use available comparator to generate a condition which would compare the Motor Velocity value and allocate a case for it.
10. For, values less than 114, the output should be the same. And for values more than 114, the output should be the value minus 114.

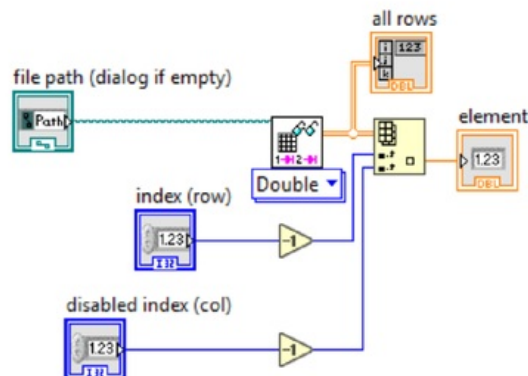
11. Run the VI to test the correct operation of the system. (moving the motor providing the feedback should move the driven motor as well)
12. Follow the same procedure to implement the control to all the axis of the robotic arm.

Retrieving data from a file is very useful and can be used to compute or control components on a VI. For a robotic arm, this could be used to store sequence of positions and can be retrieved to operate the arm. The function used for this exercise is “Read Delimited Spreadsheet” which reads and displays tab delimited text files.



all rows									
0	a	a	a	a	a	a	a	a	a
1	150	255	45	150	150	150	255	45	150
	240	255	45	150	150	240	255	45	150
	240	140	75	150	150	240	140	75	150
	240	140	75	150	180	240	140	75	150
	240	220	45	150	180	240	220	45	150
	60	220	45	150	180	60	220	45	150
	60	220	45	150	180	60	220	45	150
	60	220	45	150	180	60	220	45	150
	60	220	45	150	180	60	220	45	150
	Restart								

5. Create a test text file using tab-separated values.
6. Use the file path control from the front panel select the test file.
7. Run the program to verify the correct display of the data on the array table.
8. To access just a single value from the table; from the function palette, search “Index Array Function” and place it on the block diagram.
9. Connect the “all rows” output to the “array” input of the index array function.
10. The icon will change and now have 2 inputs:
 - Index – which is the row (need to put in row number – 1 as indexing starts from 0)
 - Disabled index – which is the column (need to put in column number – 1 as indexing starts from 0)
11. Create controls for both the row and the column and subtract the input with one using the arithmetic functions available. (this would register the correct index when the row and column number is selected from the front panel)



12. Create an indicator for the element output.
13. Select the row and column from the front panel and run the program to verify.

7.9 Performing Repetitive or Conditional Operation

The aim of this exercise is to provide basic guide to develop a program which would enable a robotic arm to perform an operation following steps in sequence. The operation can be repetitive or based on conditions, depending on the requirement. A complete solution for a Repetitive operation of two robotic arms together is available in Appendix 10.2.

1. Open a new blank VI on LabVIEW
2. In the block diagram, initialise Dynamixel, select the COM port to which the USB2Dynamixel is connected, and set up the while loop as shown in section 7.2.
3. Follow the instructions provided in section 7.8 to create a function which would read a delimited text file.
4. Create a tab delimited text file with the rows containing the steps of the operation and the columns containing the Goal Position of each motor.
5. Implement a case structure which when turned true would increase the row index by 1 (this will allow you to move on to the next step). If not true, the row index should stay the same.
6. The Column index can be set to a constant as each column would consist only the Goal position for a single servo/axis.
7. Without connecting to the Move motor function, try to run the program to verify the correct operation of the case structure and the output values. (At this stage, a Boolean controller such as a pushbutton can be used to test the case operation)
8. After verifying the values, implement the "Move Motor" function and test to verify the correct operation of the motor.
9. Implement a "Read Motor Position" function.
10. Create a comparator to compare the current position to the goal position of the motor. (Remove the Boolean controller from the case structure and replace it with the output of the comparator) This would now enable the case structure to turn true when the current position of the motor is equal to the goal position. Therefore that would increase the row index and a new goal position would be allocated.

Note: An "In range" function may be required to set a tolerance of 1 or 2 degrees to compare the current position as the motors do not always stop at the exact location.
11. Run the program to verify the correct operation of the system.
12. Repeat the process to implement the function on all the motors of the robotic arm.

To implement a repetition or conditional operation of the system, add appropriate conditions to the structures.

Chapter 8 Discussion

The laboratory experiments were designed in a way that a student can learn from the basics and work their way up to more complex designs. The first exercise was just to provide the students with a brief idea on what components are used and the software required to operate a Dynamixel motor and how to install all the required hardware and software. Once students understand what they would be working with, the next exercise provides instructions on how they can establish the communication connection between the motors and the software. These steps are put in separate experiments because they are very important and would be required in every experiment.

Before the students learn how to control the robotic arm, it is essential for them learn how to control a single Dynamixel motor. The third experiment does this and provides with instructions on how to control a single motor. During this process, instructions on how to create controls, constants, and indicators are also mentioned.

LabVIEW is well known for its user friendly Graphical user interface, which is why, the next experiment (experiment 4), is completely dedicated to providing instructions on how the input and output instruments such as knob, gauge, graph, and switches can be used. To get the most out of this experiment, multiple operating functions such as “Query Movement”, “Read Motor Position”, Read Motor Velocity”, and different instruments were used. Instructions on how to configure each instrument to meet the requirement is also stated.

Experiment 6 and 7 included tasks based on feedback from the motor. This would allow students to understand how feedbacks can be retrieved and used, and also what are the complications that a feedback system would create. In experiment 6, a gripping system based on the applied torque will be created, and therefore the grippers would be able to grab fragile things with less force and rigid things with more force. In experiment 7, one robotic arm will be used to provide its position and speed to the other, so that the other arm can replicate its movement and mimic it.

Finally, the last two experiments (experiment 8 and 9) consists of ways to import data from delimited text files and use them to operate the robotic arm. Steps are provided to guide them through developing a program that would allow the arms to operate in sequence repeatedly or based on conditions. The solution to experiment 9 is available in appendix 10.2.

8.1 Assembly

During the assembly, a lot of complications were faced. Proper instructions on how to align the motors at different stages were not clearly mentioned on the manufacturer's assembly guide. Some instructions were mentioned way later in the assembly process rather than when it should have been mentioned. This caused a lot of problems during the assembly and a few parts had to be taken off and reassembled. The assembly guide produced in this document is based on the manufacturer's guide, however, the assembly sequence, and procedure was changed and special instructions were placed to avoid any hassle.

8.2 Load and Overheating

The motors overheat very quickly, especially when a constant load is generated to grasp an object with the gripper. Within a few minutes of operation, the internal safety settings would cut off power to the motor as it would reach temperatures above the set limit. According to the manual provided by the manufacturer Robotis [4], every time the safety feature cut off power due to overheating, the motor must be left off for at least 10-15min to cool down and avoid any permanent damage.

According to the manufacturer, Load feedback retrieved from the motor is just to be used as a reference and is not the actual load/torque acting on the system. This could also be a reason for the motors to overheat as the feedback output of the Load/ Torque was assumed to be correct and therefore was being used to control the gripper angle. It was also observed that the Load reading was fluctuating up and down up to 20% at the same gripper angle.

8.3 Motor Output Glitches

During the period of using the motors, many glitches were observed. One was when multiple COM ports were used to operate multiple robotic arms, the feedbacks received from the motors were strange and flickering. When plotted onto a graph, it was seen that the values were going down to zero. The frequency of this happening was not proportional and was completely random. An example of the situation can be seen in figure 39 where the amplitude in the y-axis is the current position value retrieved from the motor.

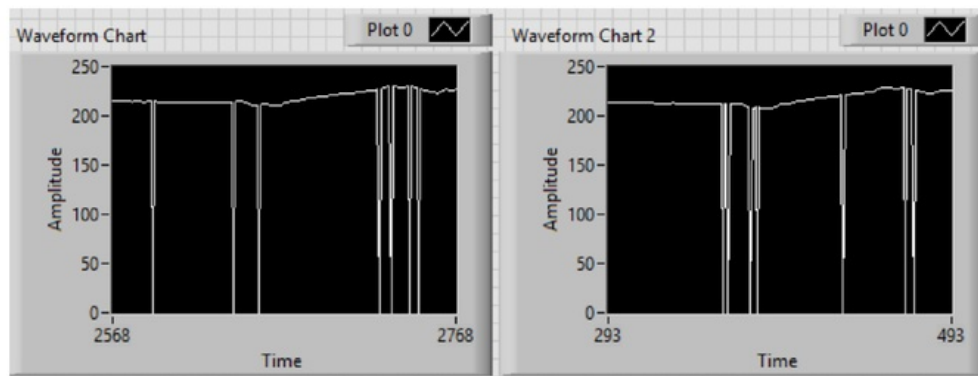


Figure 39

8.4 LabVIEW compared to MATLAB

The biggest advantage of LabVIEW is the detailed Graphical User Interface and the easily customisable front panel. For this project, it was more ideal as students would enjoy the customizable front panel instruments and can play around with many other input and output instruments. In MATLAB, these features would not be available and most computations would be conducted with values between 0-1024 as that is what is actually stored on the EEPROM and RAM of the motors and then converted to the actual unit of the value. The means to control the motion, and read feedbacks would require you to access the addresses in the RAM. The Certified LabVIEW Plug and Play Instrument Driver had allowed most function to already have the addresses and the unit conversions pre-set. However, developing the loops and storing data in MATLAB is much easier than in LabVIEW. It is very hard to setup the flow chart styled conditions and comparisons in LabVIEW, and even harder to keep track of them.

Chapter 9 Conclusion

The objective of this project was to develop a learning platform for the students. Conducting the research and experiments on the robotic arm, has helped develop nine laboratory experiments. Students will be able to learn the basics of connecting to the motor and controlling it, to more advanced tasks such as performing sequential and repetitive tasks with the robotic arm.

References

- [1] R. Schilling, Principle of Robotics, The Goodheart Willcox Co. Inc., 2013.
- [2] A. Dirks, "AX12A/AX18A Smart Robotic Arm - Kit Assembly Guide - Version 5.1," CrustCrawler Inc., 2014.
- [3] Crustcrawler, "AX-12A / AX18A Smart Robotic Arm Specifications / Advantages Rev 2.2," Crustcrawler Inc., 21 January 2015. [Online]. Available: <http://www.crustcrawler.com/products/AX-18F%20Smart%20Robotic%20Arm/docs/AX12A18A%20Smart%20Robotic%20Arm%20Specifications.pdf>. [Accessed 07 August 2016].
- [4] Robotis, "AX-12/ AX-12+/ AX-12A e-Manual v1.25.00," Robotis, 2010. [Online]. Available: http://support.robotis.com/en/product/dynamixel/ax_series/dxl_ax_actuator.htm. [Accessed 04 September 2016].
- [5] Robotis, "USB2Dynamixel e-Manual v1.27.00," Robotis, 2010. [Online]. Available: http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm. [Accessed 04 September 2016].
- [6] Robotis, "Robotis e-Manual," Robotis, 2010. [Online]. Available: <http://support.robotis.com/en/>. [Accessed 03 September 2016].
- [7] Robotis, "RoboPlus," 2010. [Online]. Available: http://support.robotis.com/en/software/roboplus_main.htm. [Accessed 05 September 2016].
- [8] H.-P. Halvorsen, "Introduction to LabVIEW," 07 September 2016. [Online]. Available: <http://home.hit.no/~hansha/documents/labview/training/Introduction%20to%20LabVIEW/Introduction%20to%20LabVIEW.pdf>. [Accessed 02 November 2016].

[9] LabVIEW, "Getting Started with LabVIEW," June 2013. [Online]. Available:
<http://www.ni.com/pdf/manuals/373427j.pdf>. [Accessed 31 October 2016].

[10] National Instruments, "Fundamentals of Motion Control," 21 July 2016. [Online].
Available: <http://www.ni.com/white-paper/3367/en/>. [Accessed 08 August 2017].

Chapter 10 Appendix

10.1 Controlling a Robotic arm

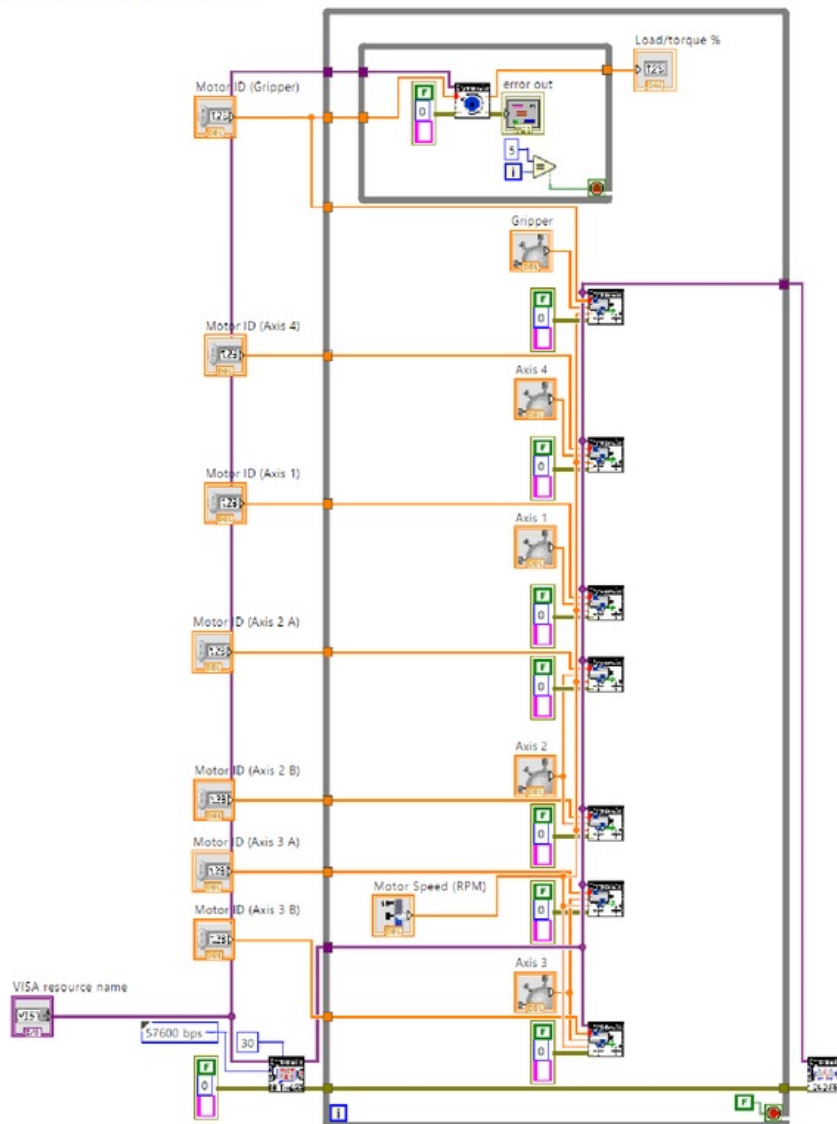
servo2.vi

C:\Users\ishra\Desktop\RA\Back up\22.08.16\servo2.vi

Last modified on 23/08/2016 at 11:00 AM

Printed on 7/11/2016 at 11:01 PM

Page 1




10.2 Performing Repetitive Operation


VISA resource name
COM6
Goal Speed (RPM)
3

Baud rate (57600 bps)
57600 bps
Tolerance
3















element
60
Numeric
0

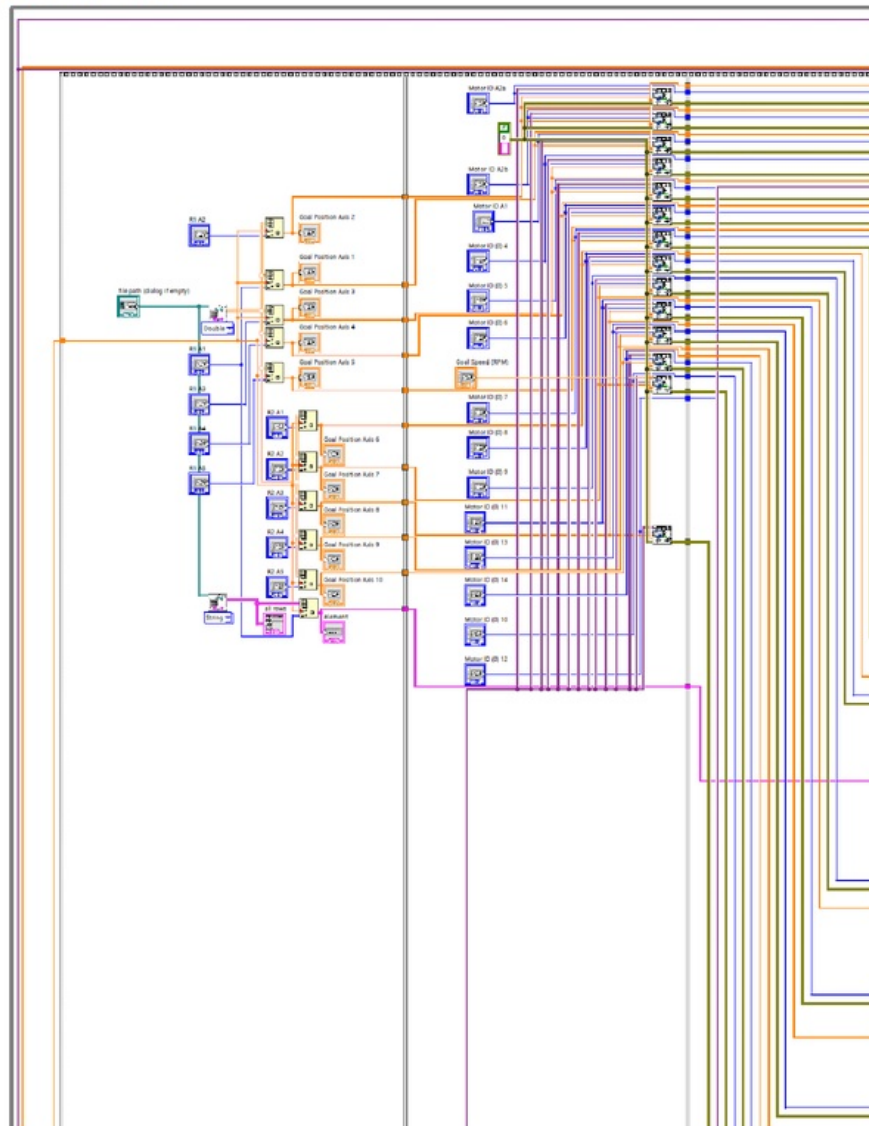
Loop Iteration
409
Step
2

All Position True


file path (dialog if empty)
C:\Users\ishra\Desktop\FA\LabVIEW\same.bt


STOP

Motor ID A1 1	R1 A1 1	Goal Position Axis 1 60	Position (deg) 208.211	Boolean 	Boolean 8 	Position (deg) 8 205.572	Goal Position Axis 6 60	R2 A1 6	Motor ID (0) 8 8
Motor ID A2a 2	R1 A2 2	Goal Position Axis 2 220	Position (deg) 2 219.648	Boolean 2 	Boolean 9 	Position (deg) 9 219.355			Motor ID (0) 9 9
Motor ID A2b 3			Position (deg) 3 219.648	Boolean 3 	Boolean 10 	Position (deg) 10 219.841	Goal Position Axis 7 220	R2 A2 7	Motor ID (0) 10 10
Motor ID (0) 4 4	R1 A3 3	Goal Position Axis 3 45	Position (deg) 4 44.5748	Boolean 4 	Boolean 11 	Position (deg) 11 43.695		R2 A3 8	Motor ID (0) 11 11
Motor ID (0) 5 5			Position (deg) 5 43.9883	Boolean 5 	Boolean 12 	Position (deg) 12 44.2815	Goal Position Axis 8 45		Motor ID (0) 12 12
Motor ID (0) 6 6	R1 A4 4	Goal Position Axis 4 150	Position (deg) 6 149.853	Boolean 6 	Boolean 13 	Position (deg) 13 149.853	Goal Position Axis 9 150	R2 A4 9	Motor ID (0) 13 13
Motor ID (0) 7 7	R1 A5 5	Goal Position Axis 5 180	Position (deg) 7 179.472	Boolean 7 	Boolean 14 	Position (deg) 14 179.472	Goal Position Axis 10 180	R2 A5 10	Motor ID (0) 14 14



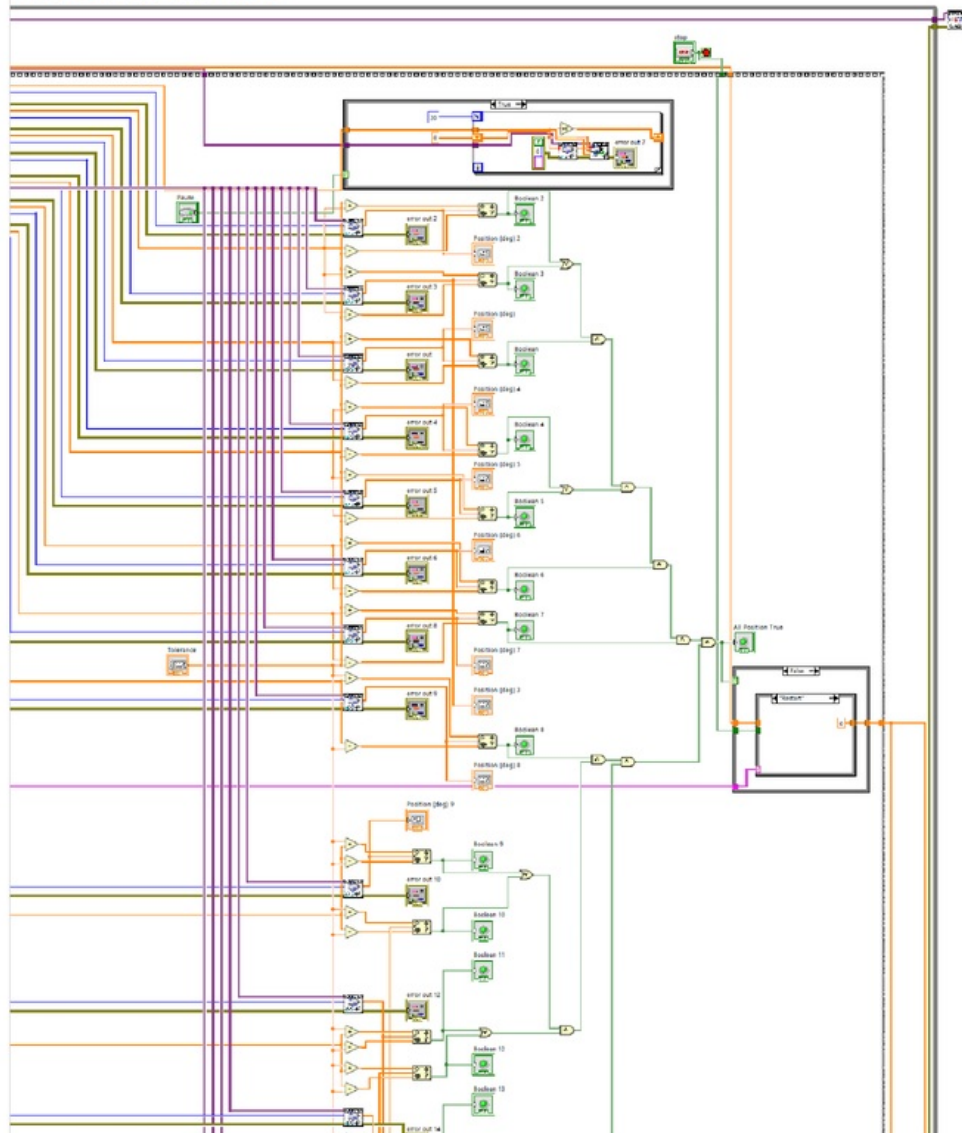
servo31.vi

C:\Users\ishra\Desktop\RA\LabVIEW\servo31.vi

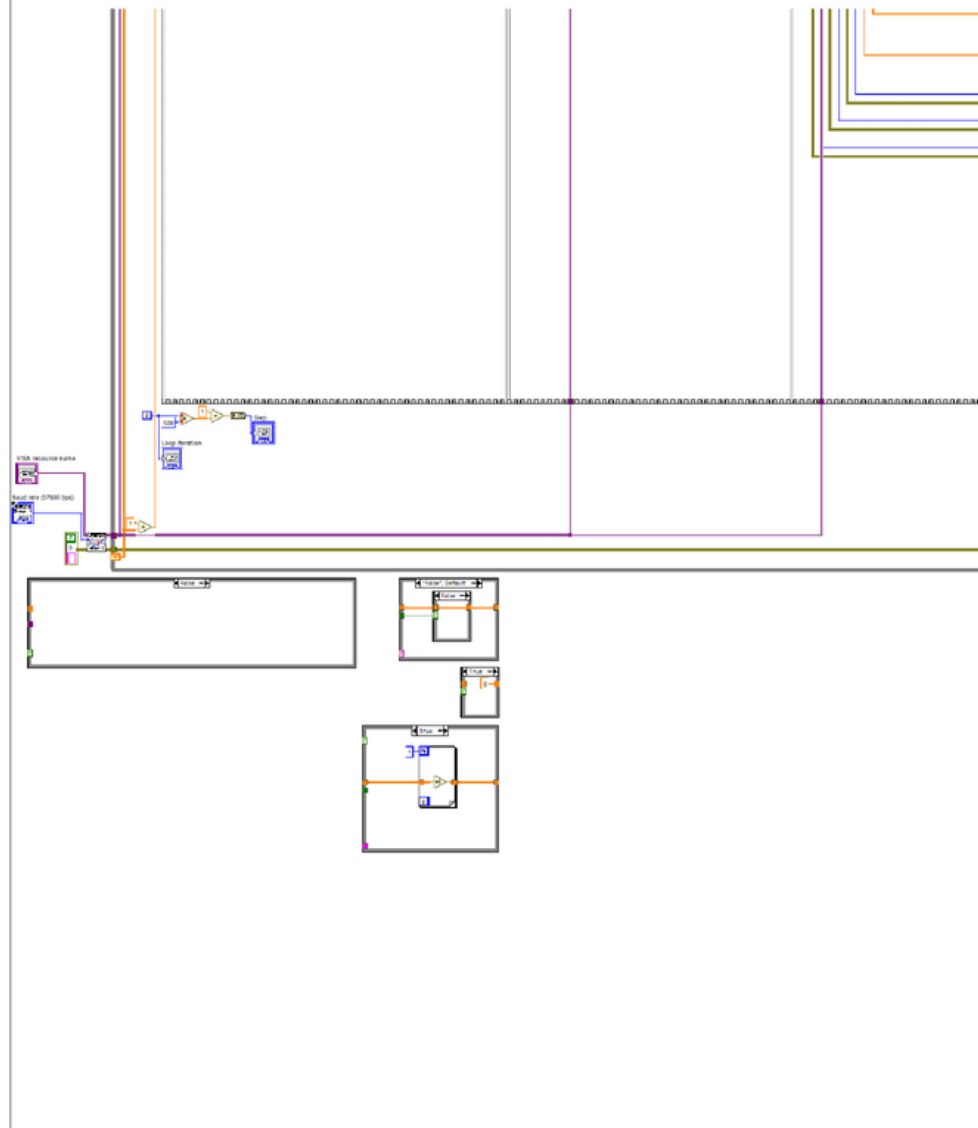
Last modified on 6/11/2016 at 5:19 PM

Printed on 7/11/2016 at 11:03 PM

Page 2



Last modified on 6/11/2016 at 5:19 PM

 Springer

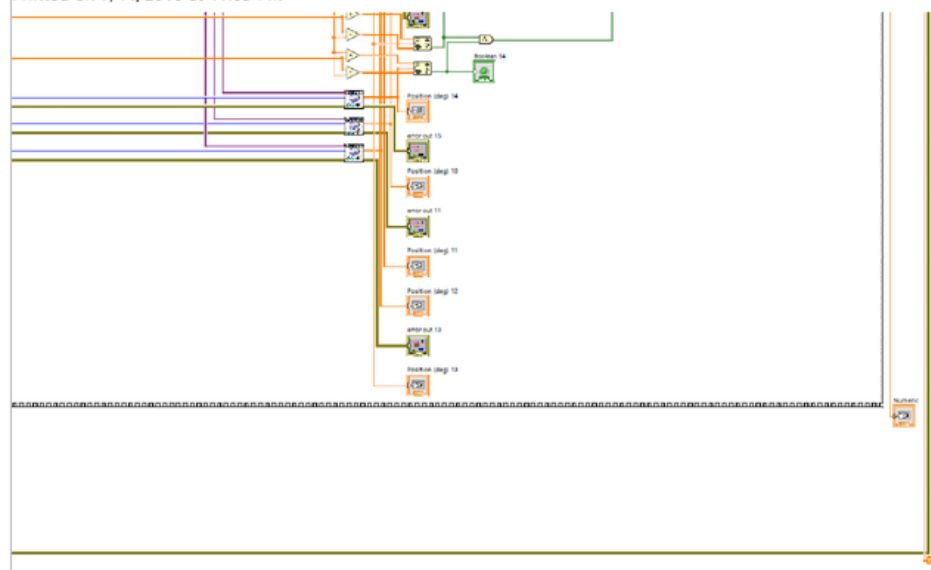
servo31.vi

C:\Users\ishra\Desktop\RA\LabVIEW\servo31.vi

Last modified on 6/11/2016 at 5:19 PM

Printed on 7/11/2016 at 11:03 PM

Page 4



Consultation Meetings Attendance Form

Week	Date	Comments (if applicable)	Student's Signature	Supervisor's Signature
1	02/08/16		Masnam	Husnopadhyay
2	09/08/16		Masnam	Husnopadhyay
3	15/08/16		Masnam	Husnopadhyay
4	22/08/16		Masnam	Husnopadhyay
5	29/08/16		Masnam	Husnopadhyay
6	05/09/16		Masnam	Husnopadhyay
7	12/09/16		Masnam	Husnopadhyay
8	06/10/16		Masnam	Husnopadhyay
9	10/10/16		Masnam	Husnopadhyay
10	17/10/16		Masnam	Husnopadhyay
11	24/10/16		Masnam	Husnopadhyay
12	03/11/16		Masnam	Husnopadhyay