0X DELTA

ROBOTIC RESEARCH PLATFORM

Software Manual

NEX ROBOTICS

Designed and Manufactured by: Nex Robotics Pvt. Ltd.

Read sentences carefully which are marked with ⚠caution symbol.

⚠Important:

User must go through hardware and software manuals before using robot.

⚠Safety precautions:

- ✔ Robot's electronics is static sensitive. Use robot in static free environment.
- ✔ Do not access any part of the robot unless robot is in the anti static environment and user is wearing anti static strap.
- ✔ If robot's battery low buzzer starts beeping, immediately charge the batteries.
- ✔ To prevent fire hazard, do not expose the equipment to rain or moisture.
- ✔ Refrain from dismantling the unit or any of its accessories once robot is assembled.
- ✔ Charge the battery only with the charger provided with the robot.
- ✔ Charge the battery in the open area and on the concrete or ceramic flooring.
- ✔ Never allow battery to deep discharge. If it is deep discharged, charger will refuse to charge the battery because of safety concerns.
- ✔ Mount all the components with correct polarity.
- ✔ Keep wheels away from long hair or fur.
- ✔ Keep your hands away from the wheels. Do not wear loose clothes while operating the robot. Loose cloth may get entangled in robot's wheels and can cause serious injury.
- ✔ Keep the robot away from the wet areas. Contact with water will damage the robot.
- ✔ To avoid risks of fall, keep your robot in a stable position.
- ✔ Do not attach any connectors while robot is powered ON.
- ✔ Never leave the robot powered ON when it is not in use.
- ✔ Before operating the robot, make sure that you have access to at least "Class A/B" type fire extinguisher.

⚠Inappropriate Operation:

Inappropriate operation can damage your robot. Inappropriate operation includes, but is not limited to:

- ✔ Dropping the robot, running it off an edge, or otherwise operating it in an irresponsible manner.
- ✔ Interfacing new hardware without considering compatibility
- ✔ Overloading the robot above its payload capacity.
- ✔ Exposing the robot to wet environments.
- ✔ Continuing to run the robot after hair, yarn, string, or any other item has become entangled in the robot's axles or wheels.
- ✔ All other forms of inappropriate operation.
- ✔ Using robot in areas prone to static electricity.

Notice

The contents of this manual are subject to change without notice. All efforts have been made to ensure the accuracy of contents in this manual. However, should any errors be detected, NEX Robotics welcomes your corrections. You can send us your queries / suggestions at info@nex-robotics.com

- **Robot's electronics is static sensitive. Use robot in static free environment.**
- **Read the Robot's manual completely before start using this robot**

Recycling:

Almost all of the robot parts are recyclable. Please send the robot parts to the recycling plant after its operational life. By recycling we can contribute to cleaner and healthier environment for the future generations.

⚠User must go through hardware and software manuals before using 0X Delta 4 wheel drive Robot.

Commonly Used Abbreviations in manual:

Short Forms	Full Forms
IR	*Infrared*
GPS	*Global Positioning System*
DGPS	*Differential Global Positioning System*
GPIO	*General Purpose Input Output*
S	*Success*
F	*Failure*
MSB	*Most Significant Byte*
LSB	*Least Significant Byte*
UL	*Ultrasonic*
WL	*White Line*
GUI	*Graphical User Interface*

⚠Important notes related to safety feature of robot:

Behavior during robot safety ON feature

1. Max velocity of robot gets limited to a safe value. This safe max velocity for 0X Delta robot is 0.4 m/Sec.

Behavior during robot safety OFF feature

1. Max velocity of robot can be altered up to robot's maximum reachable velocity.
2. Max reachable linear velocity of robot is +/- 2 m/Sec.

⚠ **In this manual, velocity is mentioned in two different ways, Motor velocity and Robot velocity. Motor velocity is different from Robot velocity. Motor velocity is effective for that particular motor i.e. either left motor or right motor. However, when it comes to robot velocity, it will be velocity of robot and not of the individual motor.**

Table of Contents

1 Introduction

0X Delta 4 Wheel Drive robot is a reliable, versatile, rugged differentially steered all-terrain robot vehicle for indoor/outdoor robotic research and application development. It's easy to use and intuitive interface gets you started on development very quickly. It's unique architecture allows it to be used in many areas of applications such as Mapping and autonomous navigation, Collaborative robotics, tele-presence and many more. This robot supports many optional accessories such as laser range finder based navigation, vision based stereo range finders, integrated inertial correction to compensate for slippage, digital compass, GPS/DGPS receiver, support for Manipulators and Grippers. For more details on hardware parts of robot consult 0X Delta 4 Wheel Drive Hardware manual.

Figure 1.1: 0X Delta 4 Wheel Drive Robot

2 0X Delta 4 Wheel Drive Robot Communication

Robot supports two different modes of communication, viz. USB (Serial) and 2.4GHz (wireless) module.

2.1 Serial Communication

USB serial communication is preferred for on board embedded PC or on board computer/ laptop. To establish a communication between robot and on board PC, connect USB cable of robot marked as “USB COMMUNICATION” to on board computer.

If 0X Delta robot is equipped with USB extension hub then user can also connect USB cable of robot to on board computer via USB extension hub.

Figure 2.1: USB cable of 0X Delta Robot

After connecting USB cable to on board PC/laptop, you may need to install USB drivers. Windows drivers for USB is available on documentation CD in /Softwares and Drivers folder.

⚠ If robot does not include on board computer then user can also connect “USB COMMUNICATION” cable to off board PC/Laptop.

2.2 Wireless Communication

⚠ To control 0X Delta robot over 2.4GHz wireless communication, make sure that "REMOTE CONTROL OVERRIDE" switch of robot is in release (OFF) state. This switch is illuminated push to ON/OFF switch. When switch is in OFF state, red LED of switch will remain OFF.

Figure 2.2: 0X Delta wireless mode selection switches

0X Delta robot is integrated with 2.4GHz wireless module with unique ID. While working on off-board computers/laptops, user can refer this communication mode. To communicates with robot using wireless module, you will need another wireless module with same unique ID that can be plugged into USB port of off-board PC/Laptop. Refer figure 2.3 for more details.

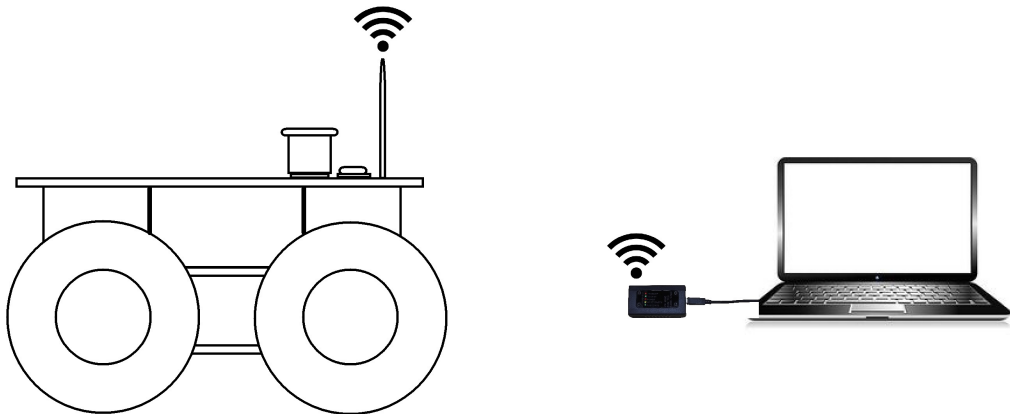


Figure 2.3: Wireless connection topology

After connecting 2.4GHz wireless module to USB port of computer, you will need to install USB drivers. Windows drivers for USB is available on documentation CD in /Softwares and Drivers folder.

Connecting wireless module to USB port of computer will create a COM port on that computer and communication between robot and computer will take place through that com port. As shown in figure 2.4, the wireless settings are printed on 2.4 GHz wireless module.

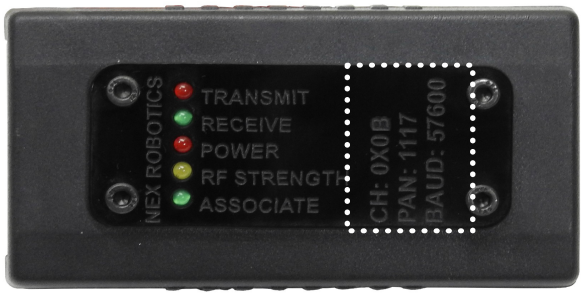


Figure 2.4: 2.4 GHz wireless module

⚠ To ensure successful communication, Always make sure that wireless setting information of 2.4 GHZ wireless module connected to USB port of PC/laptop should match with information of 0X Delta robot.

⚠ Information related to wireless module communication settings for each robot will be different.

⚠ Robot has separate communication channels for both USB and 2.4 GHz wireless module. However user can not access both communication modes simultaneously.

⚠ As a optional accessory, robot can be controlled over manual remote control. For more details please refer section 2.3.

2.3 Manual Remote Control Communication (Optional accessory)

⚠ 0X Delta robot shares same wireless link for 2.4GHz wireless communication mode and for manual remote control communication. Therefore at a given time, only one mode of wireless communication will be active, which is depend on ON/OFF status of "REMOTE CONTROL OVERRIDE" switch.

0X Delta robot can control its motions via manual remote control. On the top side, robot will have two illuminated switches named as "REMOTE CONTROL OVERRIDE" and "REMOTE CONTROL CALIBRATE". Please refer figure 2.2 for more details. Remote control override is illuminated push to ON/OFF switch. This switch takes care of manual remote control feature of robot.

Remote control override switch is press to ON and press to OFF switch, which will override 2.4GHz wireless communication (section 2.2) with manual remote control communication. Basically switching ON remote control override switch will transfer control of robot from 2.4GHz communication to wireless manual remote control communication. In this mode of communication, user can control all the motions of robot via multi channel remote control transmitter. Refer figure 2.5 for multi channel remote control transmitter.

Please go through following steps to switch ON the manual remote control mode.

Step 1: Press "REMOTE CONTROL OVERRIDE" switch.(Assuming that the robot is already in power up state)

Step 2: Now, power up remote control transmitter. On powering remote control transmitter, ideally robot should give following indications.

- Red LED of "REMOTE CONTROL OVERRIDE" switch will lit.
- Red LED of "REMOTE CONTROL CALIBRATE" will start blinking with approximate delay of 1 second.
- Red tower light of robot will start blinking with approximate delay of 1 seconds.
- Robot buzzer will beep after interval of every 4 seconds.

All above indications ensures that the robot controls are switched from 2.4GHz communication to remote control communication. To control robot motions via multi channel remote control, user must get all above indications from robot.

While switching from 2.4GHz communication mode to manual remote control mode, user may encounter with a problem that red LED of "REMOTE CONTROL OVERRIDE" switch is ON but LED of "REMOTE CONTROL CALIBRATION" switch remains in OFF state. This condition will occur when multi channel remote control transmitter is in OFF state. You need to turn it ON by sliding power switch of remote control in upward direction. Refer figure 2.5.

⚠ **To retain 2.4 GHz wireless communication, user will simply have to press and turn OFF the "REMOTE CONTROL OVERRIDE" switch.**

⚠ **Manual remote control mode will only control different motions of robot via remote control transmitter.**

⚠ **To avoid loss of communication between remote control transmitter and robot, please make sure that robot and remote control transmitter are in line of sight with each other.**

⚠ **If robot does not have manual remote control mode feature then "REMOTE CONTROL OVERRIDE" switch and "REMOTE CONTROL CALIBRATE" switches will be absent from robot.**

Controlling Fire Bird V robot via remote control transmitter

In standard package, 6 channel remote control transmitter will be supplied with robot.

Figure 2.5: Remote control configuration

Important: Do not touch the left switch and right switch, as these switches must be in down side for correct working of above remote.

Please refer table 2.1 to know functionality of each channel.

Sr. No.	Channel No	Functionality	
1	**Channel 1**	**Controls left and right motion of robot.**	
		Throttle left position = Left motion	Throttle right position = Right motion
2	**Channel 2**	**Controls forward and backward motion of robot**	
		Throttle up position = Forward motion	Throttle down position = Backward motion
3	**Channel 3**	**Controls velocity of robot.** Throttle down to up corresponds to 0 to max velocity.	
		Throttle extreme up position = Max velocity	Throttle extreme down position = Zero velocity
4	**Channel 4**	**Controls curved motions of robot.**	

Table 2.1

3 0X Delta 4 Wheel Drive Communication Protocol

The robot communicates with host PC or embedded kit using serial or wireless interface. communication settings are as follows:

Baud Rate: 57600
Data Bits: 8
Parity: None
Stop Bits: None

The communication between the host computer and 0X Delta 4 wheel drive robot is performed by using designated commands. The host computer acts as a master and 0X Delta robot acts as a slave. The communication is always initiated by host device. The host device should wait for receiving a response for sent command before sending next command. The communication protocol has a simple request and response structure. The Host PC requests a particular operation to which the target module responds accordingly. The length of each request may vary depending on the task as shown in the table below.

Table 3.1 briefly explains the command packet sent from host to robot. First three byte of each command are always “NEX” which are then followed by the command type and the appropriate command data bytes and checksum for error detection.

Command: Host to 0X Delta

Header			Command	Sub-Command/Data	Checksum
‘N’	‘E’	‘X’	1byte	1byte	1byte

Table 3.1: 0X Delta command packet

In response to the command, the robot will start processing a task, and it will send a reply. The first byte of each reply is either “S” or “F” followed by the command type and corresponding response data as shown in the table below. Here the command type is specified in response to help host identify the response for particular command sent. The first byte of response indicates whether robot executed command successfully. Here 'S' represents Successful execution and 'F' represents failure in execution of given command. Table 3.2 explains the response packet received by host from 0X Delta robot.

Response: 0X Delta to Host

Header	Command	Data	Checksum
‘S’ or ‘F’	1byte	n bytes	1 byte

Table 3.2: 0X Delta response packet

Where Header,
S = *Success, represents successful completion of command*
F = *Failure, indicates command could not be executed by robot*

Algorithm to create and verify checksum byte

Creating checksum byte for command packet string

Procedure:

1. Refer Table 3.1 and add all bytes till last sub-command byte.
2. Convert addition result into hexadecimal value.
3. If addition is greater than or equal to 0xFF then you should choose only last 8 bits of hexadecimal. For e.g if addition result is “0xABCD” then pick only “0xCD” as a addition result.
4. Take 1's complement of addition result and add 1 to it.

Header Byte1	+	**Header Byte2**	+	**Header Byte3**	+	**Command byte**	+	**Sub-command byte1**	+	**Sub-command byte2**	…	**Sub-command byte n**

Example to calculate checksum byte:

Header			**Command**	**Sub-Command/Data0**	**Sub-Command/Data1**	**Checksum**
‘N’	‘E’	‘X’	0x70	0x01	0x62	?

Header byte 1 ('N') = $(78)_{10} = (4E)_{16}$
Header byte 2 ('E') = $(69)_{10} = (45)_{16}$
Header byte 3 ('X') = $(88)_{10} = (58)_{16}$
Command byte = $(112)_{10} = (70)_{16}$
Sub-command byte 1 = $(1)_{10} = (01)_{16}$
Sub-command byte 2 = $(98)_{10} = (62)_{16}$

1. Adding all bytes of command string

Header			**Command**	**Sub-Command/Data0**	**Sub-Command/Data1**	**Total**
78	69	88	112	1	98	446

Addition result = $(446)_{10} = (01BE)_{16}$

Choosing last 8 bytes of addition result = $(BE)_{16}$

2. Taking 1's complement of addition result

Addition result	**0xBE**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**0**
1's complement	**0x41**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**

1's complement result = $(41)_{16}$

3. To calculate checksum add 1 to 1's complement's result

1's complement	0x41	0	1	0	0	0	0	0	1
	+								1
Checksum byte	0x42	0	1	0	0	0	0	1	0

Checksum byte = $(66)_{10} = (42)_{16}$
Therefore command packet string with checksum will be as follows,

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x70	0x01	0x62	**0x42**

Verifying checksum byte of response packet string

Procedure:

1. Refer Table 3.2 and add all bytes till 2nd last byte received from robot. (Do not add last byte i.e checksum byte as we have to verify this byte.)
2. Convert addition result into hexadecimal value.
3. If addition is greater than or equal to 0xFF then you should choose only last 8 bits of hexadecimal. For e.g if addition result is "0xABCD" then pick only "0xCD" as a addition result.
4. Take 1's complement of addition result and add 1 to it.
5. Now after adding 1, verify whether byte created in step 4 is equal to checksum byte which was received in response packet string from robot. If it is equal that means response packet string received from robot is correct.

Header Byte	+	Command byte	+	Data byte 1	+	Data byte2	….	Data byte n

Example to verify checksum byte:
Consider following example response string to read battery voltage of robot

Header	Command	Data	Checksum
'S'	0x20	0x78	0x15

Header byte 1 ('S') = $(83)_{10} = (53)_{16}$
Command byte = $(32)_{10} = (20)_{16}$
Data byte = $(120)_{10} = (78)_{16}$
Checksum byte = $(21)_{10} = (15)_{16}$ (Byte to be verify)

1. Adding all bytes of following response string except checksum byte.

Header	Command	Data	Checksum	Total (without checksum byte)
83	32	120	21	235

Addition result = $(235)_{10} = (EB)_{16}$

2. Taking 1's complement of addition result.

Addition result	0xEB	1	1	1	0	1	0	1	1
1's complement	0x14	0	0	0	1	0	1	0	0

1's complement result = $(24)_{16}$

3. To calculate checksum add 1 to 1's complement's result.

1's complement	0x14	0	0	0	1	0	1	0	0
	+								1
Checksum byte	0x15	0	0	0	1	0	1	0	1

Calculated checksum byte after step 3 = $(15)_{16}$

Now if calculated checksum byte = checksum byte of response string, then received byte is correct.

3.1 Sensor Module Commands

3.1.1 Get 8 bytes of ultrasonic sensor data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x05	0x01	1 byte

Response: 0X Delta to Host
Length: 11 bytes

Header	Command	Data								Checksum
'S' or 'F'	0x05	UL0	UL1	UL2	UL3	UL4	UL5	UL6	UL7	1 byte

3.1.2 Get single ultrasonic sensor data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x01	n=0 to 7	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x01	1 byte	1 byte

Here n=0 represents id of ultrasonic sensor 0

Note: Based on users requirements ultrasonic sensors integrated on robot may vary from robot to robot. To identify the type of ultrasonic sensor please refer "Datasheets\ MaxBotix Ultrasonic Range Sensors" folder from documentation CD.

Calculations for different ultrasonic sensor range sensors,

Sr. No	Ultrasonic sensor type	Conversion of raw sensor data to measured distance
1.	LV-MAXSonar-EZ4	Ultrasonic data * 5 = Distance in cm
2.	HRXL-MaxSonar-WRT (MB7380)	Ultrasonic data * 20 = Distance in mm
3.	HRXL-MaxSonar-WRLS (MB7363)	Ultrasonic data * 40 = Distance in mm

3.1.3 Get 8 bytes of line sensor data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x05	0x04	1 byte

Response: 0X Delta to Host
Length: 11 bytes

Header	Command	Data								Checksum
'S' or 'F'	0x05	WL0	WL1	WL2	WL3	WL4	WL5	WL6	WL7	1 byte

3.1.4 Get single white line sensor data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x04	n=0 to 7	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x04	1 byte	1 byte

Here n=0 represents id of white line sensor 0

3.1.5 Get 8 bytes of IR proximity data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x05	0x02	1 byte

Response: 0X Delta to Host
Length: 11 bytes

Header	Command	Data								Checksum
'S' or 'F'	0x05	IR0	IR1	IR2	IR3	IR4	IR5	IR6	IR7	1 byte

3.1.6 Get single IR proximity sensor data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x02	n=0 to 7	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x02	1 byte	1 byte

Here n=0 represents id of IR proximity sensor 0

3.1.7 Get 8 bytes of IR distance sensor data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x05	0x03	1 byte

Response: 0X Delta to Host
Length: 11 bytes

Header	Command	Data								Check sum
'S' or 'F'	0x05	Range 0	Range 1	Range 2	Range 3	Range 4	Range 5	Range 6	Range 7	1 byte

3.1.8 Get single IR distance sensor data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x03	n=0 to 7	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x03	1 byte	1 byte

Here n=0 represents id of IR distance sensor 0

Data received from IR distance sensors will be a raw data, To convert this into actual distance measured by sensor in cm refer datasheet of 10 – 80 cm IR distance sensor. (refer datasheet of GP2Y0A21YK0F).

3.1.9 Turn ON IR proximity sensor

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x09	0x01	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x09	1 byte

3.1.10 Turn OFF IR proximity sensor

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x09	0x02	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x09	1 byte

3.1.11 Turn ON white line sensor

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x09	0x03	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x09	1 byte

3.1.12 Turn OFF white line sensor

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x09	0x04	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x09	1 byte

3.1.13 Turn ON ultrasonic sensor ranging

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x09	0x05	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x09	1 byte

3.1.14 Turn OFF ultrasonic sensor ranging

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x09	0x06	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x09	1 byte

3.1.15 Turn ON servo pod ultrasonic sensor ranging

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x09	0x07	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x09	1 byte

3.1.16 Turn OFF servo pod ultrasonic sensor ranging

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x09	0x08	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x09	1 byte

3.1.17 Turn ON IR distance sensor ranging

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x09	0x09	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x09	1 byte

3.1.18 Turn OFF IR distance sensor ranging

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x09	0x0A	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x09	1 byte

3.1.19 Get IR proximity sensor ON/OFF status

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x0A	0x01	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x0A	0x00 or 0x01	1 byte

0x00 is OFF and 0x01 is ON

3.1.20 Get white line sensor ON/OFF status

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x0A	0x02	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x0A	0x00 or 0x01	1 byte

0x00 is OFF and 0x01 is ON

3.1.21 Get ultrasonic sensor ranging ON/OFF status

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x0A	0x03	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x0A	0x00 or 0x01	1 byte

0x00 is OFF and 0x01 is ON

3.1.22 Get servo pod ultrasonic sensor ranging ON/OFF status

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x0A	0x04	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x0A	0x00 or 0x01	1 byte

0x00 is OFF and 0x01 is ON

3.1.23 Get IR distance sensor ranging status

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x0A	0x05	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x0A	0x00 or 0x01	1 byte

0x00 is OFF and 0x01 is ON

3.2 Power Management Module Commands

3.2.1 Read battery voltage

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x20	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x20	Battery Voltage	1 byte

3.2.2 Read battery current

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x21	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x21	Battery Current	1 byte

3.2.3 Read battery temperature

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x22	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x22	Battery Temperature	1 byte

3.2.4 Read Battery voltage, current & temperature

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x23	0x00	1 byte

Response: 0X Delta to Host
Length: 6 bytes

Header	Command	Data0	Data1	Data2	Checksum
'S' or 'F'	0x23	Battery Voltage	Battery Current	Battery Temperature	1 byte

Battery voltage calculation:

Battery voltage = (Data * 0.14235) + 0.35

e.g. if Data = 95 dec or 0x5F,
Then Battery Voltage = (95 * 0.14235) + 0.35 = 13.87 Volts

Battery current calculation:

Battery Current = ((2.5 - (Data * 0.0129))) / 0.185

e.g. if Data = 170 dec or 0xAA,
Then Battery Current = ((2.5 - (170 * 0.0129))) /0.185 = 1.6 Amps

Battery temperature calculation:

Battery Temperature = (Data * 1.29)

e.g. if Data = 20,
Then Battery Temperature = (20 * 1.29) = 25.8 deg. C

3.3 Motion Control Commands

3.3.1 Set safety timeout

This command will set timeout period in terms of seconds to stop the robot motion. That means as long as robot is getting commands from user within this timeout period, robot will keep continuing it's current motion otherwise it will simply stop.

Command: Host to 0X Delta
Length: 7 bytes

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x7A	0x01	Byte	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x7A	1 byte

Where "Byte" in command string will contain safety timeout period for robot in seconds.
For e.g. To set timeout = 10 sec,
$(10)_{10} = (0A)_{16}$

Command string to set above timeout should be sent as follows,

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x7A	0x01	0x0A	1 byte

Now, consider an e.g. If safety timeout period set by robot is 10 seconds and if robot is moving forward direction then in order to keep it in that motion user will need to send any complete legitimate command mentioned in this manual within every 10 seconds, otherwise robot will simply stop it's motion.

3.3.2 Get safety timeout

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x7A	0x02	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x7A	Timeout byte	1 byte

3.3.3 Set safety ON / OFF

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x89	Safety	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x89	1 byte

Safety = 1 → Safety ON
Safety = 0 → Safety OFF

⚠ **Behavior during robot safety ON feature**
1. Max velocity of robot limited to a safe value. This safe max velocity for 0X Delta robot is 0.4 m/Sec.

⚠ **Behavior during robot safety OFF feature**
1. Max velocity of robot can be altered up to robot's maximum reachable velocity

3.3.4 Set mode

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x90	Mode	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x90	1 byte

Mode 0: Open loop velocity control
This is the simplest mode of operation. In this mode, amount of power to be given to the motor can be specified. This is open loop control mode. In this mode, motor has the quickest response but there is no closed loop velocity control and the actual velocity may vary depending on output torque required for movement.

Mode 1: Closed loop velocity control
This is closed loop velocity control mode. In this mode, motor controller will try to increase the power to the motor until motor achieves the specified velocity or it reaches its maximum output power. The motor will maintain its velocity even while climbing and going down the slope.

Mode 2: Position control mode
In this mode, the motor can be controlled to move by a specified number of counts. The distance is specified in terms of encoder counts for each motor. The maximum velocity and acceleration for reaching the desired position can be set. The direction of motion is always decided by the velocity. The number of encoder counts to travel is always taken as absolute value of signed target count specified. The encoder count registers for both the motors are reset and the position counting always starts from zero.

Example:
To move both motors by a count of 0x1017 with velocity 0x50, send

Header	Command	Position 1	Velocity 1	Position 2	Velocity 2
0x4E 0x45 0x58	0x40	0x00 0x00 0x10 0x17	0x50	0x00 0x00 0x10 0x17	0x50

This will reset the 32 bit encoder count registers and motors will start moving till the target encoder count is achieved. The direction of motion is decided by the velocity specified.

⚠ **In position control mode, every new position command will reset the encoder count registers of both the motors and the target encoder count will be achieved by starting from zero.**

3.3.5 Get mode

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x91	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x91	Mode	1 byte

3.3.6 Set acceleration

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x98	Acceleration	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x98	1 byte

Note: Acceleration count (n) must be in the range of 1-16.
Acceleration = (168.72 * n) mm/Sec squared

3.3.7 Get acceleration

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x99	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x99	Acceleration	1 byte

3.3.8 Set left motor velocity

Command: Host to 0X Delta
Length: 7 bytes

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x95	Velocity (MSB)	Velocity (LSB)	1 byte

Note: Velocity = 512-Full forward,
= 0-Stop,
= -512-Full reverse

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x95	1 byte

3.3.9 Get left motor velocity

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x9A	0x00	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 1	Checksum
'S' or 'F'	0x9A	Left Velocity (MSB)	Left Velocity (LSB)	1 byte

⚠ Robot direction command will act exactly opposite if motor velocity is set to -ve value. e.g if velocity set for motor is -ve, then sending forward direction command will move the motor in reverse direction.

⚠ Any change in velocity of robot will reflect only and only after resending robot direction command.

3.3.10 Set right motor velocity

Command: Host to 0X Delta
Length: 7 bytes

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x96	Velocity (MSB)	Velocity (LSB)	1 byte

Note: Velocity = 512-Full forward,
= 0-Stop,
= -512-Full reverse

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x96	1 byte

3.3.11 Get right motor velocity

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x9B	0x00	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 1	Checksum
'S' or 'F'	0x9B	Right Velocity (MSB)	Right Velocity (LSB)	1 byte

⚠ Robot direction command will act exactly opposite if motor velocity is set to -ve value. e.g if velocity set for motor is -ve, then sending forward direction command will move the motor in reverse direction.

⚠ Any change in velocity of robot will reflect only and only after resending robot direction command.

3.3.12 Set left motor velocity in meters per seconds

Command: Host to 0X Delta
Length: 7 bytes

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x70	Velocity (MSB)	Velocity (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x70	1 byte

Example to set forward linear velocity in meters per second:

If forward velocity = 0.354 meters/sec. Then,
$0.354 * 1000 = 354$ mm/sec. Now, $(354)_{10} = (0162)_{16}$

To set left motor velocity, command from Host to 0X Delta should be sent as follows:

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x70	0x01	0x62	1 byte

Example to set reverse linear velocity in meters per second:

If reverse velocity = -0.354 meters/sec. Then,
-0.354 * 1000 = -354 mm/sec. Now, $(-354)_{10} = (FE9E)_{16}$

To set left motor velocity, command from Host to 0X Delta should be sent as follows:

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x70	0xFE	0x9E	1 byte

3.3.13 Get left motor velocity in mm per seconds

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x76	0x00	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 1	Checksum
'S' or 'F'	0x76	Left Velocity (MSB)	Left Velocity (LSB)	1 byte

Left motor linear velocity calculations:

This command will receive 2 byte data which contains present left motor velocity in mm/ sec. To convert this velocity into meters/sec,

Left velocity = Data received / 1000

e.g. if Data = 0x0154,
$(0154)_{16} = (340)_{10}$
Then motor velocity = 340 / 1000 = 0.340 meters/sec (Forward velocity)

If Data = 0xFEAC,
$(FEAC)_{16} = (-340)_{10}$
Then motor velocity = -340 / 1000 = -0.340 meters/sec (Reverse velocity)

⚠ Robot direction command will act exactly opposite if motor velocity is set to -ve value. e.g if velocity set for motor is -ve, then forward direction command will move the motor in reverse direction.

⚠ Any change in velocity of robot will reflect only and only after resending robot direction command.

3.3.14 Set right motor velocity in meters per seconds

Command: Host to 0X Delta
Length: 7 bytes

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x71	Velocity (MSB)	Velocity (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x71	1 byte

Example to set forward linear velocity in meters per second:

If forward velocity = 0.354 meters/sec. Then,
$0.354 * 1000 = 354$ mm/sec. Now, $(354)_{10} = (0162)_{16}$

To set right motor velocity, command from Host to 0X Delta should be sent as follows:

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x71	0x01	0x62	1 byte

Example to set reverse linear velocity in meters per second:

If reverse velocity = -0.354 meters/sec. Then,
$-0.354 * 1000 = -354$ mm/sec. Now, $(-354)_{10} = (FE9E)_{16}$

To set right motor velocity, command from Host to 0X Delta should be sent as follows:

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x71	0xFE	0x9E	1 byte

3.3.15 Get right motor velocity in mm per seconds

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x77	0x00	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 1	Checksum
'S' or 'F'	0x77	Right Velocity (MSB)	Right Velocity (LSB)	1 byte

Right motor linear velocity calculations:

This command will receive 2 byte data which contains present right motor velocity in mm/sec. To convert this velocity into meters/sec,

Right velocity = Data received / 1000

e.g. if Data = 0x0154,
$(0154)_{16} = (340)_{10}$
Then motor velocity = 340 / 1000 = 0.340 meters/sec (Forward velocity)

If Data = 0xFEAC,
$(FEAC)_{16} = (-340)_{10}$
Then motor velocity = -340 / 1000 = -0.340 meters/sec (Reverse velocity)

⚠ Robot direction command will act exactly opposite if motor velocity is set to -ve value. e.g if velocity set for motor is -ve, then forward direction command will move the motor in reverse direction.

⚠ Any change in velocity of robot will reflect only and only after resending robot direction command.

3.3.16 Set left motor velocity in radians per seconds

Command: Host to 0X Delta
Length: 7 bytes

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x7B	Velocity (MSB)	Velocity (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x7B	1 byte

Example to set anti clockwise angular velocity of left motor in radians per second:

If anti clockwise velocity = 1.57 radians/sec. Then,
1.57 * 1000 = 1570 Now, $(1570)_{10} = (0622)_{16}$
To set left motor velocity, command from Host to 0X Delta should be sent as follows:

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x7B	0x06	0x22	1 byte

Example to set clockwise angular velocity of left motor in radians per second:

If reverse velocity = -1.57 radians/sec. Then,
-1.57 * 1000 = -1570 Now, $(-1570)_{10} = (F9DE)_{16}$
To set left motor velocity, command from Host to 0X Delta should be sent as follows:

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x7B	0xF9	0xDE	1 byte

3.3.17 Get left motor velocity in radians per seconds

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x7D	0x00	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 1	Checksum
'S' or 'F'	0x7D	Left Velocity (MSB)	Left Velocity (LSB)	1 byte

Left motor angular velocity calculations:

This command will receive 2 byte data which contains present left motor velocity. To convert this velocity into radians/sec,

Left velocity = Data Received / 1000

e.g. if Data = 0x08DE,
$(08DE)_{16} = (2270)_{10}$
Left motor anti clockwise velocity = 2270 / 1000 = 2.270 radians/sec

If Data = 0xF722,
$(F722)_{16} = (-2270)_{10}$
Left motor clockwise velocity = -2270 / 1000 = -2.270 radians/sec

⚠ Robot direction command will act exactly opposite if motor velocity is set to -ve value. e.g if velocity set for motor is -ve, then forward direction command will move the motor in reverse direction.

⚠ Any change in velocity of robot will reflect only and only after resending robot direction command.

3.3.18 Set right motor velocity in radians per seconds

Command: Host to 0X Delta
Length: 7 bytes

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x7C	Velocity (MSB)	Velocity (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x7C	1 byte

Example to set anti clockwise angular velocity of right motor in radians per second:

If anti clockwise velocity = 1.57 radians/sec. Then,
$1.57 * 1000 = 1570$ Now, $(1570)_{10} = (0622)_{16}$
To set right motor velocity, command from Host to 0X Delta should be sent as follows:

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x7C	0x06	0x22	1 byte

Example to set clockwise angular velocity of right motor in radians per second:

If clockwise velocity = -1.57 radians/sec. Then,
$-1.57 * 1000 = -1570$ Now, $(-1570)_{10} = (F9DE)_{16}$

To set right motor velocity, command from Host to 0X Delta should be sent as follows:

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x7C	0xF9	0xDE	1 byte

3.3.19 Get right motor velocity in radians per seconds

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x7E	0x00	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 1	Checksum
'S' or 'F'	0x7E	Right Velocity (MSB)	Right Velocity (LSB)	1 byte

Right motor angular velocity calculations:

This command will receive 2 byte data which contains present right motor velocity. To convert this velocity into radians/sec,

Right velocity = Data Received / 1000

e.g. if Data = 0x08DE,
$(08DE)_{16} = (2270)_{10}$
Right motor anti clockwise velocity = 2270 / 1000 = 2.270 radians/sec

If Data = 0xF722,
$(F722)_{16} = (-2270)_{10}$
Right motor clockwise velocity = -2270 / 1000 = -2.270 radians/sec

⚠ **Robot direction command will act exactly opposite if motor velocity is set to -ve value. e.g if velocity set for motor is -ve, then forward direction command will move the motor in reverse direction.**

⚠ **Any change in velocity of robot will reflect only and only after resending robot direction command.**

3.3.20 Set robot angular velocity in radians per seconds

Command: Host to 0X Delta
Length: 7 bytes

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x74	Velocity (MSB)	Velocity (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x74	1 byte

Example to set anti clockwise angular velocity of robot in radians per second:

If anti clockwise velocity = 1.54 radians/sec. Then,
1.54 * 1000 = 1540 Now, $(1540)_{10} = (0604)_{16}$
To set robot anti clockwise angular velocity, command from Host to 0X Delta should be sent as follows:

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x74	0x06	0x04	1 byte

Example to set clockwise angular velocity of robot in radians per second:

If anti clockwise velocity = -1.54 radians/sec. Then,
$-1.54 * 1000 = -1540$ Now, $(-1540)_{10} = (F9FC)_{16}$

To set robot clockwise angular velocity, command from Host to 0X Delta should be sent as follows:

Header			Command	Sub-Command/Data0	Sub-Command/Data1	Checksum
'N'	'E'	'X'	0x74	0xF9	0xFC	1 byte

⚠ Robot direction command will act exactly opposite if robot velocity is set to -ve value. e.g if velocity set for robot is -ve, then left direction command will move the robot in right direction.

3.3.21 Get robot angular velocity in radians per seconds

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x78	0x00	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 1	Checksum
'S' or 'F'	0x78	Velocity (MSB)	Velocity (LSB)	1 byte

Robot angular velocity calculations:
This command will receive 2 byte data which contains present robot angular velocity. To convert this velocity into radians/sec,

velocity = Data Received / 1000

e.g. if Data = 0x08DE,
$(08DE)_{16} = (2270)_{10}$
Robot anti clockwise velocity = 2270 / 1000 = 2.270 radians/sec

If Data = 0xF722,
$(F722)_{16} = (-2270)_{10}$
Robot clockwise velocity = -2270 / 1000 = -2.270 radians/sec

3.3.22 Set robot direction

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x94	Direction	1 byte

Direction:

0x01 is Forward
0x02 is Reverse
0x03 is Left
0x04 is Right
0x06 is Stop

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x94	1 byte

⚠ Robot direction command will act exactly opposite if motor velocity is set to -ve value. e.g if velocity set for motor is -ve, then forward direction command will move the motor in reverser direction.

For more details please refer following table,

Left Motor Velocity	Right Motor Velocity	Command requested	Direction followed by robot
e.g. Left velocity = 0.125m/s & Right velocity = 0.212 m/s or Left velocity = 1.52 rad/s & Right velocity = 1.75 rad/s		0x01	Forward
		0x02	Backward
		0x03	Left
		0x04	Right
e.g. Left velocity = -0.125m/s & Right velocity = -0.212 m/s or Left velocity = -1.52 rad/s & Right velocity = -1.75 rad/s		0x01	Backward
		0x02	Forward
		0x03	Right
		0x04	Left
e.g. Left velocity = 0.125m/s & Right velocity = -0.212 m/s or Left velocity = 1.52 rad/s & Right velocity = -1.75 rad/s		0x01	Right
		0x02	Left
		0x03	Backward
		0x04	Forward
e.g. Left velocity = -0.125m/s & Right velocity = 0.212 m/s or Left velocity = -1.52 rad/s & Right velocity = 1.75 rad/s		0x01	Left
		0x02	Right
		0x03	Forward
		0x04	Backward

Table 3.3: Robot Directions for different velocity patterns

3.3.23 Get left motor encoder counts

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x92	0x00	1 byte

Response: 0X Delta to Host
Length: 7 bytes

Header	Command	Data0	Data1	Data2	Data3	Checksum
'S' or 'F'	0x92	Byte 1 (MSB)	Byte2	Byte3	Byte 4 (LSB)	1 byte

3.3.24 Get right motor encoder counts

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x93	0x00	1 byte

Response: 0X Delta to Host
Length: 7 bytes

Header	Command	Data0	Data1	Data2	Data3	Checksum
'S' or 'F'	0x93	Byte 1 (MSB)	Byte2	Byte3	Byte 4 (LSB)	1 byte

3.3.25 Clear encoder counts

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x8C	0x00	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x8C	1 byte

3.3.26 Set position

This command will move the robot to specific position which is mentioned in terms of encoder counts with specified velocity for both motors.

⚠ For desired behavior of robot, before sending following command user should select "position control mode".

Command: Host to 0X Delta
Length: 17 bytes

Header			Command	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5
'N'	'E'	'X'	0x9C	0x00 (MSB)	0x00	0x03	0xE8 (LSB)	0x00 (MSB)	0x50 (LSB)

Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Checksum
0x00 (MSB)	0x00	0x03	0xE8 (LSB)	0x00 (MSB)	0x50 (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x9C	1 byte

Here,
Data 0 to Data 3 = 32 bit Left encoder counts
Data 4 to Data 5 = 16 bit Left motor velocity
Data 6 to Data 9 = 32 bit Right encoder counts
Data 10 to Data 11 = 16 bit Right motor velocity

After sending above command string to robot, The encoder count registers for both the motors will get reset and the position counting will always starts from zero and motors will start moving till encoder counts becomes equal to counts mentioned in Data 0 to 3 and Data 6 to 9. The direction of motion is decided by the velocity specified.

Velocity = 512 - Full forward,
= 0 - Stop,
= -512 - Full reverse

⚠ Robot direction command will act exactly opposite if motor velocity is set to -ve value. e.g if velocity set for particular motor is -ve, then forward direction command will move the that motor in reverser direction.

⚠ Every new Set Position command will reset the encoder count registers of both the motors and the target encoder count will be achieved by starting from zero.

3.3.27 Set linear position

This command will set linear position of robot with distance to be traveled by both motor of robot in terms of meters and velocity of both motors in terms of meters per second.

⚠ For desired behavior of robot, before sending following command user should select "position control mode".

Command: Host to 0X Delta
Length: 17 bytes

Header			Command	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5
'N'	'E'	'X'	0x72	Left Distance Byte1 (MSB)	Left Distance Byte2	Left Distance Byte 3	Left Distance Byte 4 (LSB)	Left Velocity Byte 1 (MSB)	Left Velocity Byte 2 (LSB)

Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Checksum
Right Distance Byte1 (MSB)	Right Distance Byte 2	Right Distance Byte 3	Right Distance Byte 4 (LSB)	Right Velocity Byte 1 (MSB)	Right Velocity Byte 2 (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x72	1 byte

Calculations to set linear position of 0X Delta robot:
e.g. Set position of robot to,
Left motor distance = 2.5 meters at Left motor velocity = 0.265 m/Sec and
Right motor distance = 1.5 meters at right motor velocity = -0.152 m/Sec

Left motor distance = 2.5 * 1000 = 2500 mm
Right motor distance = 1.5 * 1000 = 1500 mm
Left motor velocity = 0.265 * 1000 = 265 mm/Sec
Right motor velocity = -0.152 * 1000 = -152 mm/Sec
$(2500)_{10} = (09C4)_{16}$
$(265)_{10} = (0109)_{16}$
$(1500)_{10} = (05DC)_{16}$
$(-152)_{10} = (FF68)_{16}$

command from Host to 0X Delta should be sent as follows:

Header			Command	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5
'N'	'E'	'X'	0x72	0x00	0x00	0x09	0xC4	0x01	0x09

Data 6	Data 7	Data 8	Data 9	Data 10	Data 11	Checksum
0x00	0x00	0x05	0xDC	0xFF	0x68	1 byte

Above command string will move left wheel of robot to 2.5 meters in forward direction at forward velocity equal to 0.265 m/Sec and right wheel of robot to 1.5 meters in backward direction at backward velocity equal to -0.152 m/Sec.

⚠ Every new Set linear position command will reset the encoder count registers of both the motors and the target encoder count will be achieved by starting from zero.

3.3.28 Set angular position

This command will set angular position of robot with distance to be rotated by robot in terms of radians and velocity of both motors in terms of radians per second.

⚠ For desired behavior of robot, before sending following command user should select "position control mode".

Command: Host to 0X Delta
Length: 11 bytes

Header			Cmd	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Checksum
'N'	'E'	'X'	0x75	Angle Byte1 (MSB)	Angle Byte 2	Angle Byte 3	Angle Byte 4 (LSB)	Vel. Byte 1 (MSB)	Vel. Byte 2 (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x75	1 byte

Example to set clockwise angular position of 0X Delta Robot:

e.g. Set position of robot to, Angle = 180.5 degrees at Velocity = 2.45 radians/Sec
Velocity = 2.45 * 1000 = 2450
Angle = 180.5 * 1000 = 180500

$(180500)_{10} = (0002C114)_{16}$
$(2450)_{10} = (0992)_{16}$

command from Host to 0X Delta should be sent as follows:

Header			Cmd	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Checksum
'N'	'E'	'X'	0x75	0x00	0x02	0xC1	0x14	0x09	0x92	1 byte

Above command string will rotate the robot in clockwise direction.

Example to set anti clockwise angular position of 0X Delta Robot:
e.g. Set position of robot to, Angle = -180.5 degrees at Velocity = - 2.45 radians/Sec
Velocity = -2.45 * 1000 = -2450
Angle = 180.5 * 1000 = 180500

$(180500)_{10} = (02C114)_{16}$
$(-2450)_{10} = (F66E)_{16}$

command from Host to 0X Delta should be sent as follows:

Header			Cmd	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Checksum
'N'	'E'	'X'	0x75	0x02	0xC1	0x14	0x09	0xF6	0x6E	1 byte

Above command string will rotate the robot in anticlockwise direction.

⚠ **Every new Set angular position command will reset encoder count registers of both the motors and the target encoder count will be achieved by starting from zero.**

3.3.29 Get left motor voltage

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x97	0x03	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x97	V_Left	1 byte

3.3.30 Get right motor voltage

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x97	0x04	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x97	V_Right	1 byte

3.3.31 Get left motor current

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x97	0x01	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x97	I_Left	1 byte

I_Left provides 10 times number of amps i.e. value 25 decimal indicates 2.5 Amps

3.3.32 Get right motor current

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x97	0x02	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x97	I_Right	1 byte

I_Right provides 10 times number of amps i.e. value 25 decimal indicates 2.5 Amps

3.3.33 Set wheel diameter in micron

Command: Host to 0X Delta
Length: 10 bytes

Header			command	Sub-Command/ Data	Data 0	Data 1	Data 2	Data 3	Checksum
'N'	'E'	'X'	0x79	0x01	Byte1 (MSB)	Byte 2	Byte 3	Byte 4 (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x79	1 byte

Example to set wheel diameter of 0X Delta Robot:
e.g. Wheel Diameter of robot = 98.5mm
Diameter to be send to robot = 98.5 * 1000 = 98500 micron

$(98500)_{10} = (000180C4)_{16}$

command from Host to 0X Delta should be sent as follows:

Header			cmd	Sub-cmd/ Data	Data 0	Data 1	Data 2	Data 3	Checksum
'N'	'E'	'X'	0x79	0x01	0x00	0x01	0x80	0xC4	1 byte

Above command string will set wheel diameter of robot as 98.5mm

3.3.34 Get wheel diameter in micron

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x79	0x02	1 byte

Response: 0X Delta to Host
Length: 7 bytes

Header	Command	Data0	Data1	Data2	Data3	Checksum
'S' or 'F'	0x79	Byte 1 (MSB)	Byte2	Byte3	Byte 4 (LSB)	1 byte

Robot wheel diameter calculations:
This command will receive 4 byte data which contains previously set wheel diameter of robot in micron. To convert this received bytes into mm,

Wheel diameter = Data Received / 1000

e.g. if Data = 0x0001879A,
$(0001879A)_{16} = (100250)_{10}$
Robot wheel diameter = 100250 / 1000 = 100.250 mm

3.3.35 Set axle length in micron

Command: Host to 0X Delta
Length: 10 bytes

Header			command	Sub-Command/ Data	Data 0	Data 1	Data 2	Data 3	Checksum
'N'	'E'	'X'	0x79	0x03	Byte1 (MSB)	Byte 2	Byte 3	Byte 4 (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x79	1 byte

Example to set axle length of 0X Delta Robot:

e.g. Distance between two wheels of robot = 280.15mm
Distance to be send to robot = 280.15 * 1000 = 280150 micron

$(280150)_{10} = (00044656)_{16}$

command from Host to 0X Delta should be sent as follows:

Header			cmd	Sub-cmd/ Data	Data 0	Data 1	Data 2	Data 3	Checksum
'N'	'E'	'X'	0x79	0x03	0x00	0x04	0x46	0x56	1 byte

Above command string will set axle distance of robot as 280.15mm

3.3.36 Get axle length in micron

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x79	0x04	1 byte

Response: 0X Delta to Host
Length: 7 bytes

Header	Command	Data0	Data1	Data2	Data3	Checksum
'S' or 'F'	0x79	Byte 1 (MSB)	Byte2	Byte3	Byte 4 (LSB)	1 byte

Robot Axle length calculations:

This command will receive 4 byte data which contains previously set axle length of robot in micron. To convert this received bytes into mm,

Axle length = Data Received / 1000

e.g. if Data = 0x000443CC,
$(000443CC)_{16} = (279500)_{10}$
Robot wheel diameter = 279500 / 1000 = 279.500 mm

3.3.37 Set maximum robot velocity in meters per seconds

Command: Host to 0X Delta
Length: 8 bytes

Header			Command	Sub-command/ Data 0	Sub-command/ Data 1	Sub-command/ Data 2	Checksum
'N'	'E'	'X'	0x79	0x05	Byte 1 (MSB)	Byte 2 (LSB)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x79	1 byte

This command will limit maximum reachable velocity of robot during safety OFF. Above command is effective only when safety feature of robot is OFF.

Example to set maximum velocity of motors in meters per second:

If velocity = 0.354 meters/sec. Then,
$0.354 * 1000 = 354$ mm/sec. Now, $(354)_{10} = (0162)_{16}$

To maximum velocity of robot, command from Host to 0X Delta should be sent as follows:

Header			cmd	Sub-cmd/Data 0	Sub-cmd/Data 1	Sub-cmd/Data 2	Checksum
'N'	'E'	'X'	0x79	0x05	0x01	0x62	1 byte

3.3.38 Get maximum robot velocity in mm per seconds

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x79	0x06	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 0	Checksum
'S' or 'F'	0x79	Motor Velocity (MSB)	Motor Velocity (LSB)	1 byte

Robot maximum velocity calculations:

This command will receive 2 byte data which contains present maximum limited velocity of robot in mm/sec. To convert this velocity into meters/sec,

Motor velocity = Data received / 1000

e.g. if Data = 0x0154,
$(0154)_{16} = (340)_{10}$
Then motor velocity = 340 / 1000 = 0.340 meters/sec (Forward velocity)

3.3.39 Get position encoder resolution in counts per wheel revolution

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x79	0x07	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 0	Checksum
'S' or 'F'	0x79	Encoder count (MSB)	Encoder count (LSB)	1 byte

For e.g. if encoder resolution for 0X Delta Robot = 3200, Then response string from robot will be as follows,
$(3200)_{10} = (0C80)_{16}$

Header	Command	Data 0	Data 0	Checksum
'S'	0x79	0x0C	0x80	0xFA

3.4 Inertial Sensors Commands

⚠ Following Inertial measurement unit commands are not designated for 9DOF Razor IMU unit. To get commands for Razor IMU, please refer "0X Delta IMU and GPS Module Interfacing" documentation from documentation CD.

3.4.1 Get all 3 axis data of accelerometer

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x05	0x05	1 byte

Response: 0X Delta to Host
Length: 9 bytes

Header	Command	Data 0		Data 1		Data 2		Checksum
'S' or 'F'	0x05	x-axis LSB	x-axis MSB	y-axis LSB	y-axis MSB	z-axis LSB	z-axis MSB	1 byte

Data 0, 1 and 2 contains 16 bit signed int data.
Convert accelerometer data into g:

Acceleration (g) = (16 bit Raw data) * 0.004 / 16

3.4.2 Get all 3 axis data of gyroscope

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x05	0x06	1 byte

Response: 0X Delta to Host
Length: 9 bytes

Header	Command	Data 0		Data 1		Data 2		Checksum
'S' or 'F'	0x05	x-axis LSB	x-axis MSB	y-axis LSB	y-axis MSB	z-axis LSB	z-axis MSB	1 byte

Data 0, 1 and 2 contains 16 bit signed int data.
Convert gyroscope data into deg/sec:

Rate (deg/sec) = ((16 bit Raw Data) / 57.0) *0.01

3.4.3 Get all 3 axis data of magnetometer

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x05	0x07	1 byte

Response: 0X Delta to Host
Length: 9 bytes

Header	Command	Data 0		Data 1		Data 2		Checksum
'S' or 'F'	0x05	x-axis LSB	x-axis MSB	y-axis LSB	y-axis MSB	z-axis LSB	z-axis MSB	1 byte

Data 0, 1 and 2 contains 16 bit signed int data.
Convert magnetometer data into Gauss:

For X and Y axis
Magnetic Field (Gauss) = (16 bit Raw Data) / 1100.0)

For Z axis
Magnetic Field (Gauss) = (16 bit Raw Data) / 980.0)

3.4.4 Get X axis data of accelerometer

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x10	0x01	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data		Checksum
'S' or 'F'	0x10	LSB	MSB	1 byte

Data contains 16 bit signed int X axis data.
Convert accelerometer data into g:

Acceleration (g) = (16 bit Raw data) * 0.004 / 16

3.4.5 Get Y axis data of accelerometer

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x10	0x02	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data		Checksum
'S' or 'F'	0x10	LSB	MSB	1 byte

Data contains 16 bit signed int Y axis data.
Convert accelerometer data into g:

Acceleration (g) = (16 bit Raw data) * 0.004 / 16

3.4.6 Get Z axis data of accelerometer

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x10	0x03	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data		Checksum
'S' or 'F'	0x10	LSB	MSB	1 byte

Data contains 16 bit signed int Z axis data.
Convert accelerometer data into g:

Acceleration (g) = (16 bit Raw data) * 0.004 / 16

3.4.7 Get X axis data of gyroscope

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x11	0x01	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data		Checksum
'S' or 'F'	0x11	LSB	MSB	1 byte

Data contains 16 bit signed int X axis data.
Convert gyroscope data into deg/sec:

Rate (deg/sec) = ((16 bit Raw Data) / 57.0) *0.01

3.4.8 Get Y axis data of Gyroscope

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x11	0x02	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data		Checksum
'S' or 'F'	0x11	LSB	MSB	1 byte

Data contains 16 bit signed int Y axis data.
Convert gyroscope data into deg/sec:

Rate (deg/sec) = ((16 bit Raw Data) / 57.0) *0.01

3.4.9 Get Z axis data of gyroscope

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x11	0x03	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data		Checksum
'S' or 'F'	0x11	LSB	MSB	1 byte

Data contains 16 bit signed int Z axis data.
Convert gyroscope data into deg/sec:

Rate (deg/sec) = ((16 bit Raw Data) / 57.0) *0.01

3.4.10 Get X axis data of magnetometer

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x12	0x01	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data		Checksum
'S' or 'F'	0x12	MSB	LSB	1 byte

Data contains 16 bit signed int X axis data.
Convert magnetometer data into gauss:

Magnetic Field (Gauss) = (16 bit Raw Data) / 1100.0)

3.4.11 Get Y axis data of magnetometer

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x12	0x02	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data		Checksum
'S' or 'F'	0x12	MSB	LSB	1 byte

Data contains 16 bit signed int Y axis data.
Convert magnetometer data into Gauss:

Magnetic Field (Gauss) = (16 bit Raw Data) / 1100.0)

3.4.12 Get Z axis data of magnetometer

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x12	0x03	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data		Checksum
'S' or 'F'	0x12	MSB	LSB	1 byte

Data contains 16 bit signed int Z axis data.
Convert magnetometer data into Gauss:

Magnetic Field (Gauss) = (16 bit Raw Data) / 980.0)

3.5 Servo Pod Commands

3.5.1 Set servo pod pan angle

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x06	Pan angle(0 - 180)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x06	1 byte

Note: PAN angle value in the range of 0 – 180.

3.5.2 Set servo pod tilt angle

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x07	Tilt angle(0 - 180)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x07	1 byte

Note: Tilt angle value in the range of 0 – 180.

3.5.3 Set servo pod aux angle

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x08	Aux angle(0 - 180)	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x08	1 byte

Note: Auxiliary angle value in the range of 0 – 180.

3.5.4 Get servo pod pan angle

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0xAA	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0xAA	Pan Angle	1 byte

3.5.5 Get servo pod tilt angle

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0xAB	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0xAB	Tilt Angle	1 byte

3.5.6 Get servo pod aux angle

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0xAC	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0xAC	Aux Angle	1 byte

3.5.7 Get servo pod ultrasonic sensor data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x05	0x0A	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 1	Checksum
'S' or 'F'	0x05	MSB	LSB	1 byte

3.6 Robotic Arm commands

3.6.1 Set robotics arm position

Command: Host to 0X Delta
Length: 8 bytes

Header			Command	Sub-command/ Data 0	Sub-command/ Data 1	Sub-command/ Data 2	Checksum
'N'	'E'	'X'	0x26	Robotic Arm Joint Number	Robotic Arm Angle	Robotic Arm Velocity	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x26	1 byte

For e.g. To move 3rd joint of Robotics arm at 90degree angle with a velocity of 30 degrees per second we will have to send following command string

Header			Command	Sub-command/ Data 0	Sub-command/ Data 1	Sub-command/ Data 2	Checksum
'N'	'E'	'X'	0x26	0x03	0x5A	0x1E	1 byte

Here,
Data 0 = 0x03 = Selects 3rd joint of robotic arm
Data 1 = 0x5A = Selects 90 degree as destination angle for specified joint
Data 2 = 0x1E = Selects 30 degrees/sec velocity to reach specified angle

3.7 Miscellaneous Commands

3.7.1 Set robot ID

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0xF9	Robot_ID	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0xF9	1 byte

Note: Robot ID should be between 0 to 255.

3.7.2 Get robot ID

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0xFA	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0xFA	ID	1 byte

3.7.3 Get hardware version ID

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0xFB	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0xFB	HW_VER	1 byte

3.7.4 Get software version ID

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0xFC	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0xFC	SW_VER	1 byte

3.7.5 Get potentiometer data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x05	0x09	1 byte

Response: 0X Delta to Host
Length: 5 bytes

Header	Command	Data 0	Data 1	Checksum
'S' or 'F'	0x05	MSB	LSB	1 byte

Note: Potentiometer Data is 12-bit and it is split into two bytes. The upper 4 bits of MSB are always zero.

3.7.6 Get AD7998 data

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x05	0x08	1 byte

Response: 0X Delta to Host
Length: 19 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x05	16 bytes of data Byte0 = CH0_LSB & Byte 1 = CH0_MSB and so on.	1 byte

Note: Refer AD7998 datasheet for more details on data format

3.7.7 Set GPIO panel LED

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x60	LED_DATA	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x60	1 byte

Note: Lower 4 bits of LED_DATA byte will have LED ON/OFF control.
e.g. if LED_DATA = 0x05 then GPIO LED status would be as follows

Header	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LED_DATA	X	X	X	X	OFF	ON	OFF	ON
GPIO Panel LED	-	-	-	-	LED 4	LED 3	LED 2	LED 1

3.7.8 Get GPIO panel LED status

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x61	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x61	1 byte	1 byte

Note: Lower 4 bits of Data byte will have LED ON/OFF status.
e.g. if Data byte = 0x0A then GPIO LED status would be as follows

Header	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data byte	X	X	X	X	ON	OFF	ON	OFF
GPIO Panel LED	-	-	-	-	LED 4	LED 3	LED 2	LED 1

3.7.9 Get GPIO panel switch status

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x62	0x00	1 byte

Response: 0X Delta to Host
Length: 4 bytes

Header	Command	Data	Checksum
'S' or 'F'	0x62	1 byte	1 byte

Note: Lower 4 bits of Data byte will have GPIO switch ON/OFF status.
e.g. if Data byte = 0x0E then GPIO switch status would be as follows

Header	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data byte	X	X	X	X	Released	Released	Released	Pressed
GPIO Panel S/W	-	-	-	-	S/W 4	S/W 3	S/W 2	S/W 1

3.7.10 Set buzzer

Command: Host to 0X Delta
Length: 6 bytes

Header			Command	Sub-Command/Data	Checksum
'N'	'E'	'X'	0x30	buzzer	1 byte

Response: 0X Delta to Host
Length: 3 bytes

Header	Command	Checksum
'S' or 'F'	0x30	1 byte

Buzzer = 1 → Buzzer ON
Buzzer = 0 → Buzzer OFF

4 Introduction to GUI

0X Delta 4 wheel drive robot can be controlled by GUI via serial and wireless mode. The baud rate used by GUI for above communication modes is 57600 bps.

⚠ 0X Delta 4 wheel drive robot is calibrated to work on Fire Bird VI GUI. Therefore same Fire Bird VI GUI can control Fire Bird VI as well as 0X Delta robot.

4.1 Installing Fire Bird VI GUI

Step1: Copy "setup.exe" file to your pc, which is located inside documentation CD/Softwares and Drivers/Robot GUI software from NEX Robotics. Double click "setup.exe" file.

Step 2: Click Next Button to continue.

Figure 4.1a

Step 3: Browse the location where set up will install or set the default location and click Next Button to start the installation.

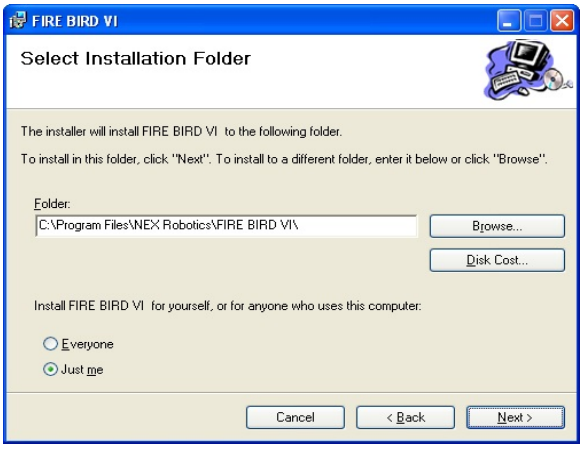


Figure 4.1b

Step 4: Press Next to Confirm Installation.

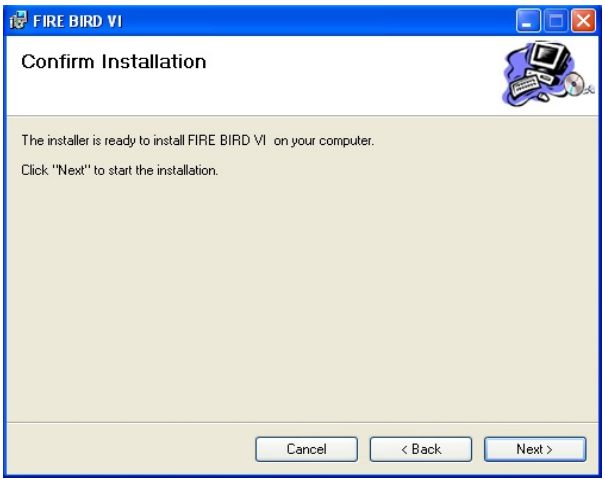


Figure 4.1c

Step 5: When installation is successfully completed, click Close to exit.

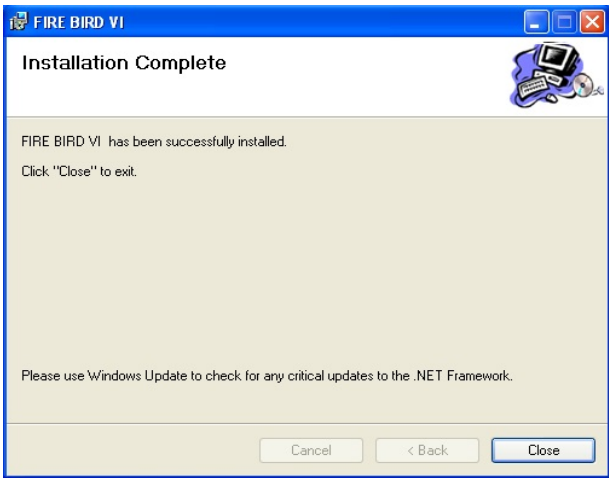


Figure 4.1d

4.2 Using GUI

Step 1: After successful installation go to Start -> All Programs -> FIRE BIRD VI -> FIRE BIRD VI or click on Fire Bird VI icon on your desktop to open the GUI.

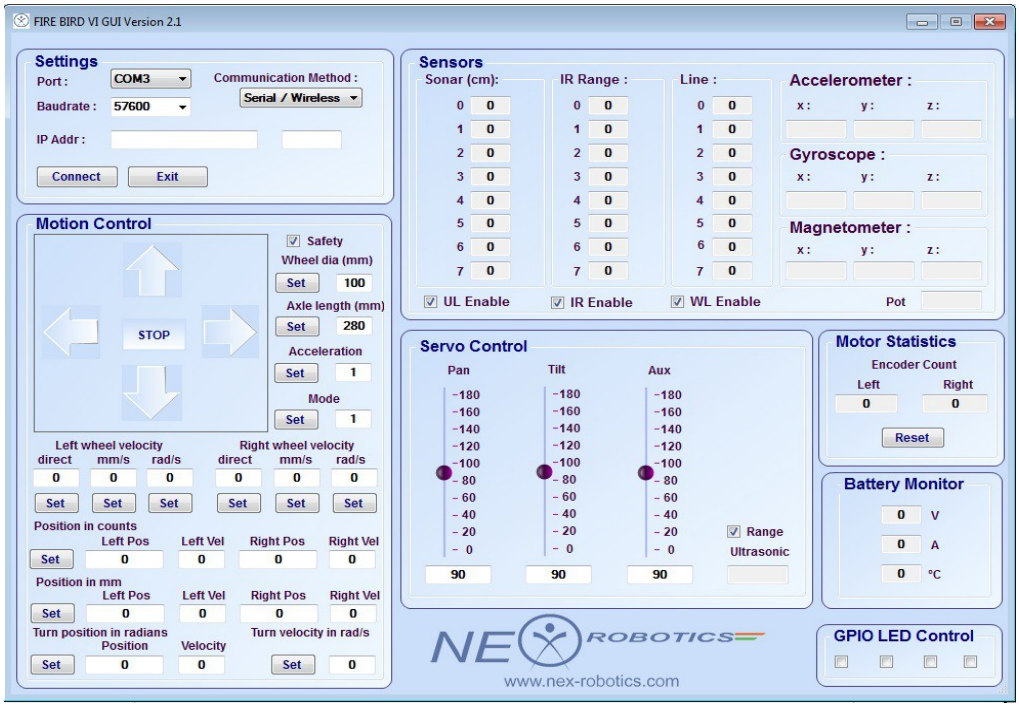


Figure 4.2: Selecting correct COM port

Step 2: Connect Robot with the PC using appropriate cable.

Step 3: For serial or wireless communication, choose communication method as “Serial/ Wireless”. Select “57600” as baud rate. Select the appropriate com port from drop down box and click Connect. For TCP/IP communication (Only if installed), choose communication method as “TCP/IP”. Enter the IP address and port of the TCP/IP module on the robot in “IP Addr” text box. The default port no. is 2000. (Note: For TCP/IP communication, the TCP/IP module on robot must be configured to connect to the same network as host computer on which GUI is running.)

Step 4: Click on “Connect” button to connect to the 0X Delta robot. This will show the data from each sensor in GUI. The robots movements can be controlled using “Motion Control” panel on the GUI.

⚠ White Line section of GUI is disabled for robot. Therefore whiteline reading on GUI will always be zero.

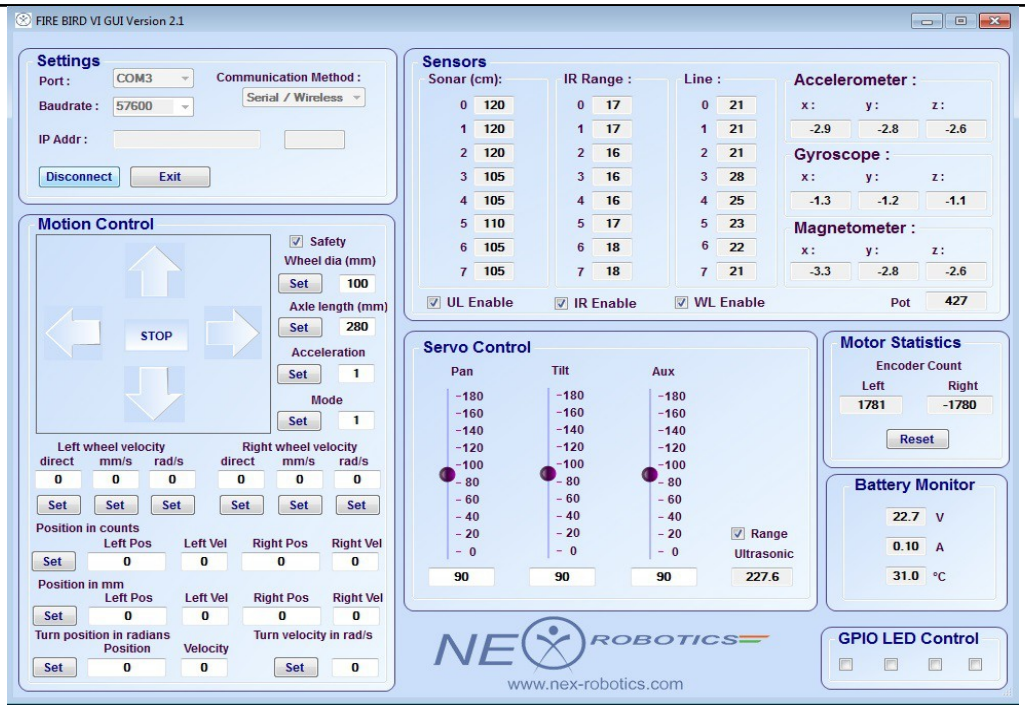


Figure 4.3: GUI showing robot's data

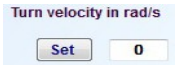
Panels on GUI:

1) **Settings**: Settings panel allows user to connect to robot. For serial or wireless communication, select the appropriate COM port. For TCP/IP communication select the "TCP/IP" from "Communication Method" drop down menu (Only if installed). Enter the IP address and port of the TCP/IP module on the robot in "IP Addr" text box. The default port no. is 2000. The "Connect" and "Disconnect" button can be used for connecting and disconnecting from 0X Delta. (Note: For TCP/IP communication, the TCP/IP module on robot must be configured to connect to the same network as host computer on which GUI is running.)

2) **Sensors**: This panel shows the data received from Ultrasonic, IR Range, White line sensor, accelerometer, gyroscope, magnetometer and potentiometer.

3) **Motion Control:** This panel provides the facility to control robots motions by setting Position, Velocity and Acceleration. The "Safety" check box enables the safety when checked. This restricts the robot's speed to 250 mm/Sec.

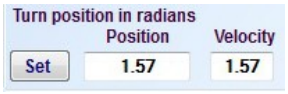
Above figure shows section to set velocity for both wheels of robot in terms of direct count, in mm/s as well as in radians/sec.



Above figure shows section to set angular velocity of robot in terms of radians/sec. The "Mode" check box sets the mode of the motion controller. The position control interface can be used only in Mode 2.
Position of robot can be set in terms of encoder counts or in terms of mm or in terms of radians (for in place turn only).

Above section of motion control unit will set robot position in terms of encoder counts. "Left Pos" sets the target position of left motor in terms of encoder count. "Left Vel" sets the velocity of left motor in the range of -512 to 512. "Right Pos" sets the target position of right motor in terms of encoder count. "Right Vel" sets the velocity of right motor in the range of -512 to 512.

Above section will set robot position in terms of mm. "Left Pos" sets the target position of left motor in terms of mm. "Left Vel" sets the velocity of left motor in terms of mm/sec. "Right Pos" sets the target position of right motor in terms of mm. "Right Vel" sets the velocity of right motor in terms of mm/sec.



Above section will set robot angular position in terms of radians. "Position" sets the target position of robot in terms of radians/sec. "Velocity" sets the velocity of robot in terms of radians/sec.

The direction of each motor is controlled by sign of the velocity component. A negative velocity will move the robot backwards.

4) **Battery Monitor:** It shows the battery parameters such as battery Voltage, Current and Temperature.

5) **Motor Statistics:** This panel shows motors encoder count for each motor. This panel also provides the facility to reset encoder count.

6) **Servo Control:** Using Pan, Tilt and Aux bars you can control the movements of servo motors connected to the servo connectors.

7) **GPIO LED Control:** This panel contains 4 check boxes. First check box corresponds to SW1 of GPIO panel like wise 4th check box corresponds to SW4 of GPIO panel. Enabling check box will Turn ON respective LED of GPIO panel.