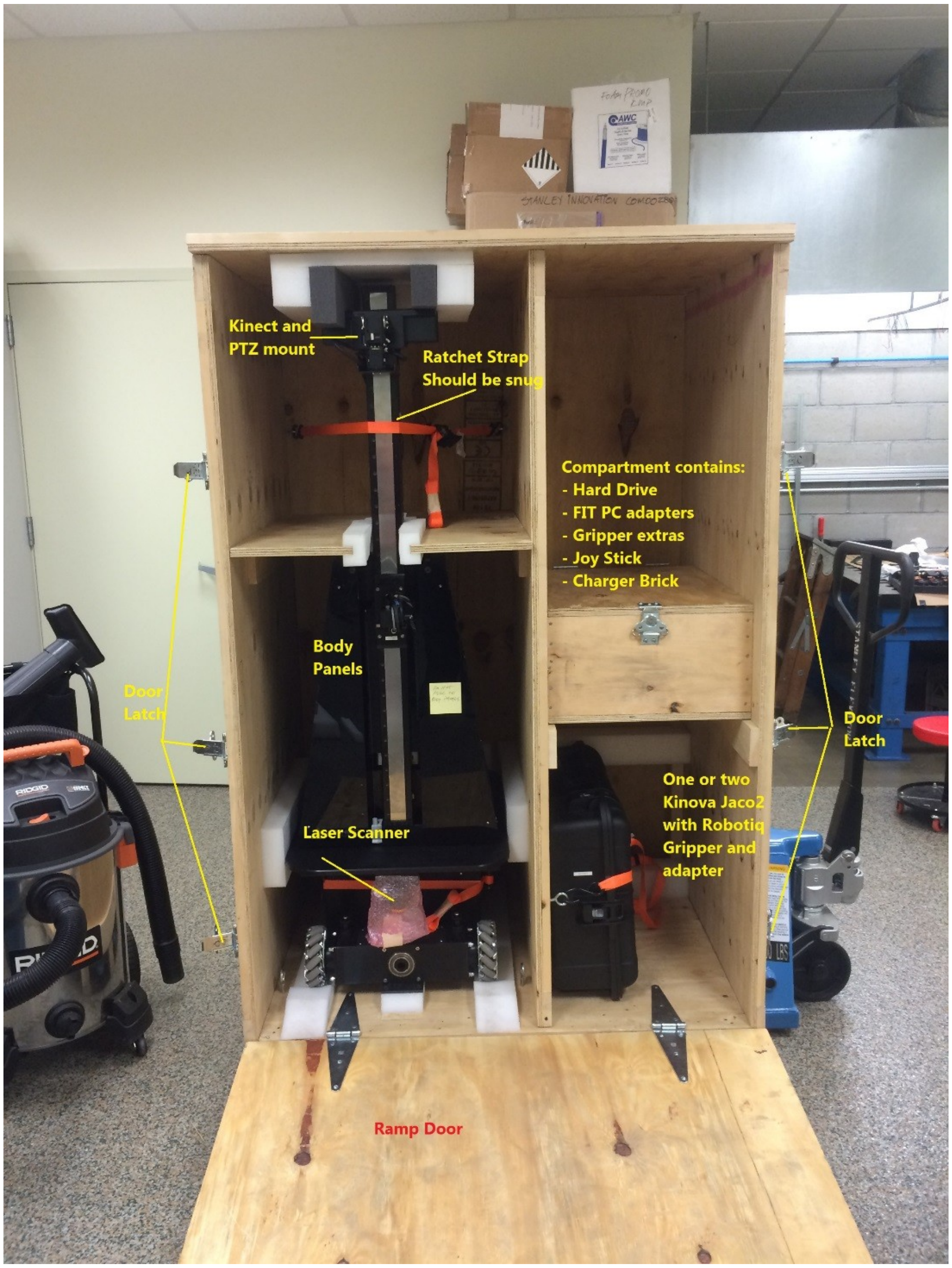
Vector Quick Start Guide

Stanley Innovation, Inc.

03/21/16

Contents

Uncrating



To uncrate Vector

1. Unlatch the crate door and slowly lower the door to the ground. It serves as a ramp.
2. Inspect the contents of the crate to ensure nothing was dislodged or damaged during shipping.

   a. Check the Kinect PTZ, laser, and body panels
   b. Also check orange ratchet straps and make sure they are still snug

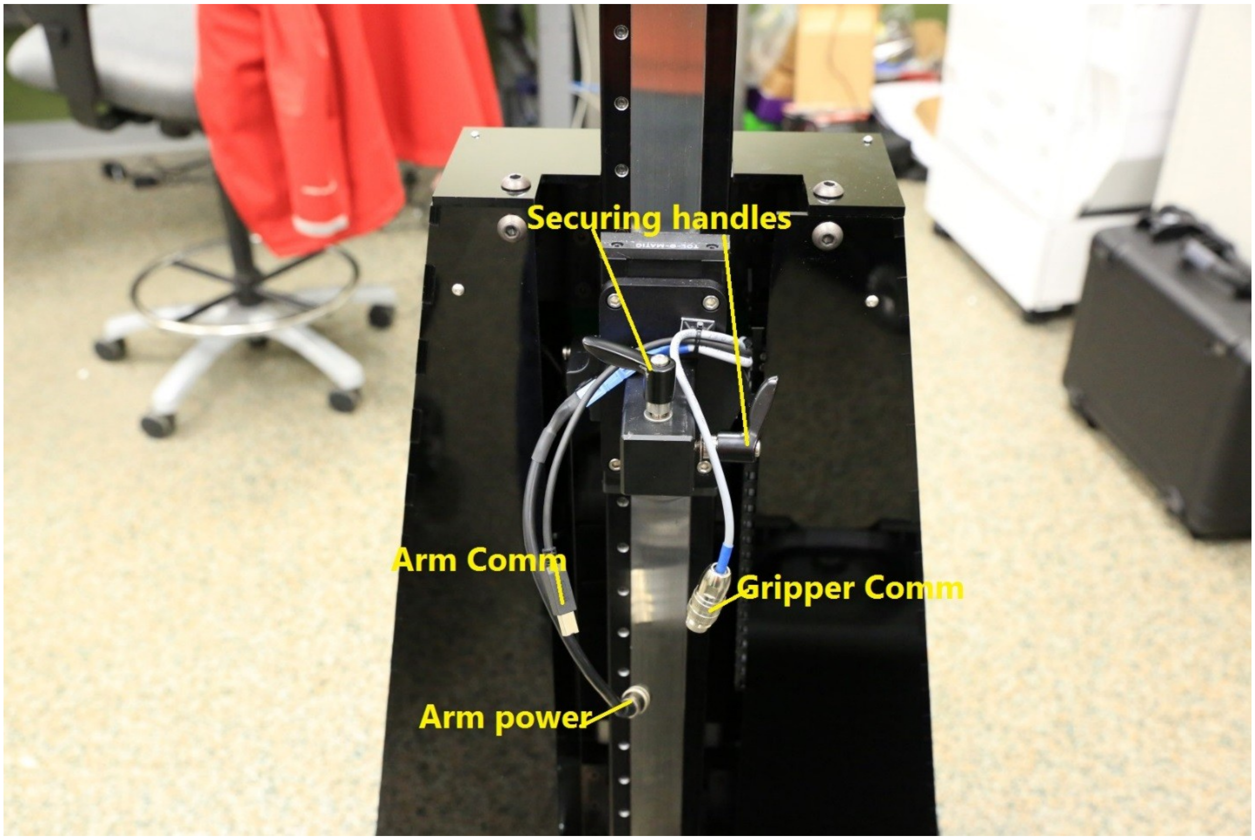
3. Open the compartment and unpack the contents there should be:
   a. Hard drive
      i. Contains backup image (clonezilla) of both PC's, the configuration files specific to the machine, and a virtual machine (VirtualBox) for remote connection to the robot.
   b. Logitech Extreme 3D Pro Joystick
   c. White box with FIT-PC connectors and adapters
   d. Brown box with gripper extras (harnesses, original adapter plate, etc)
   e. Charger Brick
4. Unlatch and open the Kinova arms
   a. The arm label will match the machine label on the linear actuator
   b. For systems with 2 arms they are labeled L (left) and R (right)
   c. Inside the Kinova box there are contents that came with the arm that were not used on the platform (USB stick with software, mount plate, power supply, clamps, expansion harness, etc)
5. Carefully remove the gray foam above the Kinect sensor
6. Carefully undo the 2 orange ratchet straps holding the platform in
   a. Be careful not to damage the laser or hit the acrylic panels with the straps or buckles
7. Very carefully have 2-3 people:
   a. 1-2 grab the payload plate
   b. 1 grab the linear actuator where the upper strap was attached
   c. **Do not grab the acrylic panels to remove the platform!!!!!!!!**
8. Carefully extract the platform by pulling on the payload plate and linear actuator.
   a. It is intentionally packed with a very snug fit, it will take some work to free it.
9. Once free be aware
   a. **Only push the platform by the linear actuator or payload plate**
   b. **The wheel actuators will power on when back driven hard enough. Due to the CANOpen State Machine they will enter active braking when back driven hard enough to raise the voltage high enough. They may make a grinding like noise when this happens, *IT WON'T DAMAGE ANYTHING BUT IT IS ANNOYING*. We are working on a solution so the platform can be moved around more quickly without powering on. *IF YOU MOVE IT SLOWLY IT WON'T DO THIS.***
10. Slowly guide the platform down the ramp and put it on a level surface where it will be convenient to do the rest of the setup
11. Remove the pink protective bubble wrap over the laser
12. UNPACKING IS COMPLETE!!!!

Preparing for first power on

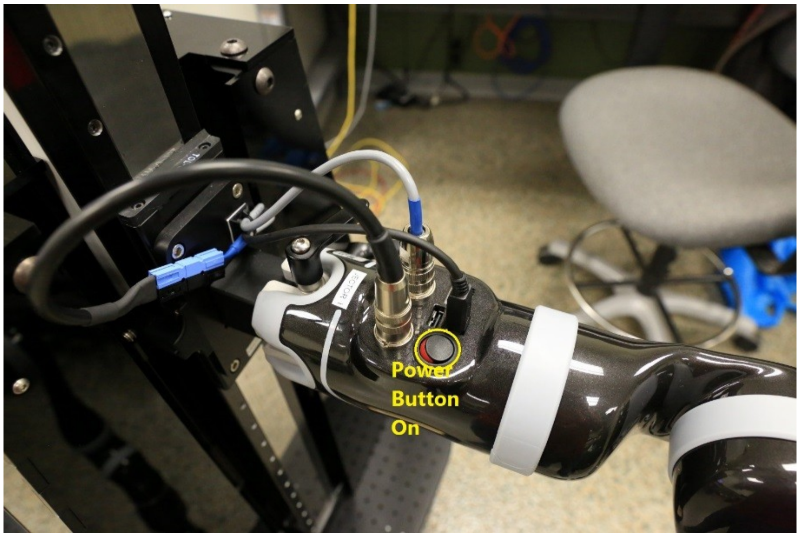
Once the system is unpacked and has been inspected for obvious damage it is time to prepare for the first power on.

Connect the Kinova Arms

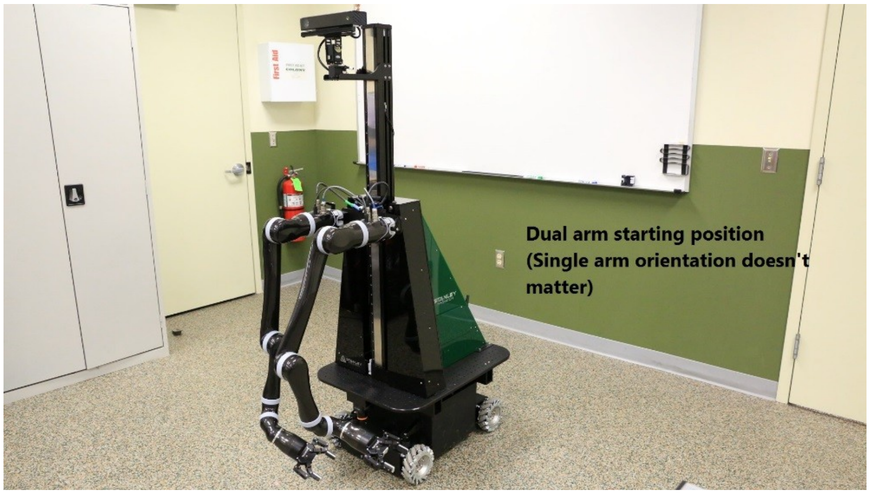
To connect the Kinova arms:



1. Unpack the arm(s) and check the labels
    a. For a dual arm system one will be labeled L (left) and one will be labeled R (right) this convention is looking at the robot from the rear (same as you would identify your left and right)
2. Cut and remove the temporary ziptie holding the arm cables on the horizontal arm support attached to the linear actuator carriage

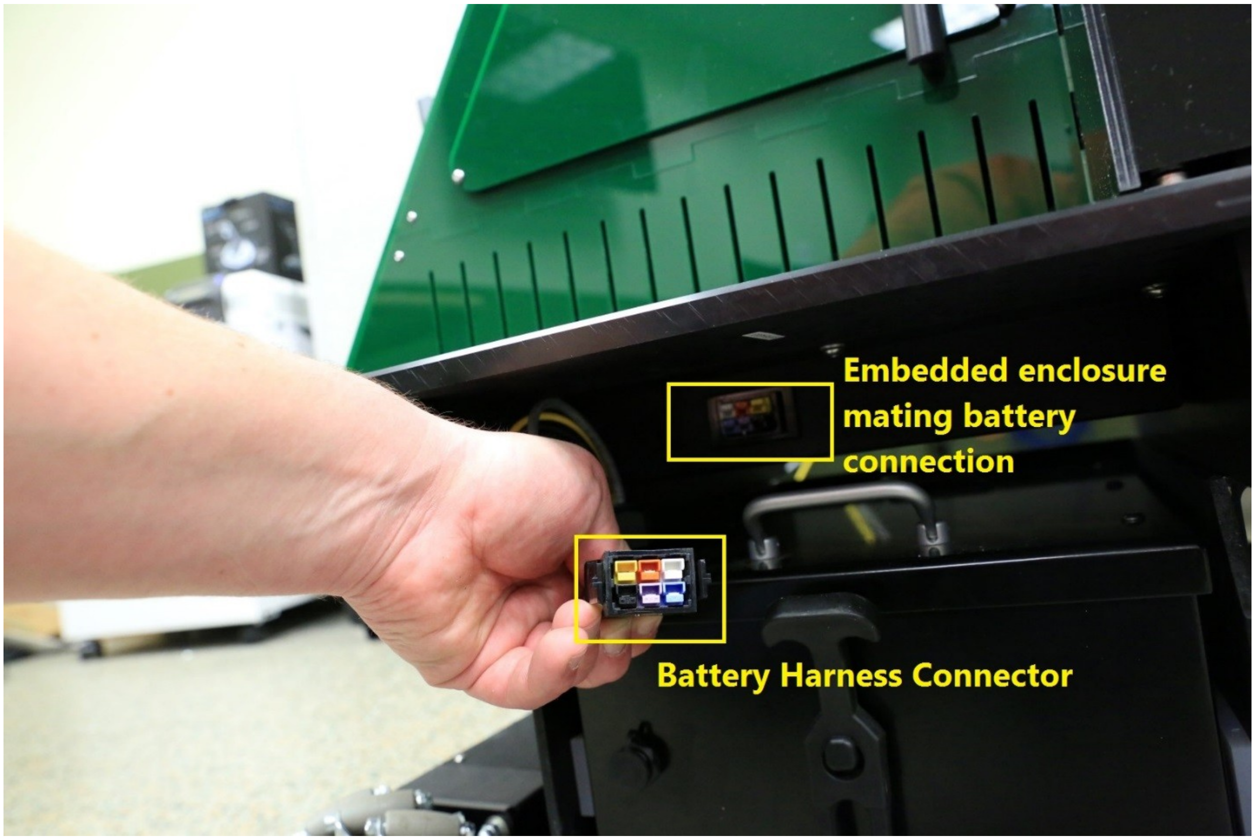


3. Loosen the handles used to secure the arm onto the horizontal support
4. Lift the washers and slide the arm onto the horizontal support
5. Firmly tighten the handles to secure the arm on the platform
6. Attach the cables as shown and ensure the power button is set to the ON position
7. Lower the arm so it is "dangling"
    a. **It is necessary to call the HomeArm API function in order to use Cartesian Velocity control via the API. THIS IS A BLIND MOVE OPERATION, THERE IS NO COLLISION CHECKING!!!! Use caution when powering on the first time, the arm may collide with objects or other arms if in the homing path. On a single arm system this is not much of a problem but on a dual arm system the arms should be started in the home orientation to avoid this.**
    b. **If the arm is going to collide with something either use the power switch on the arm or the ESTOP to cut power to the arm.**


Dual arm starting position
(Single arm orientation doesn't matter)

Connecting Battery Power

To connect battery power:



1. Cut the temporary zip tie securing the battery harness to the battery enclosure
2. Plug the battery harness into the embedded control box connector

Powering the system on

NOTE: When the platform is started it will run the upstart service which launches:

1. Platform Driver
2. PC Power Watchdog
3. Robot localization
4. Robot State Publisher and Joint State Publisher
5. Arm Driver
    a. Homes arm at initialization
6. Gripper Driver
    a. Homes gripper at initialization
7. Full System Tele-operation Node
    a. Controls all joints on the system
    b. Subscribes to /joy topic which gets published elsewhere
    c. Joystick uses mapping for Logitech Extreme 3D Pro (Included)
    d. Mapping and software is found in (/home/vector/vector_ws/src/vector_common/vector_ros/src/vector/vector_full_system_teleop.py)
8. Kinect V2 Bridge
9. Pan-Tilt Trajectory controller
10. Pan-Tilt Home Script
    a. Just does a little nod to indicate things are up and running

When powering on the system will boot-up (**this process takes about 1 minute in total**):

1. Embedded system will apply power to PC’s and all actuators
2. Power button ring led will pulse blue
3. Status LED will toggle yellow during initialization
4. Once drives are initialized the drives will emit white noise sound indicating they are enabled
5. Status LED will toggle between yellow-green indicating standby mode
6. Once the PC’s have booted they will run the upstart job
    a. Arms and grippers will home
    b. Kinect face led will light up white and shutoff
    c. Pan-Tilt will nod
7. The system is now up
8. If the system doesn’t perform all these steps, something went wrong (see Trouble Shooting)

Now that you understand the initialization process see the next section for how to power on……

To power the system on the first time:

1. Ensure the ESTOP button is not depressed, it is a twist lock (once pressed it latches and must be twisted to unlock)
2. Press the momentary power button
    a. Press it all the way in for about 0.5 sec
    b. Do not hold it pressed in
        i. Will enter bootloader if held for longer than 3 seconds; do not do this until you have read instructions (the PC's will not be protected for instant power off, so the procedure needs to be followed). A separate document and bootloader tool will be added to the repo on the next embedded firmware release
    c. If the blue LED ring around the power button doesn't begin to pulse within a second, the button was not pressed correctly (try again)

The system will now start the initialization sequence, as indicated above if it does not pass initialization and shuts down see troubleshooting

Powering off the machine

Once the machine has been allowed to initialize you can power off by quickly pressing the power button.

1. All power to torque producing actuators (except pan-tilt) will be disabled
2. The Upstart service has a PC power watchdog that will be notified and the PC's will be shutdown.
3. Any faults get written to the faultlog
4. The blue led will rapidly pulse and the system will completely power off in 30 seconds allowing time for the onboard PC's and peripherals to shutdown.

Connecting to the platform

Note: All the passwords for the PC's and the wireless are **Welcome00**

Setup remote virtual machine

1. Install VirtualBox
2. Add a new PC and select the VDI and virtual PC configuration included in the directory /vector_remote on the hard drive
3. Connect host machine to robot network
4. Ensure network connection is for guest is set to bridged using the wireless adapter on the host connected to the robot network
5. Start the virtual machine
6. Edit /etc/hosts and include the lines
    a. vector1 10.66.171.4
    b. vector2 10.66.171.3
7. Open a terminal and run
    a. ssh-keygen –t rsa
    b. press enter on all the prompts
8. Copy the SSH key to vector1
    a. ssh-copy-id vector@vector1
    b. Enter the common password: **Welcome00**
9. SSH onto vector1 to test connection
    a. ssh vector@vector1
10. Close terminal
11. Edit the file 50.vector_network_config.sh to have the correct settings (both locations unless you clean and re-compile)
    a. Sourced from /home/vector/vector_ws/devel/etc/catkin/profile.d
    b. Compiled from /home/vector/vector_ws/src/vector_network/env-hooks
12. Copy the vector_config.sh file from the harddrive /vector1_pc_config to the directory
    a. /home/vector/vector_ws/src/vector_common/vector_config
13. Source the workspace
14. Plug in the extreme 3D Pro Joystick to the host and add the device to the guest
15. Run:
    a. roslaunch vector_viz view_robot.launch to open the RVIZ configuration to see the robot
    b. roslaunch vector_remote_teleop vector_remote_teleop.launch dev:=/dev/input/***<the joystick device>***
        i. ***<the joystick device>*** usually defaults to js2 for virtual machines

There are 3 important aliases that can be run on vector1 once an SSH connection is established

vstop (stops upstart service)

vstart (starts upstart service or restarts once it is stopped)
vchk (tails the system log for the vector-core upstart service and prints the most recent 100 ROS output lines)

Control the system with Teleop

The controller mapping is defined in the variable self.ctrl_map (the buttons are number on the joystick and match the numbers for 'index' key) in the file:

/home/vector/vector_ws/src/vector_common/vector_ros/src/vector/vector_teleop_full_system.py

Control the mobile base

Press Button 9 to enable mobile base control (pan-tilt and arm control will be disabled)

1. Button 2 puts platform in Standby Mode(no motion commands allowed)
2. Button 3 puts platform in Tractor Mode
3. Trigger (Button 0) is dead man switch and must be pressed to issue motion commands when in tractor
4. For-Aft axis maps to X velocity
5. Left-Right axis maps to Y velocity
6. Twist axis maps to Z angular velocity (yaw)

Control Pan-Tilt

Press Button 8 to enable pan-tilt control (base and arm control will be disabled)

1. Trigger (Button 0) enables commands
    a. release to hold current position
2. For-Aft axis maps to tilt position
3. Twist axis maps to pan position

Control Arms

This is the default arm for single arm systems

Press Button 10 to enable right arm control (base, pan-tilt and left arm (if present) control will be disabled)

Press Button 11 to enable left arm control (if present) (base, pan-tilt and right arm control will be disabled)

1. Trigger (Button 0) enables commands
    a. release to hold current position
2. D-PAD up/down rotates the end-effector around y axis
3. D-PAD left/right rotates the end-effector around x axis
4. Button 2 and 3 rotate the end-effector around Z axis
5. For-Aft axis maps to x Cartesian velocity of end-effector
6. Left-Right axis maps to y Cartesian velocity of end-effector
7. Twist axis maps to z Cartesian velocity of end-effector
8. If the thumb button (Button 1) is pressed the Twist axis maps to linear actuator velocity through a mapping function that outputs position by integrating the signal

9. Flipper paddle axis opens and closes the gripper

Embedded firmware updates

From time to time Stanley Innovation will release new embedded firmware to correct bugs or improve system functionality. This section covers the process of updating the embedded firmware on the platform.

Enter Bootloader Mode

To put the embedded system into “bootloader” mode, from the powered off state, press and hold the power button. While the button is held, the blue light ring around the button will illuminate. It will first slowly “breath”, followed by a quick “blink”. Once the light ring blinks, it can be confirmed the embedded system is in bootloader mode and you can release the power button.

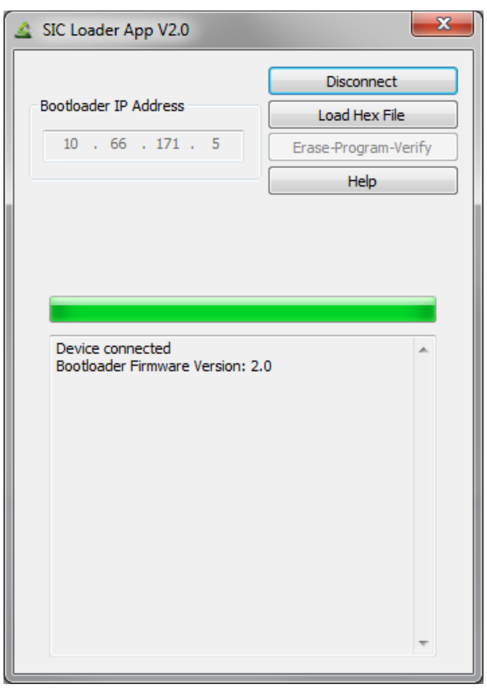
Connect To System

With the unit now powered on in bootloader mode, connect to the system’s local network either via WiFi or with a hardwired connection to the router.

Using the Bootloader Tool

Open the “SIC Loader App v2.0.exe” tool, and click connect. You should now see that the device is connected.

NOTE: This tool only works in Windows 7 or later

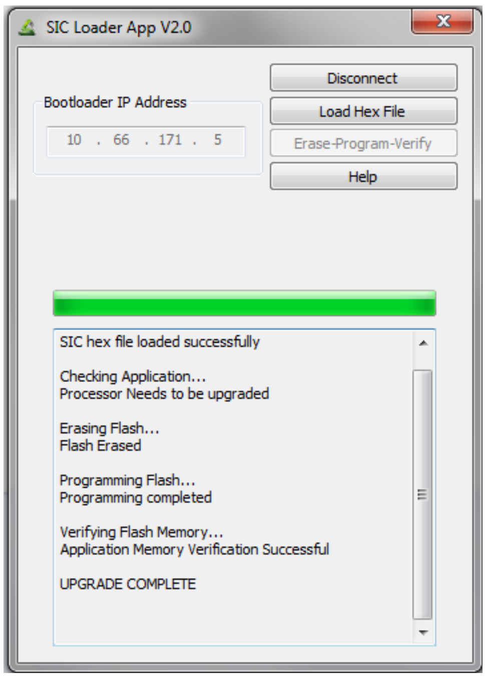


If you see a message stating “No Response from the device. Operation failed”, ensure you have a network connection to the embedded system and that you can ping 10.66.171.5.

Choose “Load Hex File”, and browse to and select the appropriate hex file. You should now see “SIC hex file loaded successfully”.

With the hex file selected, click “Erase-Program-Verify”, and wait for this to complete.

You should now see a message “UPGRADE COMPLETE”.

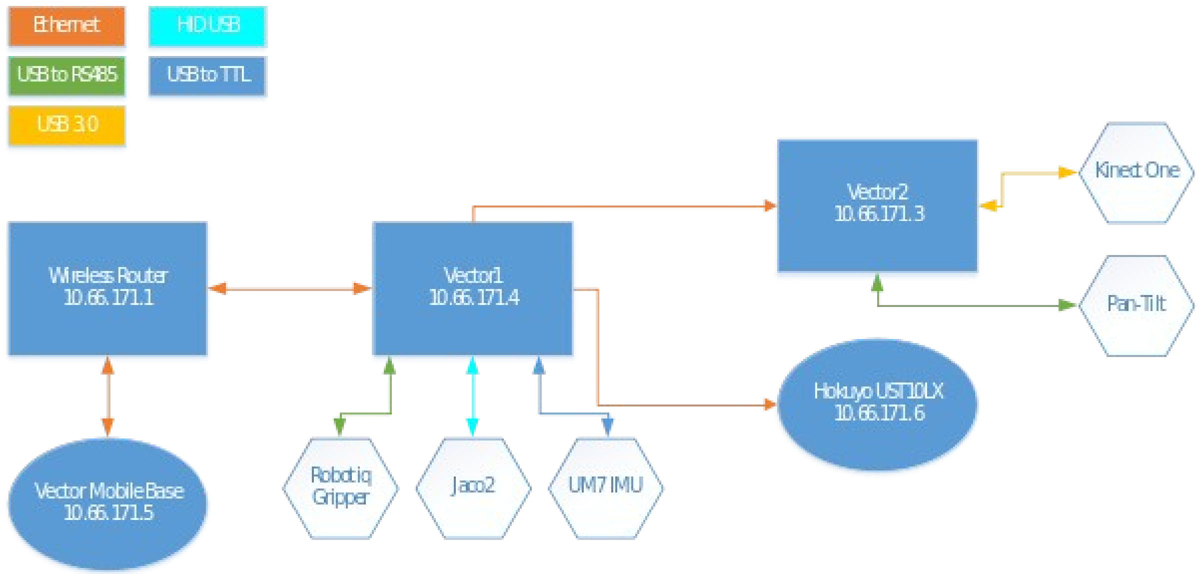


You have now successfully completed the upgrade. Now click Disconnect, and close the SIC Loader App. A short press of the power button will begin the shutdown process. Wait several seconds for the light ring and indicator LED to turn off. Everything should now be safely powered down and you can start the system normally.

Network configuration

Block Diagram

The following block diagram outlines the communication connections in the system



Vector1 PC

The vector1 PC is the ROS master in the system it is responsible for launching the entire system via upstart service. All ports aside from eth5 are bridged to the master IP. Eth5 is left to DHCP for easy connection to another network. It runs all nodes except the pan-tilt controller and the kinect bridge. Vector1 also is a stratum 10 chrony server so that other PCs in the network can synchronize time with it.

Ethernet interfaces	Type	IP
br0: eth0-4	Static IP	10.66.171.4 (hostname vector1)
eth5	DCHP	NA

The ROS networking variables are set by the environment hooks file in **~/vector_ws/src/vector_network/env-hooks/50.vector_network_config.sh on vector1**

ROS networking environment variables	Value
ROS_MASTER_URI	http://10.66.171.4:11311
ROS_IP	10.66.171.4
ROBOT_NETWORK	br0

Vector2 PC

The vector2 PC is responsible for running the pan-tilt controller and the kinect bridge. It left primarily unused for perception development. Vector2 is connected to br0 on vector1. Vector2 is a client to the vector1 chrony server, this is done to synchronize time between the PCs.

Ethernet interfaces	Type	IP
eth0	Static IP	10.66.171.3 (hostname vector2)
wlan0	DCHP	NA

The ROS networking variables are set by the environment hooks file in **~/vector_ws/src/vector_network/env-hooks/50.vector_network_config.sh on vector2**

ROS networking environment variables	Value
ROS_MASTER_URI	http://10.66.171.4:11311
ROS_IP	10.66.171.3
ROBOT_NETWORK	eth0

Wireless router

The system includes a Linksys WRT-1900AC wireless router. The SID depends on the platform. It has a default IP of 10.66.171.1, and allows DHCP for none static connections. The username is admin and the password is Welcome00 for the wireless key and the login to the router configuration.

Vector Mobile Platform

The vector mobile platform IP is 10.66.171.5 and is connected via the wireless router to the system

Hokuyo Laser Scanner

The Hokuyo UST-10LX IP is 10.66.171.6 and uses the default port for the Hokuyo interface. It is connected directly to vector1 via br0.

Other connections

There are a few other connections on the platform they are listed below:

PC	Peripheral	Connection Type
vector1	CH Robotics UM7 IMU	USB to TTL serial
vector1	Kinova Jaco^2	USB
vector1	Robotiq 85-gripper	USB to RS485
vector2	Pan tilt	USB to RS485
vector2	Kinect ONE	USB 3.0

System launch files and upstart job

The entire system is launched via upstart service at power-on. The upstart service and the launch sequence is controlled by vector1. The packages are located on vector1 at:

~/vector_ws/src/vector_robot

Upstart Job

The upstart job, is a service that runs when the PC is powered on. It performs the environment setup and a few checks to make sure the system is up and running before launching ROS.

Templates

The templates for the upstart service are located at:

~/vector_ws/src/vector_robot/vector_upstart/templates

The file job-start-vector.em is responsible for controlling the upstart service and is the file you should modify if you want to add/change the service.

Modifying

You only need to re-install the upstart service after making changes to the following files on vector1 (the launch files will be explained in the next section):

1. **~/vector_ws/src/vector_robot/vector_bringup/launch/vector_system.launch**
2. **~/vector_ws/src/vector_robot/vector_upstart/templates/job-start-vector.em**

The environment variable configuration, other launch files, nodes, etc can all be modified without re-installing the upstart service.

To reinstall the upstart service on vector1:

1. Modify the upstart service template or the system launch file
2. Open a terminal or SSH to vector1 and run **vstop**
3. Uninstall the upstart service
    a. **rosrun vector_bringup uninstall_vector_core**
4. Reinstall the upstart service
    a. **rosrun vector_bringup install_vector_core**

This only needs to be done on vector1 as it is responsible for launching everything (including the nodes on vector2).

Launch files

System

Vector1 launches all the nodes in the system. The system bringup is time controlled and is broken up into major components (platform, sensors, localization, teleop, manipulation, perception). The timing is controlled by the file:

~/vector_ws/src/vector_robot/vector_bringup/launch/vector_system.launch

If this file is modified you must re-install the upstart service for the changes to take effect.

Environment Variables

Environment variables control many aspects of the launch, removing the need to modify the launch files themselves, these environment variables are loaded at startup and are located in the file:

~/vector_ws/src/vector_common/vector_config/vector_config.sh

Any of these can be modified at any time and the upstart service can just be restarted for them to take effect. See the variable descriptions in the file for what each one does.

IMPORTANT: This file should be the same on each PC in the vector system for things to work correctly when launching a distributed system.

Platform

The mobile base driver, power watchdog, robot_description and joint_state_publisher are launched by the file:

~/vector_ws/src/vector_robot/vector_bringup/launch/platform/vector.launch

The mobile base configuration parameters (velocity/acceleration limits and other runtime parameters) can be perminantly modified (ie they are loaded at startup) in the file:

~/vector_ws/src/vector_robot/vector_bringup/launch/platform/config/vector_params.yaml

Or can be modified at runtime using rqt_reconfigure and the dynamic_reconfigure server launched in the vector_driver node.

Sensors

All the sensors in the system (laser scanners, IMU, etc) and associated filtering (with the exception of the kinect) are launched from the file:

~/vector_ws/src/vector_robot/vector_bringup/launch/sensors/vector_sensors.launch

Depending on the sensors present as defined in the environment configuration, it will launch the sensor drivers and associated filtering. The filter configurations can be found in:

~/vector_ws/src/vector_robot/vector_bringup/launch/sensors/config

Localization

If the environment variable **VECTOR_USE_PLATFORM_ODOMETRY** is set to true, the embedded system manages the odometry estimate. This is the preferred setup, unless other sensors for more accurate odometry are added to the system (such as TOF localization sensors). The embedded system fuses the wheel data and onboard IMU to generate an accurate odometry estimate, which gets published along with the transform if the environment variable is set true.

Otherwise odometry is calculated using robot_localization EKF, fusing various available sources. The file:

~/vector_ws/src/vector_robot/vector_bringup/launch/localization/vector_odometry.launch

Controls which data sources and what variables from those sources are fused in the estimate (when not using the onboard odometry). By default the external IMU and platform odometry are fused. The documentation for robot_localization can be found here: http://wiki.ros.org/robot_localization

AGAIN this estimation method is only exposed incase the onboard odometry can be improved by additional sensors we recommend setting VECTOR_USE_PLATFORM_ODOMETRY=true in vector_config.sh

Manipulation

The manipulation in the system is launched with the file:

~/vector_ws/src/vector_robot/vector_bringup/launch/manipulation/vector_manipulation.launch

As delivered it just launches a simple Jaco2 node for teleoperation control and the Robotiq 85 gripper nodes. This file should be were you add the MoveIt! launch, trajectory controllers, etc once they have been developed.

It was decided by the customers that Stanley Innovation would not be responsible for delivering MoveIt! integration, trajectory control or any advanced manipulation functionality. As such we will not be providing support for that functionality; although there is some “freebie” development that was included to help get things going. None of it is

complete or tested and should be treated as such. We will not be providing documentation or support for those packages.

Teleoperation

The teleoperation, interactive marker control via RVIZ, and cmd_vel mux are launched via the file:

~/vector_ws/src/vector_robot/vector_bringup/launch/teleop/vector_teleop.launch

By default the full_system_teleop node is used which allows the user to control all degrees of freedom in the system.

The cmd_vel mux identifies sources of commands for the mobile base (teleop, assisted teleop, interactive marker, and navigation) and also assigns priority to each. The purpose is to ensure that the user always has the ability to override autonomous commands if desired. The configuration is found in:

~/vector_ws/src/vector_robot/vector_bringup/launch/teleop/config/cmd_vel_mux.yaml

Vector2 (Kinect One bridge and Pan-Tilt)

The Kinect ONE bridge and pan-tilt driver/controllers are launched from vector1 remotely. The file that controls the remote launch is:

~/vector_ws/src/vector_robot/vector_bringup/launch/vector2/vector2.launch

This file is responsible for the remote launch of the pan-tilt and Kinect One bridge on vector2.

Kinect One Bridge

The file that controls the Kinect One bridge launch is located at:

~/vector_ws/src/vector_robot/vector_bringup/launch/vector2/kinect/kinect2_bridge.launch

The documentation for the iai_kinect2 package can be found here:

https://github.com/code-iai/iai_kinect2

Pan-Tilt Driver and Controller

The files for launching the pan tilt drivers and controller are located here:

~/vector_ws/src/vector_robot/vector_bringup/launch/vector2/pan_tilt

The launch file for the controller manager, controller spawner, and the configuration files for the controller are all located here. The documentation for the dynamixel package can be found at:

http://wiki.ros.org/dynamixel_motor and here
https://github.com/arebgun/dynamixel_motor

System feedback and published topics

Vector System Nodes, Topics and important files

The vector package is based in python the associated code can be found here:

~/vector_ws/src/vector_common/vector_ros/src/vector

The important nodes and files are

File	Function
vector_comm.py	Platform driver
vector_data_classes.py	Publishes all /vector/feedback topics
system_defines.py	Interface definitions
vector_teleop_full_system.py	Full system teleoperation node
vector_system_wd.py	Shuts the PC system down upon embedded cmd
vector_control_marker.py	Interactive maker control for vector via RVIZ

Topics Subscribed

Topic	Function
/vector/cmd_vel	Twist command for platform
/vector/linear_actuator_cmd	Linear actuator position command
/vector/gp_cmd	General purpose command for switching modes
/move_base/DWAPlannerROS/parameter_updates	Message to indicate movebase is online and the performance parameters for the system should be sent to movebase

Topics Published

/vector/feedback/faultlog

Data	Type	Description
data	Uint32[]	Raw faultlog data array extracted at startup

/vector/feedback/status

Data		Description
fault_status_words	Uint32[]	These are the compact fault status of the system they contain the bitmaps defining active system faults. (see system_defines.py)
operational_time	Float32	Time in seconds that the system has been

		operational
operational_state	Uint32	General purpose command for switching modes
dynamic_response	Uint32	Present dynamic response of the system to any present faults (see system_defines.py)
machine_id	Uint32	Machine identifier

/vector/feedback/battery

Data		Description
battery_status	Uint32	The battery status bits. (see system_defines.py)
battery_soc	Float32	The state of charge in 0-100%
battery_voltage_VDC	Float32	Battery bus voltage sensed by the battery
battery_current_A0pk	Float32	Battery bus current sensed by the battery
battery_temperature_degC	Float32	The maximum temperature of the battery

/vector/feedback/propulsion

Data		Description
wheel_motor_status	Uint32[]	Array containing the motor status for each wheel motor in the system. (see system_defines.py)
wheel_motor_current_A0pk	Float32[]	Array containing the motor current reported by each wheel motor in the system
wheel_motor_speed_rps	Float32[]	Array containing the motor speed reported by each wheel motor in the system
wheel_motor_position_rad	Float32[]	Array containing the motor position reported by each wheel motor in the system
linear_motor_status	Float32	Status of the linear actuator motor
linear_motor_current_A0pk	Float32	Linear actuator motor current
linear_motor_speed_rps	Float32	Linear actuator motor speed
linear_motor_position_rad	Float32	Linear actuator motor position

/vector/feedback/dynamics

Data		Description
x_vel_target_mps	float32	
y_vel_target_mps	float32	
yaw_rate_target_rps	float32	
linear_actuator_target_m	float32	
x_vel_limit_mps	float32	
y_vel_limit_mps	float32	

yaw_rate_limit_rps	float32	
linear_actuator_vel_limit_mps	float32	
wheel_vel_mps	float32[]	
wheel_pos_m	float32[]	
linear_actuator_vel_mps	float32	
linear_actuator_position_m	float32	
x_accel_mps2	float32	
y_accel_mps2	float32	
yaw_accel_mps2	float32	
yaw_angle_rad	float32	
odom_yaw_angle_rad	float32	

The rest of the topics will get added tonight

Navigation Demos

Coming tomorrow

Troubleshooting

For now just try powering the system off and powering it back on

Coming soon see GitHub repo for updates