|  |  |
| --- | --- |
|  | SIBOT –  MANUAL |
|  |  |
|  | Stanley Innovation, Inc.  2/12/15 |

Contents

[Receiving the robot: 4](#_Toc411524493)

[Uncrating 4](#_Toc411524494)

[Installing the batteries 5](#_Toc411524495)

[Manuals 5](#_Toc411524496)

[SIBOT Overview 6](#_Toc411524497)

[Networking 6](#_Toc411524498)

[XBOX360 Controller Mapping 6](#_Toc411524499)

[Robot Power-ON and Emergency Disable 7](#_Toc411524500)

[Switch off the service disconnect 7](#_Toc411524501)

[Hardware kill button 7](#_Toc411524502)

[Powering on and off 7](#_Toc411524503)

[Charging SIBOT 8](#_Toc411524504)

[Calibration 8](#_Toc411524505)

[Prepare a remote computer 9](#_Toc411524506)

[Using the virtual machine included 9](#_Toc411524507)

[Starting the virtual machine 9](#_Toc411524508)

[Manually preparing a remote PC 10](#_Toc411524509)

[Installing Ubuntu 12.04LTS and ROS HYDRO 10](#_Toc411524510)

[Setting up bash aliases for remote monitoring: 10](#_Toc411524511)

[Creating remote catkin workspace for visualizing data: 10](#_Toc411524512)

[Powering on and connecting for the first time: 11](#_Toc411524513)

[Connecting to the Wireless: 11](#_Toc411524514)

[Force time sync with sibot1: 11](#_Toc411524515)

[Setting up SSH Keys: 12](#_Toc411524516)

[Setting up SFTP: 13](#_Toc411524517)

[SIBOT Functions 14](#_Toc411524518)

[Starting and stopping a function 14](#_Toc411524519)

[Basic Teleoperation 14](#_Toc411524520)

[Command 14](#_Toc411524521)

[Description 14](#_Toc411524522)

[Assisted Teleoperation 15](#_Toc411524523)

[Command 15](#_Toc411524524)

[Description 15](#_Toc411524525)

[2D SLAM Mapping 15](#_Toc411524526)

[Command 15](#_Toc411524527)

[Description 15](#_Toc411524528)

[Global Navigation 16](#_Toc411524529)

[Command 16](#_Toc411524530)

[Description 16](#_Toc411524531)

[Local Navigation 17](#_Toc411524532)

[Command 17](#_Toc411524533)

[Description 17](#_Toc411524534)

[Global Navigation with Pre-set goals 17](#_Toc411524535)

[Setting Pre-set goals 17](#_Toc411524536)

[Navigating Pre-set goals 18](#_Toc411524537)

[GPS navigation 18](#_Toc411524538)

[SIBOT Configuration Parameters 19](#_Toc411524539)

[RQT\_reconfigure 19](#_Toc411524540)

[Modifying configurations files at launch 19](#_Toc411524541)

[Odometry 19](#_Toc411524542)

[Other odometry input sources 19](#_Toc411524543)

[Platform velocity and acceleration limits 21](#_Toc411524544)

[Navigation Stack Configuration 21](#_Toc411524545)

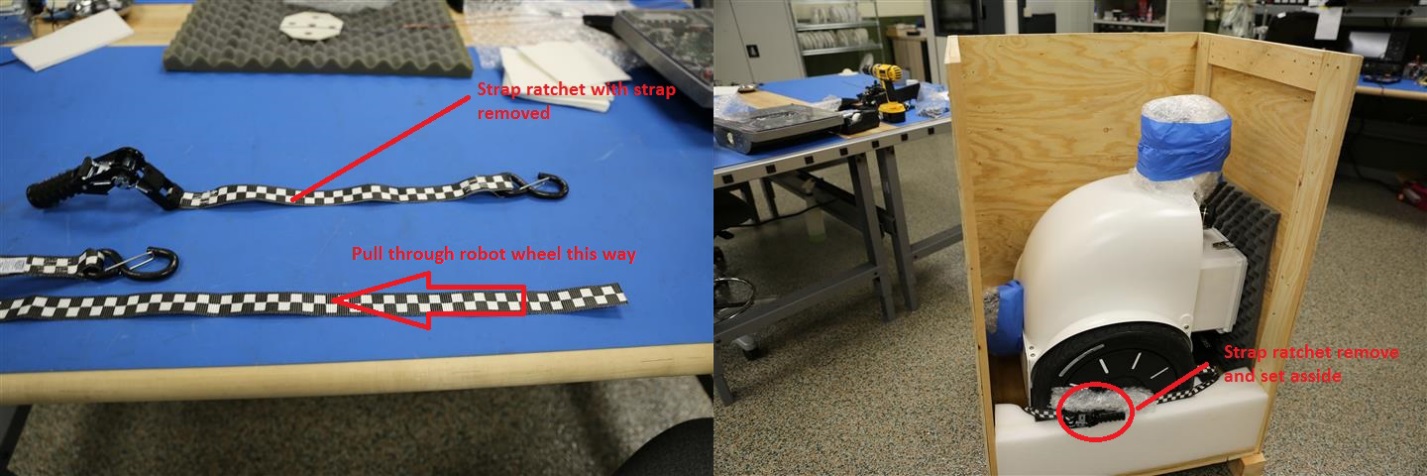
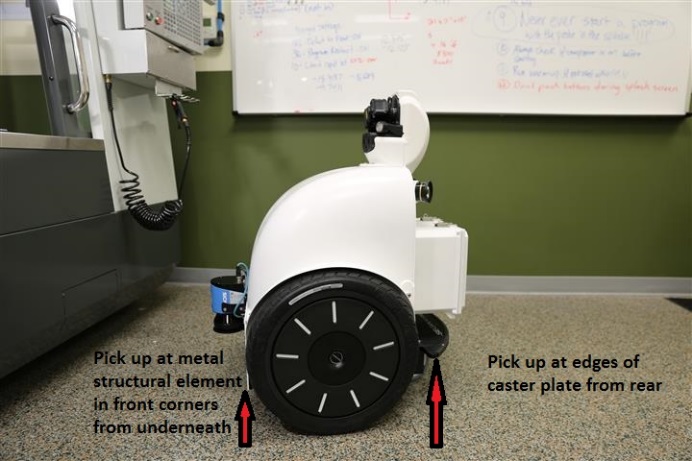
[Perception Filtering 22](#_Toc411524546)

[Troubleshooting 22](#_Toc411524547)

# Receiving the robot:

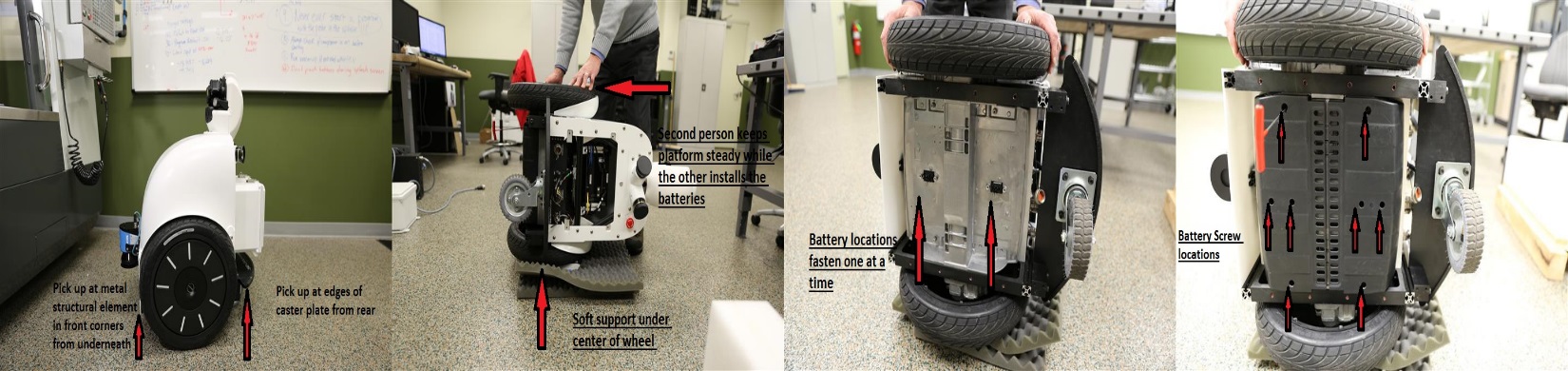
## Uncrating

Be sure to only lift the robot from structural members. *!!!!!****Be very careful not to lay the robot on any parts of the body during this process. The white plastic body parts are not structural; do not support the weight of the platform with the shell of the head or the body parts!!!!!!* You should never lift the platform by the body; this will likely result in damage to the body.** To unpack the robot:

1. Remove the screws securing the crate top and place it to the side
2. Remove the screws for each side panel and place them to the side
   1. **Important: have 2 people helping; one removing screws and one holding the panel to keep it from falling on the robot**
3. Once all the panels have been removed; release the strap ratchets on both sides.
   1. **DO NOT PULL THE STRAP AGAINST THE BODY!!!!**
4. Remove the strap from the strap ratchet on both sides
5. For each side:
   1. Unhook the clasp **(ONLY ON THE RATCHET SIDE)** from the eye hook and set the strap ratchet end to the side
   2. Very carefully pull the strap end through the robot body to free it
      1. **MAKE SURE IT DOES NOT CATCH AND PULL ON THE FENDERS**
   3. Unhook the clasp from the eye hook and set the strap aside
6. With 2 people who can lift 70lbs; pick the robot straight up out of the crate grasping at the locations indicated

## Installing the batteries

The batteries will only have to be installed once. They had to ship separately because they are class 9 hazmat due to the lithium content. *!!!!!****Be very careful not to lay the robot on any parts of the body during this process. The white plastic body parts are not structural; do not support the weight of the platform with the shell of the head or the body parts!!!!!!***

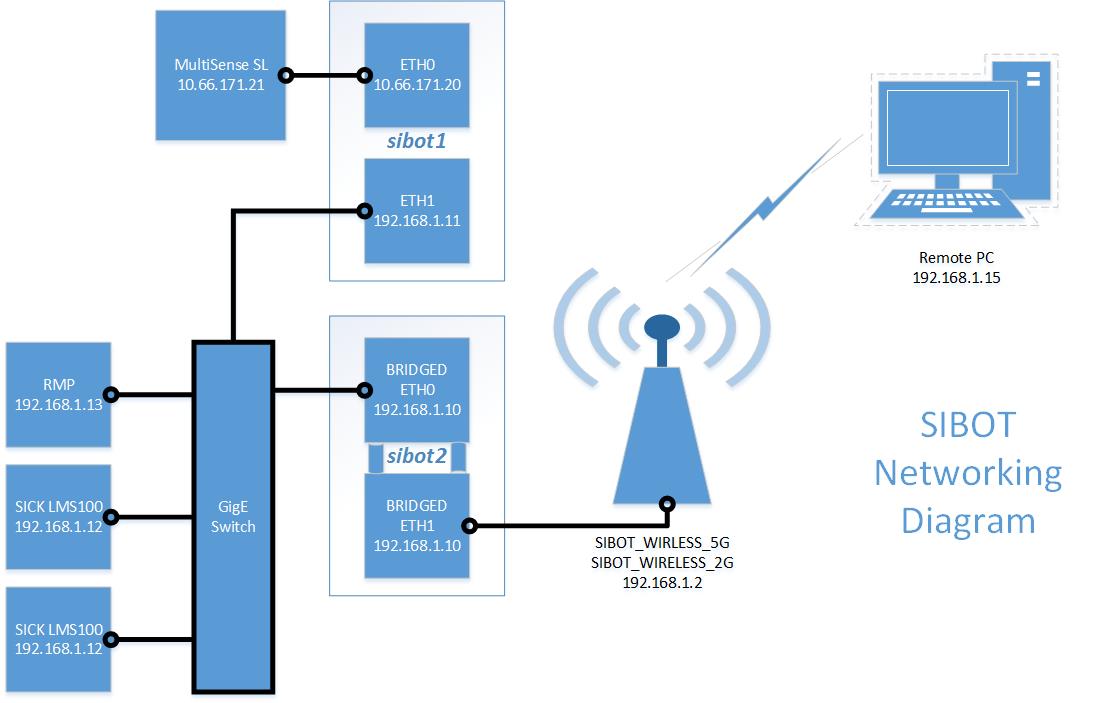
1. Get the 8 battery screws included in parts shipment and a long T-handle 4mm Allen key
2. Place a block of wood (3-6 inches thick) on the ground.
3. Cover it with foam or a soft pad
4. With 2 people roll the robot to the side with the block under the wheel being sure not to grab the body.
   1. Only grab the points illustrated in figure xx
5. Install each battery securing the 4 screws
6. Roll the platform back on its wheels

# Manuals

Manuals for all the hardware are located on the USB stick in the directory “docs”

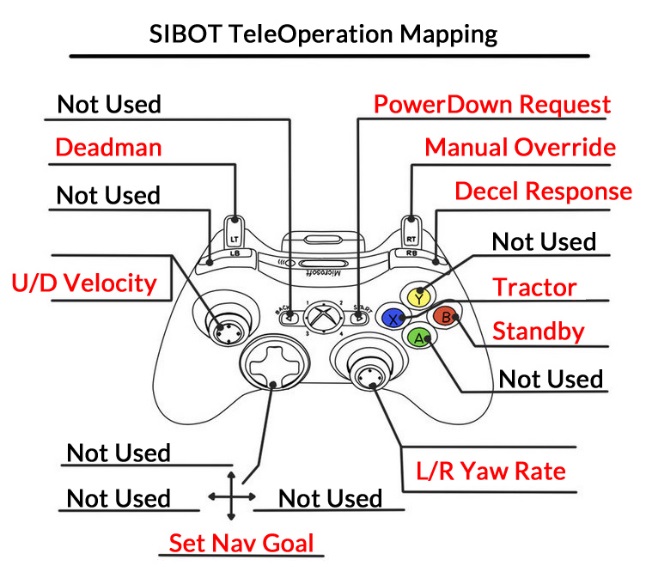
# SIBOT Overview

## Networking



This is the basic network diagram for the system. The remote PC IP address is not important but the provided virtual PC is set to this because as a remote PC on attached to a ROS Master you must define ROS\_IP as your own IP and ROS\_MASTER\_URI as the platform master PC which is sibot2. This configuration can be seen in the “manual\_pc\_setup\_files /bashrc”

## XBOX360 Controller Mapping

SIBOT has teleoperation from a joystick that is linked directly to the platform. The inputs are defined to the left.

Important: if for any reason there is a need to stop the platform there are several ways to do it:

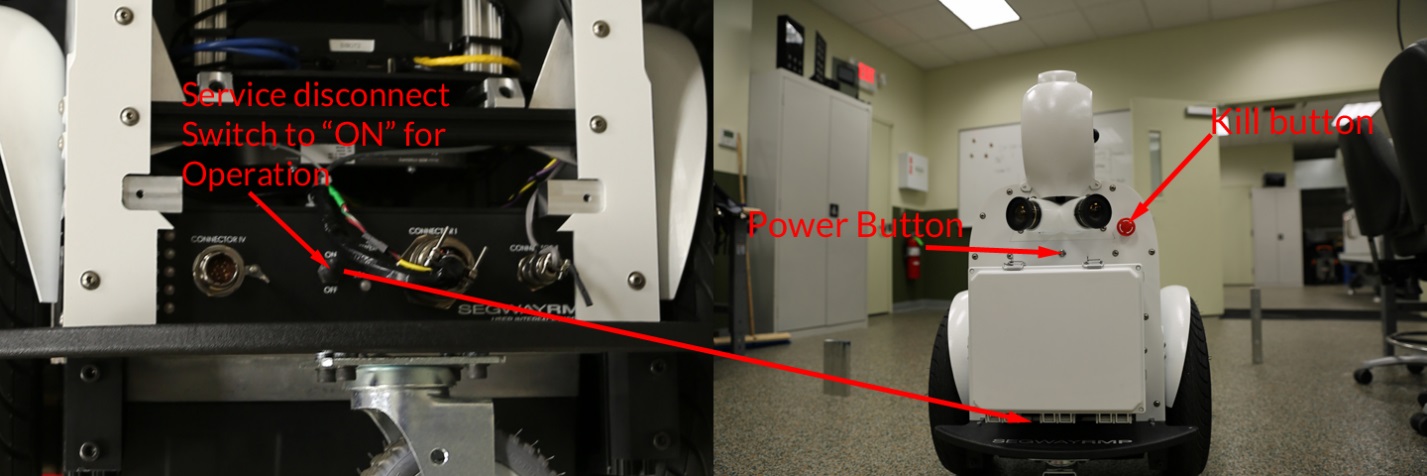
**-Temporarily:** Hit Standby button, platform will stop executing motion commands until Tractor mode is re-activated

**-Permanently:** Hit Decel Response the platform will perform a decel to zero speed safety response (DTZ) and wait until powered off. One executed the platform must be power cycled to continue.

Other functions will be described in the section below.

# Robot Power-ON and Emergency Disable

Diagram of platform switches



## Switch off the service disconnect

There is a service disconnect on the UI box inside the platform this is used to keep the platform from being powered on. It should never really be switched off unless you want to prevent the momentary power button from turning the platform on, such as in shipping. Turn the service disconnect switch to “ON”

## Hardware kill button

There is a red kill button on the platform. It is used as an emergency stop should people be near the robot in a use case and an emergency requires the user to immediately disable the platform. It opens the bridge of the motordrives, completely disables motion from the platform, and shuts the platform down. This overrides all other commands and behavior via hardware signal.

To deassert the disable signal rotate the button in the direction of the arrows and it will release if in a pressed state. To assert the disable signal press the button

**THE KILL SIGNAL SHOULD ONLY BE USED IN AN EMERGENCY; NOT AS A METHOD TO TURN THE PLATFORM OFF IN NORMAL OPERATION!!!**

## Powering on and off

The platform is powered on and off by pressing the silver momentary switch.

To turn the platform ON:

1. Press the silver power button.
2. The BLUE LED ring should illuminate.
3. The platform will initialize and you will hear an audio indicator when it is online (takes ~5-10 seconds).
   1. If it does not make sure the kill button is not pressed and the service disconnect is switched to ON

To power the platform off press the button again.

1. The platform will indicate shutdown with audio
2. The system power management will then shutdown all the components in the system , including the PCs. This will take ~30 seconds

## Charging SIBOT

Follow the instructions in the RMP manual: 

1. Plug the charger power supply box into the wall
2. Ensure the charger power supply is turned off
   1. **(NOT DOING THIS COULD RESULT IN DAMAGE TO THE HIGH CURRENT CHARGING CIRCUITRY)**
3. Connect the charger cord into the platform securing the connector IV
4. Turn on the charger power supply

To disconnect:

1. Turn off the charger power supply box
   1. **(NOT DOING THIS COULD RESULT IN DAMAGE TO THE HIGH CURRENT CHARGING CIRCUITRY)**
2. Disconnect the charger cord from the platform connector IV

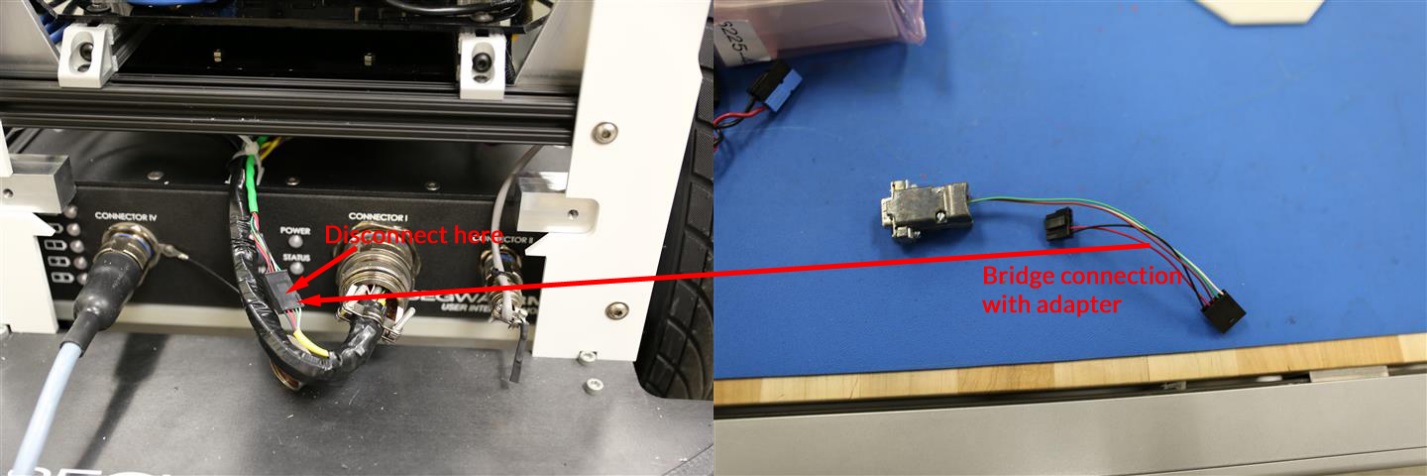
# Calibration

All of the hardware aside from the rear cameras have been calibrated. The rear cameras can be calibrated following the tutorials for monocular cameras:

<http://wiki.ros.org/camera_calibration>

The Multisense SL can be calibrated according to the manual which is located on the USB stick in the directory “docs”

The Microstrain IMU may need to be calibrated as the geomagnetic field differences in CA may be significant. To do this follow the instructions on the included MicroStrain USB stick. Before you start you need to install the Y harness that disconnects the IMU from the platform for external connection and bridges the power. Disconnect the microfit connector near the UI box on the platform and connect the external harness. Then power the platform on to power the IMU:



# Prepare a remote computer

## Using the virtual machine included

The platform shipped with a virtual machine which has been all setup to operate the platform. To use this install Oracle VirtualBox included on the USB stick and add a new machine to virtual box using the VDI included on the USB stick.

1. Install virtualbox from the installation file included on the usb stick
2. Copy the directory *stanford\_vbox* to the directory which holds your virtual machines for virtualbox. *C:\Users\username\VirtualBox VMs* on windows
3. Start virtualbox and select stanford\_vbox
4. Check all the settings in the settings tab to make sure it complies with your PC performance and that the network interfaces are correct
5. The network should be bridged to the WIFI adapter you will use for connecting to the robot

### Starting the virtual machine

Power on SIBOT by pressing the power button.

1. Make sure the host PC WIFI adapter connected to the guest bridge is connected to SIBOT the SSIDs and key is listed below
   1. **SSID: SIBOT\_WIRELESS\_5G**
   2. **SSID: SIBOT\_WIRELESS\_2G**
   3. **Key: 1505F3AE5D**
2. Start the virtual machine
3. The username and password of the virtual machine are:
   1. Username: stanford
   2. Password: Welcome00
4. Open a terminal
5. If everything worked you should be able to ping all the system components try to ping each individually (sibot1, sibot2, lms100, rmp)

You’re ready to start using the platform and can skip the next section, it is included in case you want to setup a host linux machine.

## Manually preparing a remote PC

## Installing Ubuntu 12.04LTS and ROS HYDRO

1. Download and install Ubuntu 12.04LTS on a computer or virtual machine. All development was done using a virtual machine in Oracle VM VirtualBox.
2. Install SSHD

sudo apt-get install openssh-client

1. Download and install ROS Hydro following the instructions

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main" > /etc/apt/sources.list.d/ros-latest.list'

wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -O - | sudo apt-key add –

sudo apt-get update

sudo apt-get install ros-hydro-desktop-full

sudo rosdep init

rosdep update

echo "source /opt/ros/hydro/setup.bash" >> ~/.bashrc

source ~/.bashrc

1. Add SIBOT computer to /etc/hosts

sudo gedit /etc/hosts

Insert the following lines:

192.168.1.11 sibot1

192.168.1.10 sibot2

192.168.1.12 lms100

192.168.1.13 rmp

## Setting up bash aliases for remote monitoring:

Now that you can ssh into the computers its time to setup some bash aliases to make things easier to navigate. Open a terminal and type:

gedit ~/.bashrc

Copy the lines in the file “manual\_pc\_setup\_files/bashrc” on the USB stick included

## Creating remote catkin workspace for visualizing data:

This section describes creating a Catkin workspace for the remote computer for visualizing the robot data in RVIZ and to send navigation goals and initial platform pose while running the navigation stack.

Copy the file “manual\_pc\_setup\_files /start\_sibot” to the home directory

Copy the directory “manual\_pc\_setup\_files /sibot\_remote\_ws” to the home directory

Open a new terminal and run:

cd ~/sibot\_remote\_ws/src

catkin\_init\_workspace

cd ~/sibot\_remote\_ws

catkin\_make

sws

## Powering on and connecting for the first time:

The first time you connect to the platform you need to setup the remote PC to connect to it. Start by pressing the silver button in the back, the system should initialize and you should hear the platform make a beep noise when it is up.

### Connecting to the Wireless:

The wireless is setup as a standard AP. It is dual band so there are 5GHz and 2.4GHz links. To connect choose the SSID from your wireless networking and connect to it.

**SSID: SIBOT\_WIRELESS\_5G**

**SSID: SIBOT\_WIRELESS\_2G**

**Key: 1505F3AE5D**

Make sure you can ping everything in the system by opening a command prompt and entering ***ping <device name>*** for each of the following (sibot1, sibot2, lms100, rmp)

### Force time sync with sibot1:

You need to sync your PC time with SIBOT1 to ensure that the data coming back is time aligned. SIBOT1 runs the NTP server that syncs everything in the system so the remote computer should be synced with it. Follow these steps:

1. Install chrony, this is to time synchronize the remote computer with the NTP server running on SIBOT1

sudo apt-get install chrony

sudo gedit /etc/chrony/chrony.conf

replace the lines that say “server …” (around line 19 of the file) with:

server sibot1 minpoll 0 maxpoll 5 maxdelay 8

save the file and exit

1. Force the time sync

sudo /etc/init.d/chrony stop

sudo ntpdate sibot1

sudo /etc/init.d/chrony start

### Setting up SSH Keys:

Access to the system can be a simple SSH or remote desktop. Due to the additional resources of remote desktop we have chosen SSH. You can always setup remote desktop if SSH is not adequate.

Step One—Create the RSA Key Pair

The first step is to create the key pair on the client machine (there is a good chance that this will just be your computer) open a terminal and type:

ssh-keygen -t rsa

Step Two—Store the Keys and Passphrase

Once you have entered the Gen Key command, you will get a few more questions. You can press enter here, saving the file to the user home (in this case, my example user is called demo).

Enter file in which to save the key (/home/demo/.ssh/id\_rsa):

It's up to you whether you want to use a passphrase. Just press enter for no passphrase

Enter passphrase (empty for no passphrase):

The entire key generation process looks like this:

ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/home/demo/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/demo/.ssh/id\_rsa.

Your public key has been saved in /home/username/.ssh/id\_rsa.pub.

The key fingerprint is:

4a:dd:0a:c6:35:4e:3f:ed:27:38:8c:74:44:4d:93:67 demo@a

The key's randomart image is:

The public key is now located in /home/demo/.ssh/id\_rsa.pub The private key (identification) is now located in /home/demo/.ssh/id\_rsa

Step Three—Copy the Public Key

Once the key pair is generated, it's time to place the public key on the virtual server that we want to use.

You can copy the public key into the new machine's authorized\_keys file with the ssh-copy-id command. Make sure to replace the example username and IP address below.

ssh-copy-id sibot@sibot1

You should see something like this after (type yes):

The authenticity of host '192.168.1.11 (sibot1)' can't be established.

RSA key fingerprint is b1:2d:33:67:ce:35:4d:5f:f3:a8:cd:c0:c4:48:86:12.

Are you sure you want to continue connecting (yes/no)? ***yes***

Both sibot1 and sibot2 passwords are ***Welcome00***

Now you should see, the next message asking for a password:

Warning: Permanently added '192.168.1.11'(RSA) to the list of known hosts.

sibot@sibot1's password: ***Welcome00***

Now try logging into the machine, with "ssh 'user@12.34.56.78'", and check in:

~/.ssh/authorized\_keys

to make sure we haven't added extra keys that you weren't expecting.

Now you can go ahead and log into sibot@sibot1 and you will not be prompted for a password. However, if you set a passphrase, you will be asked to enter the passphrase at that time (and whenever else you log in in the future).

ssh sibot@sibot1

Repeat the above steps starting with the command for the second computer:

ssh-copy-id sibot@sibot2

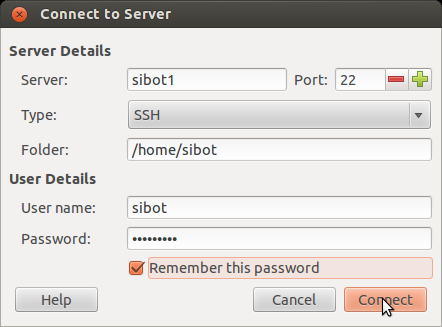
### Setting up SFTP:

To access the file system from the remote computer filesystem GUI; setting up SFTP is easiest.

Open file browser and select File->Connect to Server…

Select SSH from the dropdown.

Enter server information for each server (sibot1 and sibot2)



Once you have connected to both right click the mount point under Network and select Add Bookmark for both computers

You should now be all set to start working with the robot. The locations that will be of interest:

1. Sibot2: /home/sibot/vision\_ws/src
   1. This is the location where you will be adding your algorithms
2. Sibot2: /home/sibot/vision\_ws/src/viso2
   1. This is the location for the stereo odometry algorithm, it should only be modified if you are going to test and verify that the odometry is not affected. This is run as part of the navigation system but because it uses vision it was put on the vision PC to distribute computational load accordingly
3. Sibot2: /home/sibot/vision\_ws/src/point\_grey\_camera\_driver
   1. This is a slightly modified version of the USB3 Flea3 camera drivers
4. Sibot2: /home/sibot/vision\_ws/src/vision\_bringup
   1. Location for launch files for the vision system components. The files here are used in the system launch to start the vision components up. They should only be modified if you are sure it will not affect the rest of the system.
5. Sibot2: /home/fncs
   1. Modular startup scripts for the platform
6. Sibot2: /home/start\_sibot….
   1. System startup scripts; these will get called to start the base system completely. There is one script for each function described in the next sections

## SIBOT Functions

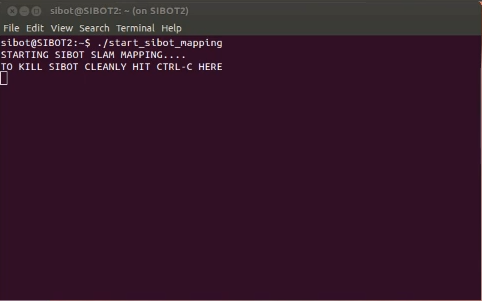
## Starting and stopping a function

Functions are started and stopped easily with initialization timing of various components handled by scripts. To start a function:

1. Make sure the XBOX360 controller is powered on the green LED around the XBOX button should be flashing.
2. Close terminals and open a new terminal with the cwd in home
3. Run the command ./start\_sibot <function> [args]
4. Allow the platform to start completely before doing anything, it takes ~20 seconds to initialize all modules
5. Once the RVIZ GUI is launched the platform is ready to run in the desired function

To cleanly stop a function and shutdown ROS and the parameter server:

1. Find the terminal that has the message “*TO KILL SIBOT CLEANLY HIT CTRL-C HERE*” and type ctrl+c



1. Close all the terminals left open and SIBOT is ready to start a new function

## Basic Teleoperation

### Command

From the remote computer shutdown any other functions, open a new terminal and type this command and hit return:

./start\_sibot basic\_teleop

### Description

In basic teleoperation all of the sensors and odometry are running but no obstacle avoidance is enabled. To drive forward hold the deadman trigger in and use the joysticks to command motion to the platform.

## Assisted Teleoperation

### Command

From the remote computer shutdown any other functions, open a new terminal and type this command and hit return:

./start\_sibot assisted\_teleop

### Description

In assisted teleoperation all of the sensors and odometry are running and obstacle avoidance is enabled. To drive forward hold the deadman trigger in and use the joysticks to command motion to the platform.

If the platform gets stuck because the inflated cost is higher than the lethal cost for moving in the trajectory commanded (ie a potential collision with the robot footprint) you can override the collision avoidance by holding both the left and right triggers (Deadman and Manual Override) to get the robot past the obstacle. This can happen for a few reasons:

1. The inflation radius in the costmap configuration is too high and a lethal cost is indicated even though it will not actually collide. The inflation radius is essentially a buffer zone for the robot when calculating the cost function for a given trajectory
2. The platform is driven too close parallel to a wall and the platform cannot safely turn in place.
   1. The footprint is defined as an elongated circle because the body and front laser protrude from the circumscribed radius. The alternative would be setting the footprint to a constant radius circumscribing the entire platform but that would affect the ability to traverse through narrow passages by increasing the width used in cost functions.
3. The platform is actually going to collide and there is no valid trajectory

Use caution when manually overriding the obstacle avoidance, the platform is completely unguarded.

## 2D SLAM Mapping

### Command

From the remote computer, shutdown any other functions open a new terminal and type this command and hit return:

./start\_sibot mapping

Once a map has been created, save the map with the automated command by opening a new terminal and typing:

save\_map <map\_name>

where map name is the name you want to reference the map by. For example a map of our lab is lab\_map. The map is saved in the correct directory for referencing for global navigation.

### Description

In mapping mode the platform is in assisted teleoperation mode with gmapping SLAM running. Simply drive the platform around to build a map. See more information on assisted teleoperation for details on the drive mode.

## Global Navigation

### Command

From the remote computer open a new terminal and type this command and hit return:

./start\_sibot global\_nav map\_name

Where map name is the name of the map in which to navigate

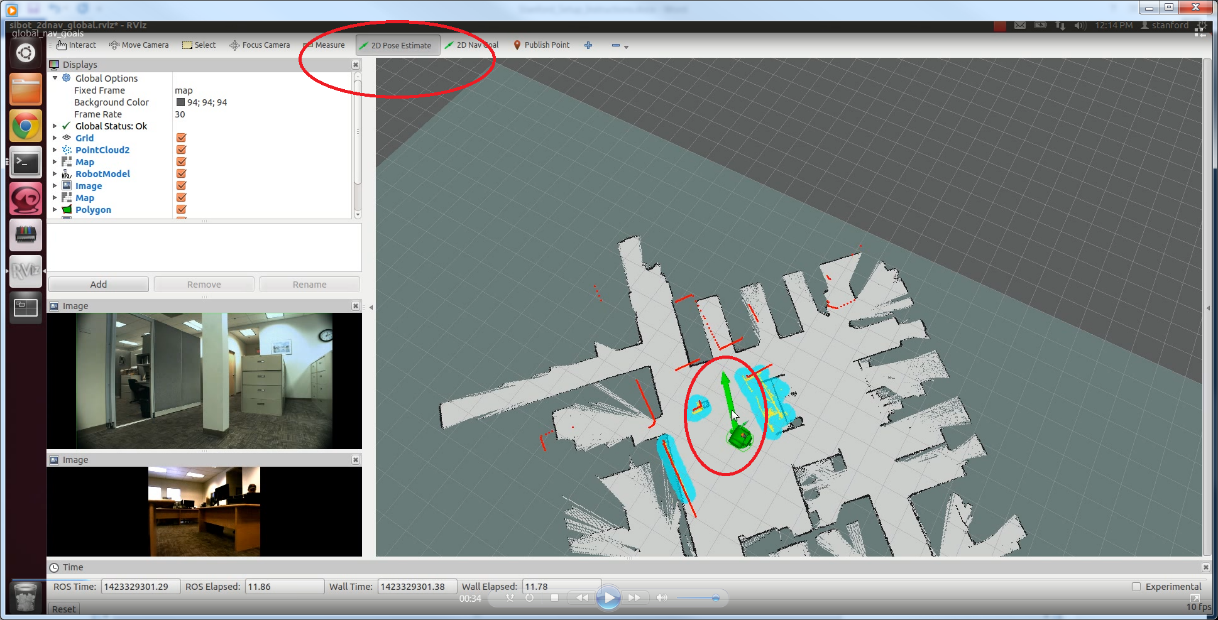
### Description

In global navigation mode the platform is using a map created to navigate globally in a static frame referenced by the map origin (/map). At startup you must initialize the AMCL global localization node. To do this with RVIZ click on the  button, left click on the point in the map reasonably close to the actual location and drag in the approximate heading. After try rotating the platform with the joystick to lock in localization. This step may be repeated if the localization does not initialize properly, you will know if the laser data in red lines up with the map.

You can then navigate by clicking  and selecting a point in the map or by writing your own script which publishes 2DNavGoals to /move\_base\_simple/goal or an action server like in the global\_nav\_goals function which sends the platform to pre-defined goals in the map. A map must be defined to use this type of navigation. If no map is defined use local\_nav instead. Goals should be published to the /map frame.

An action client collects the goals and issues them to the move base action server while watching the status of the goal being executed.

IMPORTANT: if at any time the platform needs to be stopped use the XBOX controller by pressing the left trigger or the Standby button (B) or by pulling the deadman trigger (Left Trigger)



## Local Navigation

### Command

From the remote computer shutdown any other functions, open a new terminal and type this command and hit return:

./start\_sibot local\_nav

### Description

In local navigation mode the platform navigates in the local /odom frame. This is useful if there is no map data available, or could be used to build a map via autonomous navigation in the local frame if gmapping is run separately. You can navigate by clicking  and selecting a point relative to the robot or by writing your own script which publishes 2DNavGoals to /move\_base\_simple/goal or an action server which sends the platform to pre-defined goals in the local /odom frame. Goals should be published to the /odom frame.

IMPORTANT: if at any time the platform needs to be stopped use the XBOX controller by pressing the left trigger or the Standby button (B) or by pulling the deadman trigger (Left Trigger)

## Global Navigation with Pre-set goals

As an example of using the navigation functions to move between preset waypoints the following functions can be used to set the goals and then navigate between the setpoints

### Setting Pre-set goals

#### Command

From the remote computer shutdown any other functions, open a new terminal and type this command and hit return:

./start\_sibot global\_nav\_set\_goals map\_name

Where map\_name is the name of the map in which to set the goals

#### Description

In global navigation set goals mode the platform is using a map created to set global goals in a static frame referenced by the map origin (/map). At startup you must initialize the AMCL global localization node. To do this with RVIZ click on the  button, left click on the point in the map reasonably close to the actual location and drag in the approximate heading. After try rotating the platform with the joystick to lock in localization. This step may be repeated if the localization does not initialize properly, you will know if the laser data in red lines up with the map.

The platform is in assisted tele-operation mode and the platform can be driven around, when you get to a point in the map where you want to set a goal press the Set NAV Goal (DPAD Down) on the controller.

### Navigating Pre-set goals

#### Command

From the remote computer shutdown any other functions, open a new terminal and type this command and hit return:

./start\_sibot global\_nav\_goals map\_name

Where map\_name is the name of the map in which to set the goals

#### Description

In global navigation goals mode the platform is using a map created to globally navigate between preset goals in a static frame referenced by the map origin (/map). At startup you must initialize the AMCL global localization node. To do this with RVIZ click on the  button, left click on the point in the map reasonably close to the actual location and drag in the approximate heading. Once an initial pose is received the platform will rotate to localize in the /map frame.

Once it has localized the platform will begin navigating between the preset goals, while tracking status.

IMPORTANT: if at any time the platform needs to be stopped use the XBOX controller by pressing the left trigger or the Standby button (B) or by pulling the deadman trigger (Left Trigger)

## GPS navigation

This function is not fully implemented, but is in development. It is not part of the standard navigation functionality in ROS and poses some challenges we are working on. It will function as follows and will be made available once it is stable.

1. Use Google maps to create a origin location and number of waypoints of interest and save a KMZ file
2. Software loads the user defined KMZ
3. Using GPS the platform will localize in the UTM coordinate frame with a valid NavSatFix.
4. Longitude and latitude offsets and local geomagnetic field are defined by the origin location which is used to localize the platform in the world frame.
5. A global\_nav\_goals module then reads the GPS waypoints set by the user in the KMZ file and dispatches an action client to send the platform from waypoint to waypoint evaluating progress along the way.
   1. Once a waypoint record is created this step can be repeated at will. Additional waypoints can be added.

# SIBOT Configuration Parameters

There are a number of configuration parameters that can be tuned to change the behaviour of the robot, camera system or the lasers. I will cover the important ones here.

## RQT\_reconfigure

Most of the configuration parameters can be modified live. This can be done by opening a new terminal and running the command:

rosrun rqt\_reconfigure rqt\_reconfigure

A GUI should launch that contains dropdowns for all available functions on the parameter server. Most of the documentation for these parameters can be found on the websites for the modules.

<http://wiki.ros.org/base_local_planner>

<http://wiki.ros.org/amcl>

<http://wiki.ros.org/move_base>

<http://wiki.ros.org/navfn>

<http://wiki.ros.org/gmapping>

The parameters under SIBOT contain the namespace parameters for the MultiSense SL, this had to be done in order to keep the TF tree in compliance with ROS standards.

# Modifying configurations files at launch

There are a number of modules that may need to be modified depending on the use case the ones of interest are noted here but as you add algorithms you may need others

## Odometry

Robot\_localization UKF is used for odometry. This module is perhaps the most important in the system as it generates the local odometry estimate and can run a global localization via GPS. Several methods were implemented, but a simplified version was delivered to reduce the risk of inputs to the filter not being useful for the intended environment. The files are located on sibot2 in the directory: /home/sibot/vision\_ws/src/vision\_bringup/localization. The file /home/sibot/vision\_ws/src/vision\_bringup/sibot\_odometry.launch determines which inputs to the filter are used. By default only IMU and wheel odometry are used to create a local odometry estimate. I will explain the other potential inputs and configurations here:

### Other odometry input sources

Stereo Odometry

Utilizes viso2 for visual odometry: /home/sibot/vision\_ws/vision\_bringup/localization/stereo\_odometry.launch

Use arg “use\_stereo” set to “true” when calling /home/sibot/vision\_ws/src/vision\_bringup/sibot\_odometry.launch

Was used because of the computational overhead and minor improvements

#### Laser Odometry

Utilizes 2D laser for ICP scan matching for odometry

/home/sibot/vision\_ws/vision\_bringup/localization/sibot\_laser\_pose.launch

Use arg “use\_laser” set to “true” when calling /home/sibot/vision\_ws/src/vision\_bringup/sibot\_odometry.launch

Was not used because of the dependency on a 2D environment.

#### GPS

Utilizes GPS to create a global reference in UTM.

/home/sibot/vision\_ws/vision\_bringup/sibot\_navsat\_transform.launch

/home/sibot/vision\_ws/src/vision\_bringup/sibot\_ukf\_global.launch

Use arg “use\_gps” set to “true” when calling

/home/sibot/vision\_ws/ src/vision\_bringup/sibot\_odometry.launch

Not used because it requires no other module be publishing /map 🡪 /odom transform, such as AMCL. Since this is dependent on the GPS navigation functionality it was not included. You can add GPS to the local odometry estimate by adding the following lines to /home/sibot/vision\_ws/vision\_bringup/sibot\_ukf\_local.launch

<!-- =============================================================================== -->

<!-- Configure odom1 (GPS\_ODOM) -->

<param name="odom0" value="/sibot/odometry/gps"/>

<rosparam param="odom0\_config">[true, true, false,

false, false, false,

false, false, false,

false, false, false,

false, false, false]</rosparam>

</rosparam>

<rosparam param="odom0\_differential">[false, false, false, <!-- x, y, z position -->

false, false, false] <!-- roll pitch yaw -->

</rosparam>

</node>

And include this line in /home/sibot/vision\_ws/src/vision\_bringup/sibot\_odometry.launch

<include file="$(find vision\_bringup)/localization/sibot\_navsat\_transform.launch" />

## Platform velocity and acceleration limits

Right now the limits have to be set in a configuration file. We are working on a release for the platform drivers that take the limits from base\_local\_planner parameter server to simplify. To modify the performance limits of the platform edit:

On SIBOT1 /home/sibot/nav\_ws/src/sibot\_comm/config/sibot\_config\_params.yaml

**An important note!!!!!!!!! Be careful setting the performance limits, the platform is may tip forward if the acceleration limits are set to high and not enough weight is added in the rear. CG location will dictate the maximum acceleration limit. To be cautious do not go above 1.0 (m/s2)**

**Also keep in mind the minimum stopping distance when setting the maximum velocity. If the acceleration limits are low and the velocity limit is high, stopping distance will increase for tele-operation. The trajectory planners will deal with this but it is much less intuitive for teleoperation.**

Other things to keep in mind are the relative laser rotational speed and the platform speed. Increasing speed of the platform with the same laser rotational speed will increase laser skew and accuracy of the ground plane filtering. You can modify the speed of the rotating laser either through rqt\_reconfigure /sibot or by editing this line in /home/sibot/nav\_ws/src/sibot\_startup/

<node name="$(anon dynparam)" pkg="dynamic\_reconfigure" type="dynparam" args="set\_from\_parameters sibot" clear\_params="true">

<param name="motor\_speed" type="double" value="3.14" />

<param name="network\_time\_sync" type="bool" value="true" />

<param name="fps" type="double" value="15.0" />

</node>

To update the Trajectory planner limits for navigation either use the parameter server by setting them in rqt\_reconfigure, or by modifying the file:

On SIBOT1 /home/sibot/nav\_ws/src/sibot/sibot\_navigation/sibot\_navigation\_config/config/base\_planner\_params.yaml

These are the lines of interest:

# http://wiki.ros.org/base\_local\_planner

TrajectoryPlannerROS:

# Robot Configuration Parameters

acc\_lim\_x: 0.5 #default 2.5

acc\_lim\_y: 0.0 #default 2.5

acc\_lim\_theta: 1.0 #default 3.2

max\_vel\_x: 0.5 #default 0.5

min\_vel\_x: 0.1 #default 0.1

max\_vel\_theta: 1.571 #default 1.0

min\_vel\_theta: -1.571 #default 1.0

min\_in\_place\_vel\_theta: 0.4 #default 0.4

## Navigation Stack Configuration

All of the navigation stack parameters are available on the parameter server using rqt\_reconfigure to set them perminantly you can modify the files here:

/home/sibot/nav\_ws/src/sibot/sibot\_navigation/sibot\_navigation\_config/config

Depending on the use case specific parameters are located in the function folders here:

/home/sibot/nav\_ws/src/sibot/sibot\_navigation/

## Perception Filtering

Depending on the environment the perception filtering may need to be modified such as the SAC ground plane filter parameters if you are experiencing False positives in the obstacle costmap layer. The files to modify the filters are located here:

/home/sibot/nav\_ws/src/sibot/sibot\_navigation/sibot\_navigation\_perception/ground\_plane.launch

/home/sibot/nav\_ws/src/sibot/sibot\_navigation/sibot\_navigation\_perception/config

It is likely that the ground plane detection threshold may have to change depending on the environment. It is setup for level surfaces relative to the platform pose.

# Troubleshooting

Most information on troubleshooting the ROS portion of the platform can be found on the wiki or the ROS forum:

<http://answers.ros.org/questions/>

<http://wiki.ros.org/>

For hardware specific questions please post them on the RMP forum, we will create a new page for it soon. In the meantime you can email questions to [RMP@stanleyinnovation.com](mailto:RMP@stanleyinnovation.com)