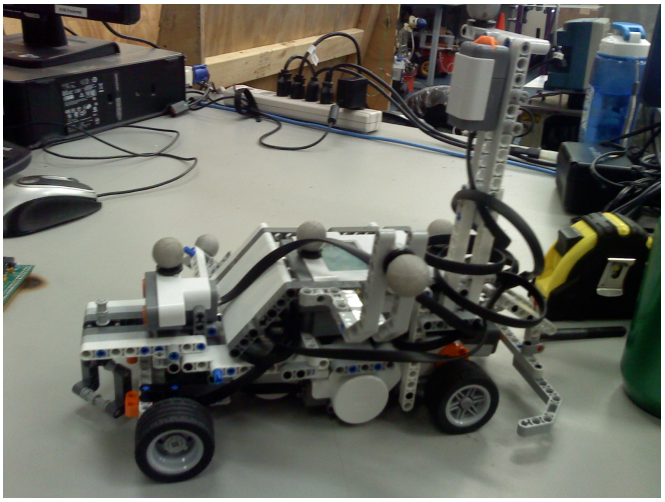
How to use the NXT with LTLMoP

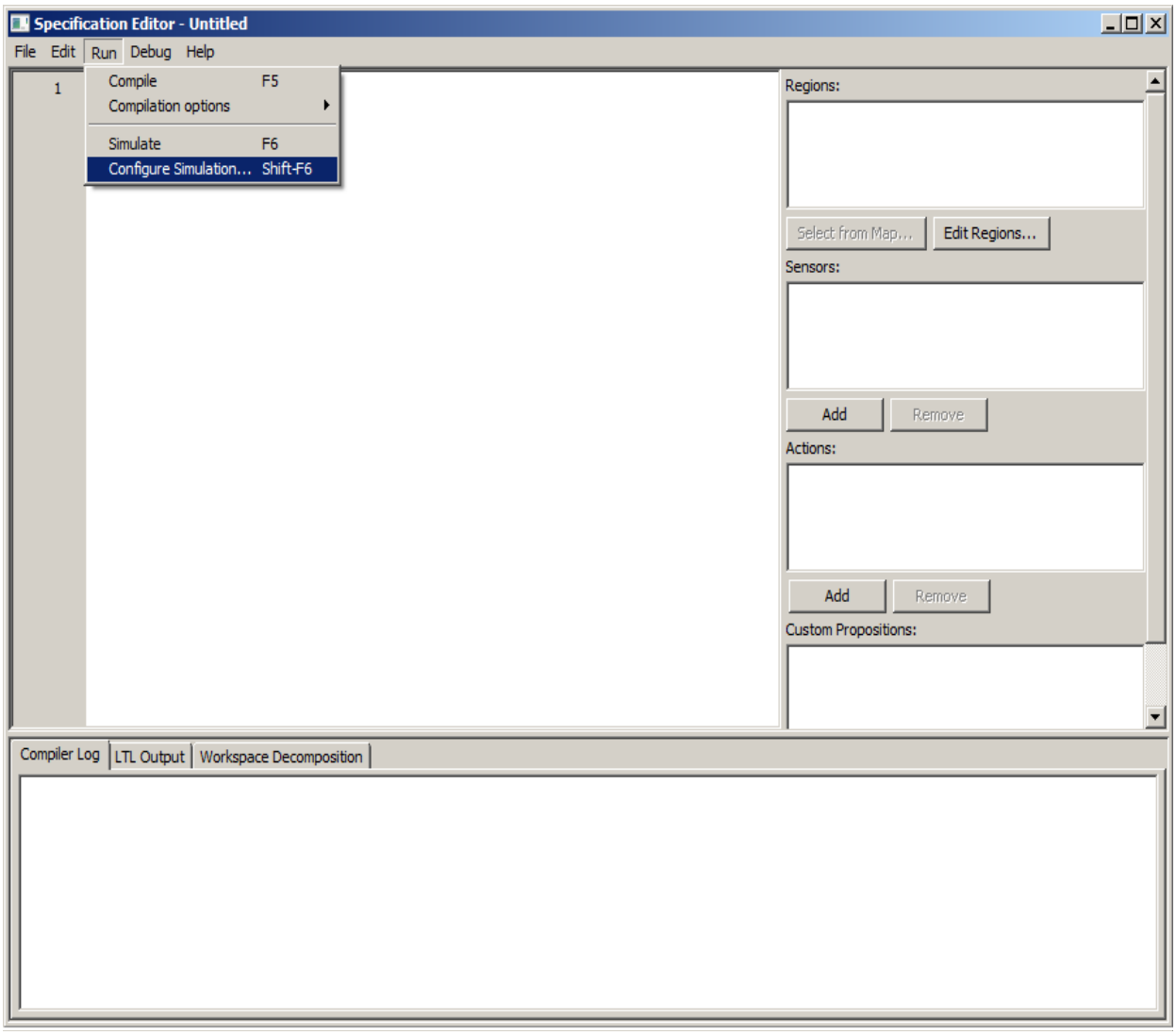
1. **Get nxt-python: http://code.google.com/p/nxt-python/**
2. **Follow steps to install (you will probably need pyUSB and pyBluez)**
    a. **The steps: http://code.google.com/p/nxt-python/wiki/Installation**

WITH VICON

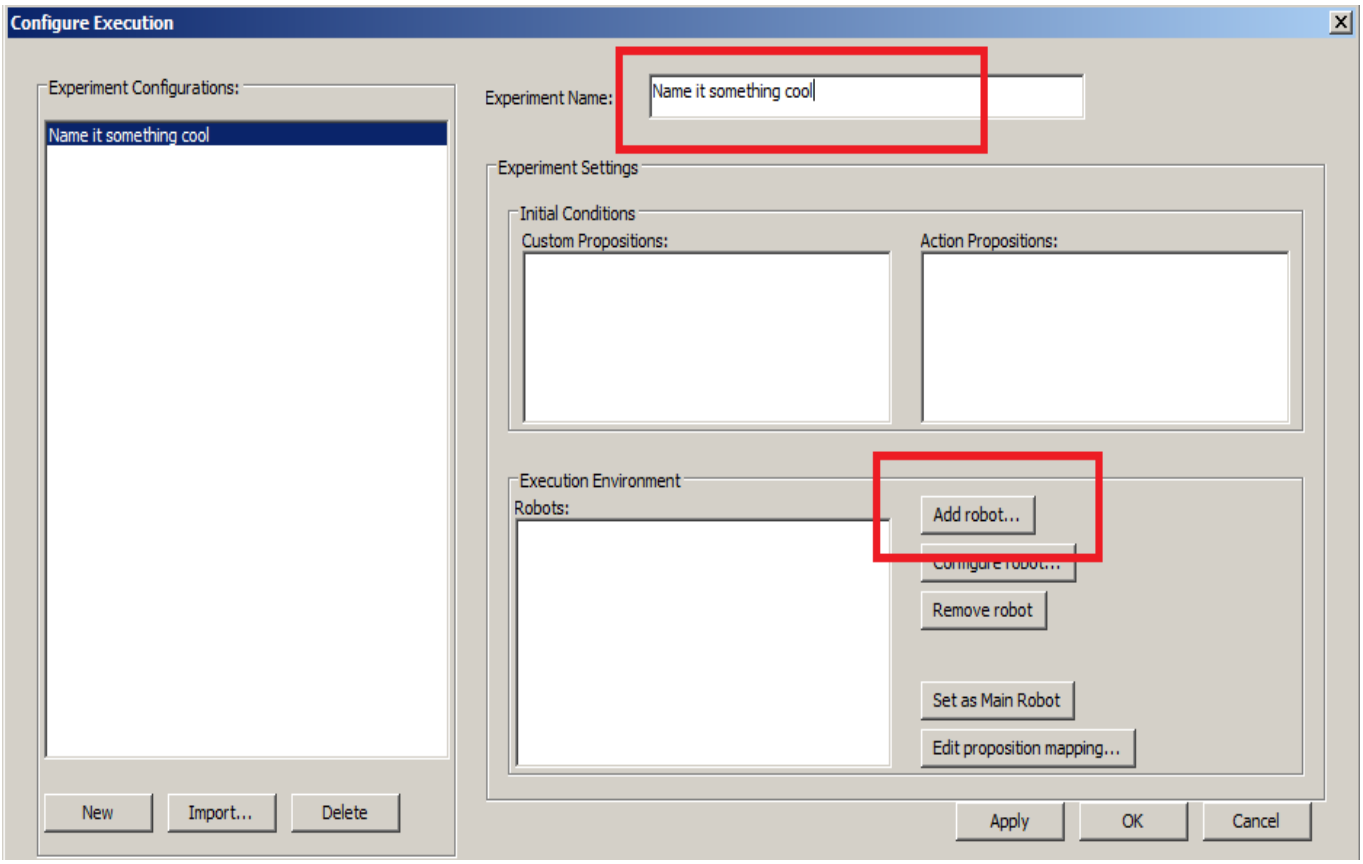
1. Make sure Vicon is on
2. If you are using the given robot design:



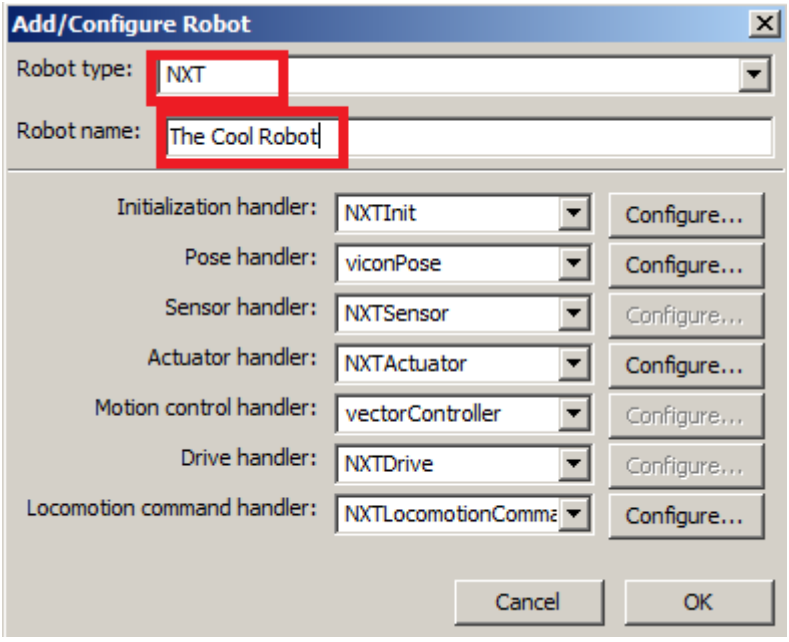
    a. Using the vicon computer, with vicon running:
        i. In Vicon subjects, open C:/Program Files/vicon/ASL data/Stephen's Data/NXT/NXT/NXT.vsk
    b. Place the NXT in the field and make sure Vicon picks up the subject
3. If you are using a **new** design:
    a. Create a Vicon subject for your design, see: http://cornell-asl.org/wiki/index.php?title=Vicon_System
    b. Place the NXT in the field and make sure Vicon picks up the subject
4. Turn the NXT on (orange button)
5. Start LTLMoP
6. Create or open a spec (see ltlmop tutorial)
7. *Run Configuration Simulation*



a. Name your config
b. Add Robot...



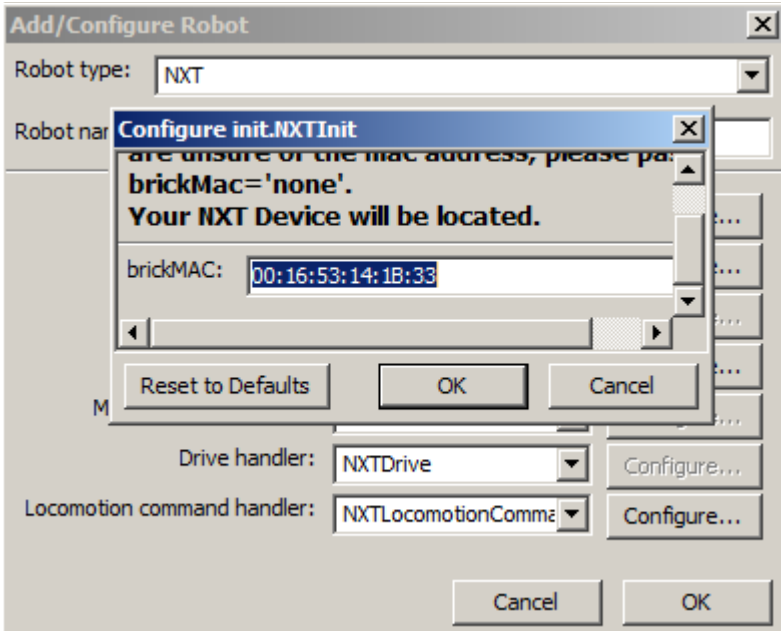
c. Select NXT in robot type
d. Name the robot



e. Initialization Handler
    i. Give the handler the NXT's mac address
        1. On the NXT, navigate to Settings (wrench)->NXT Version (NXT)

2. The digits following ID are the mac address
3. Use Colons between every two digits as seen in the image below

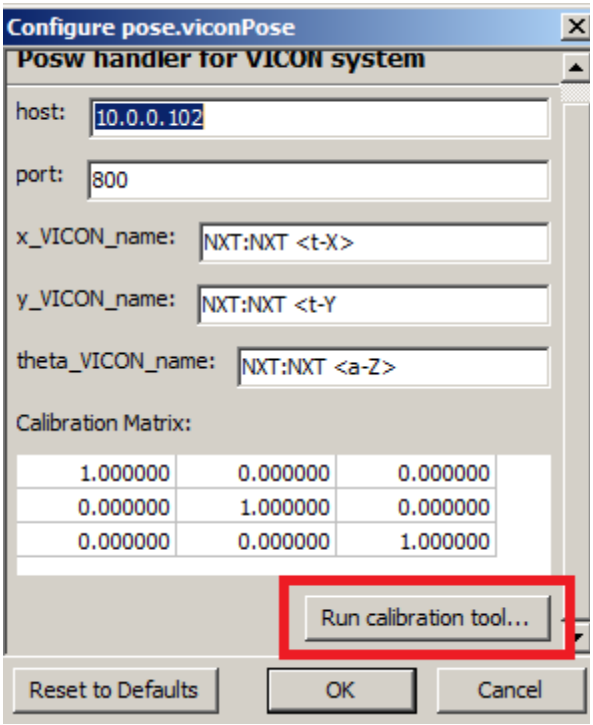
ii. Specify ‘none’ if you don’t know it, the software will search for the nxt device (this takes longer to connect)



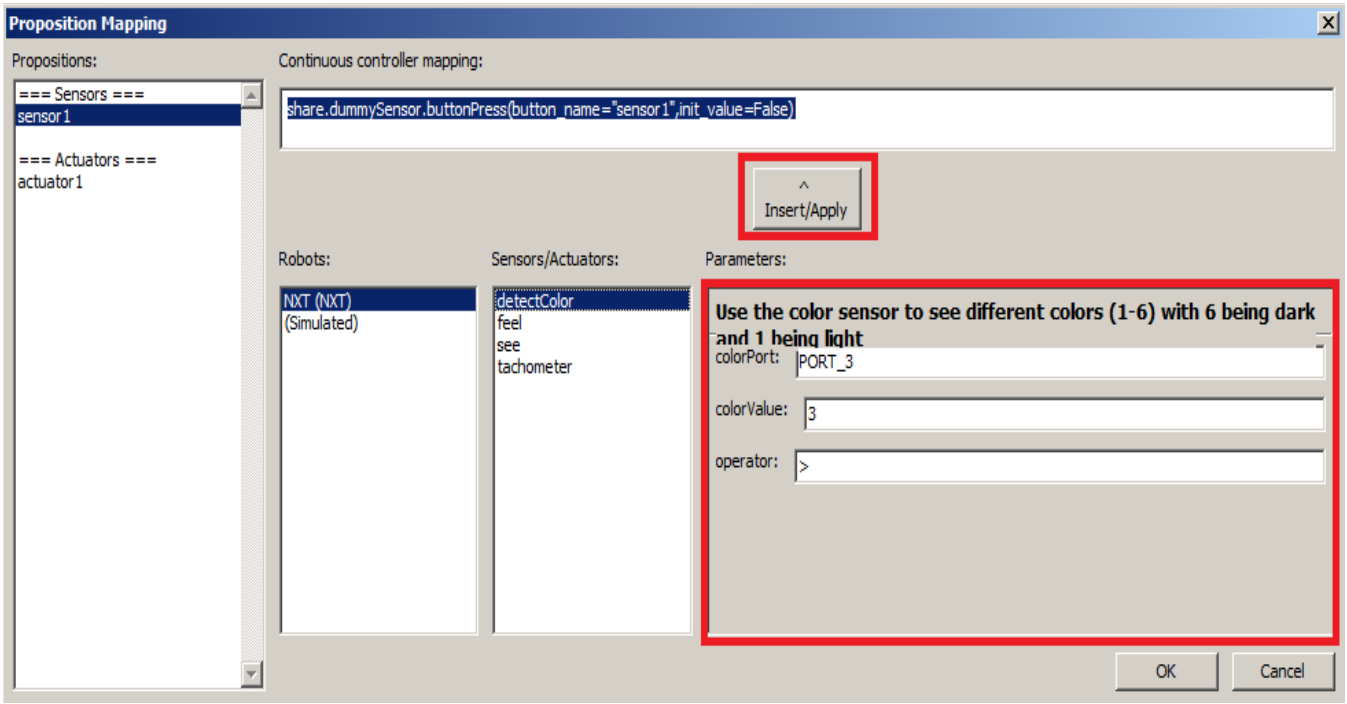
f. ViconPose (pose handler)

i. Calibrate the NXT with the field

1. On your map make sure you have selected some calibration points
2. run the calibration tool and follow the directions

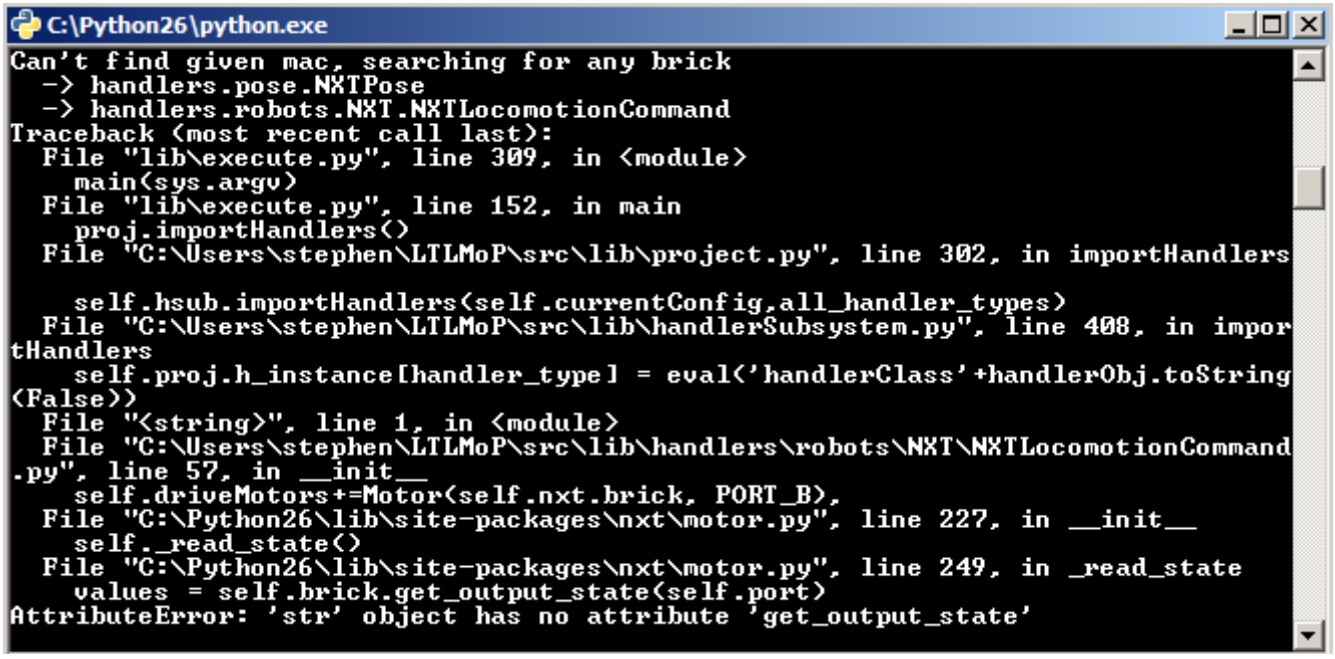


g. NXTSensor (sensor handler)
    i. See Edit proposition mappings (don't do anything yet)
h. NXTActuation (actuation handler)
    i. See Edit proposition mappings (don't do anything yet)
i. NXTLocomotion (locomotion handler)
    i. Give the port definitions for the different drive motors and steering motor if applicapable
    ii. Specify whether positive power implies forward movement on the drive motors
j. Click ok, ok
k. Edit Proposition Mappings
    i. Sensing type applications
        1. Color Sensor type propositions (detectColor)
            a. Specify the port
            b. Specify how the sensor is to detect True
            c. Color -- give it =,!= (not equal), <, or > for the following values:
                i. 1 black
                ii. 2 blue
                iii. 3 green
                iv. 4 yellow
                v. 5 red
                vi. 6 white



        2. Touch(feel) -- True is pressed, False is not pressed
            a. Specify the port for the touch sensor
        3. Ultrasonic(see) -- range is 0-255, where 25 is about a foot

        a. specify the port
        b. specify <,=,>,!= for a given value
      4. Tachometer -- range is -99999 to 99999 in degrees
        a. specify an operator like above and a value
      5. facingDirection -- range is -180 to 180
        a. specify an operator like above and a value
        b. useful for dead reckoning only, gets pose data
    ii. Assign your actions to action on the nxt. Options can be specified.
      1. **Multiple motor or sensor assignments are delimited with periods, i.e. PORT_A.PORT_B.PORT_C implies you want all three motors doing the same action**
      2. playSoundFile plays any .rso sound file on the root of the NXT. There exists online .wav converters for this file format
      3. playTones, takes a series of frequencies and durations where the nth frequency lines up to the nth duration and each tone is played in the order given. This should allow for the full specification of songs and jingles. Frequencies less than two or three hundred will default to the low value which has yet to be determined. As with motors, the frequencies and durations are delimited with periods.
  l. Click ok, ok
8. Compile and Run your simulation
  a. Compile with f5, simulate with f6
  b. On loading the simulator, it should attempt to connect to the bluetooth on the NXT.
    i. If you have an error:



      1. make sure the nxt is on
      2. make sure the computer recognizes the nxt as a bluetooth device

3. you may have to reconnect the device, this should require applying a passcode between the computer and the nxt, the default '1234' should be fine
4. You must kill spec editor and restart in order to attempt a reconnect

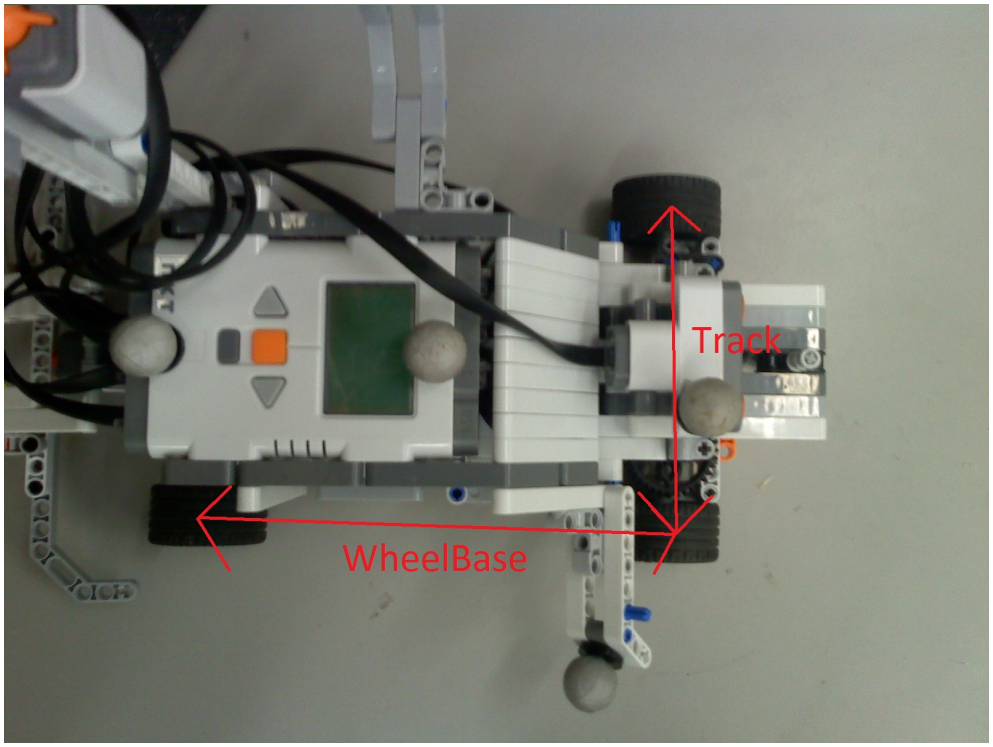
ii. If the simulation window comes up, LTLMoP has connected successfully to the NXT

c. Check again that Vicon is recognizing the NXT where you want it to be
d. Make sure if you are using a car type robot, that the wheels are oriented straight
e. Run

9. If the NXT seems to be moving strangely, then the batteries are probably dying

NO VICON

1. See steps 4-7 above
2. for 7-f:
   a. You will not be using the Vicon Pose Handler, instead select the Dead Reckoning Pose Handler
   b. Configure the NXTPose in accordance with your robot, make sure compareAgainstVicon is deselected unless you are looking to compare your dead reckoning data with vicon data. You may ignore the calibration matrix.
   c. You will want to fully describe your robot for accurate pose handling with dead reckoning. Track and wheelbase are shown below. StarX, StartY, and StartTheta specify where you will be starting your robot relative to your map.



   d. continue on g-j

3. Edit Proposition mappings
    a. Here you can specify movement as actions. Movement can be handled with actuations and dead reckoning inside the single or multiple regions.
    b. For sensors, the tachometer may be useful and can be utilized like the other sensors with values. The reading is in degrees and is zeroed when the NXT is turned on. Negative and positive values can be used with the tachometer. You may specify a value and an operator such that the sensor will return true when the value read is [operator] [value] (ex. < -25)
    c. For actuations, you have a few options for movement depending on how your spec is defined and how you have configured your robot
    Remember that multiple motors are specified with a period between ports: (PORT_B.PORT_C)
        i. arcTurn is useful for differential drive robots
            1. It is highly inaccurate
            2. It will make an arc with radius and degrees
            3. You will need to specify other variables related to your robot
        ii. runMotorAngle is useful for a car type robot but will run any motor for the desired output angle
            1. Utilizes decreasing speed to bring the motor to a stop at the right angle
            2. starts with the given power
            3. The angle is an output angle, so if you give it a gear ratio other than 1, the motor will turn more or less than your angle such that the output gear has spun the desired angle
        iii. runMotorDistance is useful for any wheeled movement
            1. you can specify the drive motors and distance
            2. The algorithm will decrease speed from the given power to the ending point
            3. The wheel radius of the nxt wheels is .0415 m and is set as the default
            4. The 3 gears provided have 36, 20, and 12 teeth. Make sure you specify the drive gears in the pose handler
        iv. runMotorTime is useful for time based situations
            1. works like the actuation, but could be used for movement
        v. turnMotorOn simply does what it says
            1. You can specify multiple motors as always
            2. Make sure that they will eventually turn off
            3. Calling a 2nd time will issue a new command, so different powers can be provided
        vi. turnMotorOff
            1. Specify all motors for any combination to turn off
            2. Safe to call even if motors are off
4. See Steps 8 and 9 above.