* 01/21/15
* Prepare ROS on ubuntu via virtualbox
* ROS (indigo, ubuntu 14.04)
  + basic commands and concept
  + browse
    - rosls, rospack, roscd +tab [package]
  + create pkg
    - roscreate-pkg
* 01/22/15
* prepare hokuyo via virtualbox
  + can’t detect serial port
  + need to set in virtual box and enable usb device(hokuyo)
  + then plug in usb after vm is fully booted up
    - tried cu.usbmodem.. not working
      * <http://stuffthingsandjunk.blogspot.com/2009/03/devcu-vs-devtty-osx-serial-ports.html>
* Problem
  + limited video memory makes simulation really hard
  + hokuyo\_node is using config in vcg format, however ROS indigo use YAML format
* Solution
  + prepare desktop for dual bootup
  + install ROS Hydro on desktop with ubuntu 12.04.
  + Done
* All ROS environment setup complete
  + basic tutorial passed, could stream/read/display data from laser scanner
    - \*.vcg config file —> .rviz (YAML for version above Groovy)
    - just look into config settings, and add a laser scan layer in rviz
    - then set subscribe topic to /scan, and fixed map topic to /laser
  + tomorrow: focus on hector\_slam\_node
    - **follow tutorial and MAP OUT!**

Hydro, 12.04 dual boot

01/25/15

* ROS
  + how to build catkin package
    - build, make, compile...
  + turtle simulation
    - use rqt\_graph to see what node subscribe/publish to topic
    - use turtle\_teleop\_key node to control movement of turtles
    - use **rostopic echo** to see realtime data stream
      * rostopic list
      * rostopic type [topic] | rosmsg show
      * rostopic pub -1 [topic] [type] ‘’ ‘’ (could use to test movement)
      * rostopic hz [topic]
      * **only one** **rostopic** **can publish to a topic**
    - use **rqt\_plot** to draw realtime plot of data published
  + communication
    - through topic
    - service
      * rosservice list/call
      * rosservice type /spawn | rossrv show
  + rqt\_console (**debug, record debugging msg**)
    - to display log/warning messages shown in runtime
    - same info output as rosrun [package] [node]
  + relaunch
    - write launch file to create 2 node simulator
    - plus a mimic node to remap input of turtle1 and output to turtle2
  + **Subscriber/Publisher**
    - publisher in python
      * initialize publisher, node, frequency (rate)
      * while loop
        + init message
        + print logging(print to screen, write to node’s log file, write to rosout), publish message
        + rate.sleep() //maintain sleep time automatically to meet frequency setup
    - subscriber
      * init node listener
      * init subscriber with callback
  + **Service and Client**
    - enable/add dependency package in package.xml
      * message\_generation, message\_runtime
    - also in CMakelist.txt
      * add message\_generation //for msg and service
      * uncomment add\_service\_files
      * add service file
* Hector SLAM
  + install ros-hydro-ros-hector\_slam, hokuyo\_node
  + nodes: hector mapping, hector\_trajectory\_server, hector\_geotiff
  + create pkg of slam\_scan
    - launch simulator with roslaunch slam.launch
  + **Q: what do we expect from hokuyo?**
    - map building, to build the road?
      * hokuyo could now build map with walls, but edge of road?
      * then what for?
      * **navigation (camera)**
    - If for ***specific use***, what is expecting to be the output?
      * map information for path generating?
      * how is the scanner gonna cooperate with other components?
    - How precise can hokuyo module with SLAM algorithm detect objects?
      * demo videos are carried out in a wide maze, with a width of 4~7 meters.
  + **Q2: Is there an overall system structure to illustrate how different components cooperate with each other?** (What functionalities are expected from each parts )
    - Hokuyo scanner
    - FRDM
    - Spark core (Wifi module)
    - H-bridge + mbed for motor control
    - Beaglebone (Linux on)
  + *MPC, PID?*
  + *Possible goal: Get a car running the verified scenario, and see whether the behavior is expected the same as modeling*
  + *Hokuyo module*
    - *object detection(car, stop sign?), range finder*
    - *find any existing packages on ROS written*
* **Friday:** map out (houses, obstacles) map out then localization
  + *at least the standard of* <http://mailmanforpenn.blogspot.com/2013/12/ese519-final-project-lidar-mailcar-team.html>
  + no need to worry about obstacles though
    - F1 tracks have barrierss

01/27/15

* Experiment
  + √the edges could be detected
  + sometimes the angle measured is wrong
    - the actual angle turned is 90 degrees but the pos on the map turns less than 90 degrees(let’s say 60) thus the output roadmap is not perpendicular to each other
    - 
  + see what parameters measure the intrinsic turn (API to find)
    - <http://wiki.ros.org/hector_slam>
    - may need encoder? for **odometer** input
      * In a typical setup the odom frame is computed based on an odometry source, such as wheel odometry, visual odometry or an inertia measurement unit.
    - may need to change the launch files
    - may need to change the frames in rviz
    - may need to refer to how the guy scan the maze

01/28/15

* Turning problem to fix
  + coordinates transform between laser & base\_link
    - <http://wiki.ros.org/tf#static_transform_publisher>
  + parameters to tune for hector\_slam.launch
  + **FIX the LIDAR, no roll/pitches**
    - stuff sponge pieces under the laser scanner
    - **better! smaller roll/pitch, barely any yaw**
    - **still have problem fanning in/out in the hallway**
    - better start facing something plain, clear edge to avoid infinite measurement
  + What’s the format of Laser msg?
    - <http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html>
  + 
* **Next steps (tomorrow)**
  + **Note: current LIDAR needs to move smoother and slower**
  + tuning parameters?
    - <http://wiki.ros.org/hector_mapping>
    - "scan\_subscriber\_queue\_size" value="25"
    - map\_update\_angle\_thresh
    - map\_update\_distance\_thresh //need to be above jitter (0.2m, 0.4 rad?)
    - dead\_zone V: laser\_min\_dist <= 0.4(default)?
    - laser\_z\_min\_value, max (valid data points)
    - seems working √ 01/29/15
      * **tuning map\_update\_angle\_threshold (avoiding too much overlapping of maps)**
  + **Pose estimation** problem
    - add in odo\_frame (odo solution)
      * see if the quality of map is better
    - How does an **odometer** works with ROS?
      * add an odom frame, but what react to the frame
      * any existing \*.msg containing convention?
      * how does an encoder work?
      * **IMU**
      * problem:
        + **yaw**: slow and smooth won’t affect, however spinning too fast(~ 0.5 s, not at constant speed) will cause visible data loss, thus estimate of angle position might be wrong. add in **odometer** with higher acquisition frequency(~30Hz perhaps) might compensate the situation.
        + **roll/pitch**: since algorithm will recognize LIDAR’s status as stable/static by default, distance information will be wrong(laser), then angular position might be wrong, small obstacle encountered might cause big problem.
        + Corridor:

can’t distinguish stable/move straight forward without knowing odometry info.

* + **MOVE**
    - <http://wiki.ros.org/ros_control>
    - ROS control mbed to move the car
      * get it moving using ROS
        + ROS control over **chassy**? (/another package)

cmd\_vel? left/right/front/back

motor controller

* + - * + spark core + mbed?
      * then obtain encoder data and see how to integrate it into the perception system(publish data to odom frame)
    - ***Steps to follow***
      * **√**[hardware connection] Mbed FRDM —> H-bridge —> motor (message:%3CCAC85EqiskyvTjxkYrdaHj\_\_bW=6gxKB0NbeR5hzF7Er\_\_zQ9gQ@mail.gmail.com%3E)
        + **√**make sure understand how actuation works
        + **√**how control is applied
        + **√** how am I supposed to know that the H-bridge is working?

first row GND

2nd row 6V (controlled by regulator)

Switch

1 is for 6V

4 is for 5V

2，3 for 2 motors (left, right)

all off to save battery

Enable

* + - * [ROS control] Laptop (mimic beaglebone) —> Mbed
        + <— —>
        + talk to mbed through serial port, send a char (wsad)

ttyACAM1?

* + - * + receive encoder information by parsing serial output?
      * Wifi connect beaglebone and laptop, remote ssh (might need to build master and robot/ distributed system), make it move **without LIDAR**
      * Get encoder information and try integrate to map/localization
        + try integrate LIDAR to it
      * Path following (how a sequence of action is applied)
        + manually
        + automatically

**FSM?**

* + "Dual ROS” (distributed levels of system, another option)
    - <http://wiki.ros.org/ROS/Tutorials/MultipleMachines>
    - Host desktop, daemon on laptop/beaglebone
    - How they two communicate?
      * goal: simulation/rviz on desktop
      * perception/realtime control on laptop (must connect to hokuyo module physically)
        + need to change code for Publisher (on embedded device) and Subscriber(on desktop)
        + reason:

help share CPU workload of simulation/mapping

**Planner:** multi control over several autobots

can ROS work on mult-bots?

* + **Discussion topics**
    - **Goal: Fix scenario: highway lane change (overtake, change lane)**
      * OODA (Observe, Orient, Decide, Act) Orient and Decision part might need to interact with the **PLANNER**
      * Simulator generate scenario —> **DEMO: Enter scenario setting first**
      * Observe
        + Perception data needed

mapping, localization

distance(**how to decide it’s an obstacle object**), LIDAR could do

velocity(can LIDAR do that with ROS?)

* + - * **Orient**
        + Analysis (ETA, estimated processing time)
        + ***[might not need now] new information(detect whether new scenario)***

*for demo use, just execute scenario*

* + - * + previous experience?

don’t know how to take advantage of this experience

machine learning? might be another whole new topic to learn… Stanford Andrew autonomous helicopter

* + - * **Decide**
        + Decision of whether to take action, then path planning, hypothesis
        + Steps to take (perhaps on local car)
      * Act
        + given path, how to reach to destination?
        + execute moves
    - **structure/architecture (how my part interact with the modeling language?)**
      * Hardware:
        + something to detect velocity of another car (LIDAR?)
        + supersonic sensor for distance(LIDAR?)
        + camera (might get complicated)

augmented with distance information?

* + - * + hall effect sensor? ()
      * Software:
        + design of interface
        + base: FSM?
      * Reminder:
        + Design should cover how do we use the demo?

demo a scenario requires same path planning on both simulator and real testbed, require they act in the same way

simulator: virtual car

testbed: real car

enter scenario, then react, does it mean that the set path order comes from a higher level planner running on ROS

verify the safety of modeling + same behavior => equally safe

* + - * + Realtime, continuous OODA loop
    - Possible difficulty:
      * Can hector\_slam **spot moving objects**?
        + static obstacles are fine, with proper map generated, scenarios could be carried out.
        + how to get possible velocity of a moving object?

online SLAM is used for localization, pose & kinetic parameters

SLAM also will possibly treat the car as solid wall, but the speed of a wall..?

any package on ROS for LIDAR to detect moving object?

or do we also need to add a supersonic sonar sensor

* + - * + what about **moving objects**?

online slam?

//can we still use the former map, or a combination of both realtime and offline map?

it could detect a difference during mapping, may need to define range of detection for online SLAM to get realtime map, then could detect/recognize specific scenario by planner, generate solution, then execute

how much time it’ll take to make a decision?

how fast can system react before it’s too late? (scenario planner should also calculate estimated time to carry out the solution and the **ETA** of a possible obstacle)

* + - Goal2: stop sign
      * OODA...

**Other:**

* + general ROS tutorial: <https://support.clearpathrobotics.com/hc/en-us/categories/200165835-ROS>
  + **blinking LED for left/right turn**
* Localization
  + After integrating encoder, figure out how localization (localizer) works in hector\_slam
    - current knowledge just using obstacles
  + AMCL?
  + OMPL (ROS)
* Navigation
  + Path finding A\*
  + ROS Navigation Package

**02/03/15**

* Tuesday Meeting

**Huge steps this semester**

1. dumb bot

2. perception/motor testing

3. platform, control/ build robot city

**Aim: "Trueman show" :** add realtime feedback during automation

**Further plan beyond next week**

1. Line following capability: to calibrate straight line, not trivial: getting it run in a line
2. Hall effect sensor & magnet: **GPS**

**Hardware**

1. camera

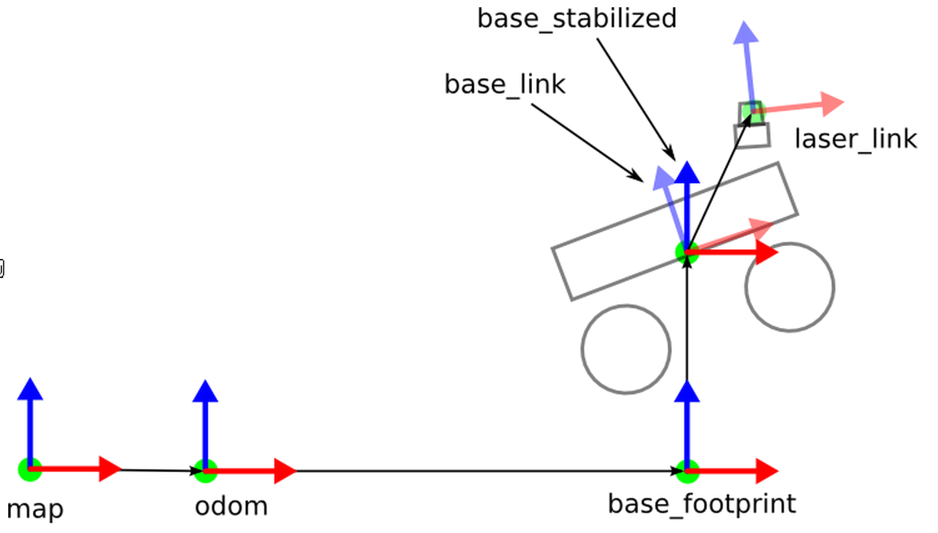
2. PIR sensors, infraed sensors (paints)

3. hall effect?

spark.io

Today

* ROS control over the car using arrow keys
  + reference to mail car
  + **!! need calibration (going straight —>** hall effect, PIR sensors**)**
* ROS & encoder to odom



* base\_footprint frame
  + 2D pose (position and orientation)
* base\_stablized frame
  + **height** relative to map/odom
  + if no pitch/roll, this frame could be same as base\_link
* base\_link frame
  + attached to robot
  + adds roll/pitch information according to stablized frame
* **laser\_link**
  + transformation from base\_link to laser\_link is typically provided by static transform publisher

Reference URLs

* TF: <http://wiki.ros.org/navigation/Tutorials/RobotSetup/TF>
* <http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom>
* <http://wiki.ros.org/odometry_publisher_tutorial>

Realization of ROS control:

Option 1:

mbed <--> handler(on embedded platform) <--> ROS

**mbed:**

mbed does little calculation, handle commands as interrupts through serial port;

print encoder information on serial port (at specific frequency)

could also print estimated velocity & position

**embedded platform:**

write python/c++ program to read data from serial port

parse and obtain encoder data (could use python code to parse)

write publisher to ROS, subscribe instruction from ROS and send to mbed through Serial port via interrupt

Option 2: (may prefer)

mbed < -- > ROS (through serial port) using rosserial\_lib

**mbed:**

obtain raw data and publish to ROS

subscribe instructions directly from ROS

more computation on mbed (might help load off some burden of embedded platform)

easier to maintain and add in new sensors

Steps to achieve:

1. enable mbed < -- > ROS

could check encoder position and velocity information from rostopic echo

<https://developer.mbed.org/users/nucho/code/rosserial_mbed/>

1. manual control over mbed via ROS package
2. Integrate encoder info to odometry frame.

<http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom#Publishing_Odometry_Information_Over_ROS>

02/05/15

Steps to achieve:

√ enable mbed < -- > ROS

could check encoder position and velocity information from rostopic echo

<https://developer.mbed.org/users/nucho/code/rosserial_mbed/>

## need to use up-to-date rosserial\_mbed\_lib

<http://developer.mbed.org/users/jjzak/code/rosserial_mbed_lib/>

## comment out UART0,1,2,3 for LPC17xx (FRDM KL25Z board doesn’t need these)

install rosserial

<http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup>

has to be rosserial hydro to talk with mbed with rosserial hydro, no mismatch allowed

√ manual control over mbed via ROS package

add in keyboard\_control.cpp

add dependency in package.xml

add dependency and build executable in CMakelist.txt

<http://wiki.ros.org/ROS/Tutorials/ExaminingPublisherSubscriber>

modify launch file to add python executable serial\_node.py (rosserial

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

**Problem:**

unsync..

keep warning about possible mismatch of version between mbed and ros

Solution:

**try sample on LPC1768**

**reason: rosserial\_mbed\_lib is adapted for LPC17xx series**

**remember to change Stream.h (private to public or comment the constructor)**

02/06/15

Try tutorial sample on LPC1768

Problem: protocol version of client (mbed) is Rev 0(rosserial 0.4 or earlier)

rosserial on server end is rev 1, rosserial 0.5+

Solution:

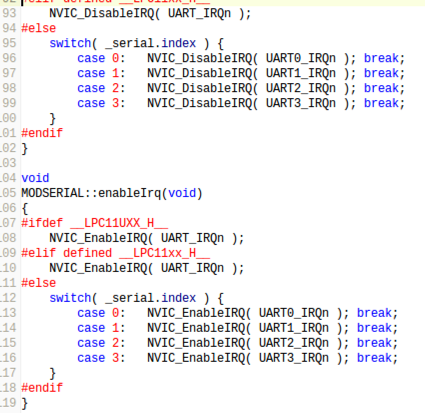
**TROLL! default baud rate is 57600, use this rate for hydro communication.**

**Now keyboard could control mbed via ros topic /cmd\_vel.**

**mbed could subscribe /cmd\_vel and react.**

**mbed could publish to new topic :) (e.g. encoder)**

Next move: try if same solution could work on FRDM KL25Z. (won’t work.. seems that the library is designed for LPC17xx)

* in order to compile the code on KL25Z, some lines in MODSERIAL library need to be commented (IRQ related)  
  
* However, if these lines remain commented, same code won’t work on LPC1768. Thus these lines are essential/critical to get rosserial running on KL25Z. (lead to a dead end to KL25Z..)

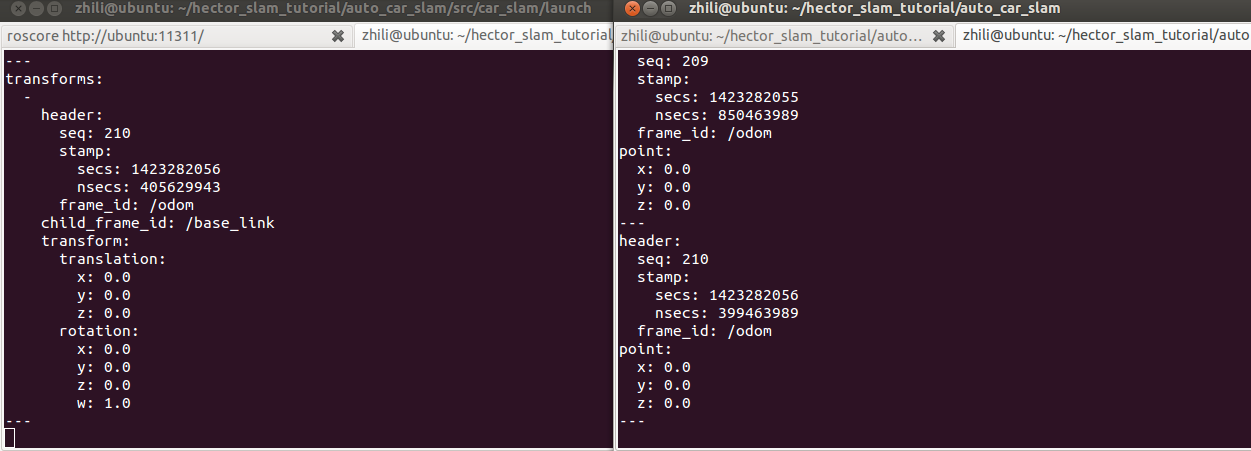
Could remap the pin layout to LPC1768 will still work.

Finished.

* **Integrate encoder info to odometry frame.**
  + **ref:** <http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom#Publishing_Odometry_Information_Over_ROS>
  + <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29>
  + **add encoder\_odom\_tf.cpp**subscribe encoder topic from mbed on ROS with callback function readEncoder  
    readEncoder will calculate x position, y position, and theta turned  
    then broadcast odom\_tf (TransformStamped msg) to base\_link through broadcaster

need to modify package.xml to add build/run dependency

need to add executable build information to CMakelist.txt  
 modify hector\_mapping.launch file to add odom frame as /odom.



**left: odom\_tf TransformBroadcaster right: encoder output from mbed**

\*\* could publish to nav/Odometry

Question: how to differentiate forward/back simply from encoder information?

* **Experiment:** 
  + Goal: check if transform from encoder\_odom to base\_link is correct

**(currently left=y pos, front=x pos)**

**Procedure:**

* + roscore
  + source devel/setup.bash
  + sudo chmod a+rw /dev/ttyACM0 //hokuyo
  + sudo chmod a+rw /dev/ttyACM1 //mbed
  + rosrun car\_slam keyboard\_control
  + rosrun car\_slam encoder\_odom\_tf
  + rosrun car\_slam rosserial\_python serial\_node.py
  + **rosrun tf view\_frames (print out frames for mapping)**
  + Check topic data
    - rostopic echo /car\_slam/encoder
    - rostopic echo /tf // this tf contains all transform broadcasters, including static ones from laser to base\_link
* result:
  + seems that the frame of scanmatcher\_frame is the one that trajectory server is using, odom tf does not affect the trajectory, perhaps the parent-child relationship is not correctly set
    - one child could only have one parent
    - **maybe adjust the level of priority?**

**Things to make sure:**

1. calibration
   1. width in between wheels
   2. ticks per meter
   3. currently motor control is similar to pwm (can control % time on/off)
2. how to distinguish back/forward from encoder?

by command

also consider the effect of latency (use former command direction)

forward: +

reverse: -

Problem met:

1. chassis won’t run..?

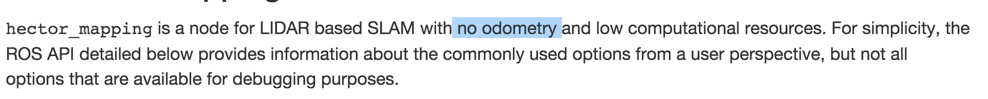
h-bridge output is fine, but the motor won’t start..

2. adjust layer of priority in ros frame  
 <http://www.ros.org/reps/rep-0105.html>

The transform from odom to base\_link is computed and broadcast by one of the odometry sources. The transform from map to base\_link is computed by a localization component. However, the localization component does not broadcast the transform from map to base\_link. Instead, it first receives the transform from odom to base\_link, and uses this information to broadcast the transform from map to odom.

set pub\_map\_scanmatch\_transform to.. false?

how to deal with pub\_map\_odom\_transform?



Odometry could thus be thought of more as an hint, directing a scan

matching procedure to a region of valid scan matches, and some SLAM

algorithms can produce accurate results without odometry at all

Odometry frame non-static:

<http://answers.ros.org/question/10302/the-tf-between-map-and-odom-is-not-static/#15185>

* Parameter change

<param name="map\_multi\_res\_levels" value="2" />

what this does is performing the registration on courser grained maps in parallel to your original map resolution. The purpose is preventing to get stuck in local minimas, which is whats happening in your case.

**Conclusion:**

* it seems that the odom frame is only used as short-term reference, and the mapping node will calculate the difference between laser data and odometry data, then publish /map → /odom tf.
* After setting pub\_map\_odom\_transform to true, the system will calculate the error between odometry data and laser scan data, then publish the difference in coordinates discontinuously.
* After proper setting up launch files, it should be up and working. Need to figure out after we could drive car on Beaglebone black.
* **need to use base\_footprint? (try without base\_footprint on car first)**

3. ROS on beaglebone black

make the chassis go straight! (use encoder data)

may control speed?

PID controller?

A summary on how hector\_slam works, from a tf point of view

1. how “slam\_out\_pose” topic is published
   1. Kalman Filter for pose estimation and trajectory planners

02/10/15

* add in base\_footprint layer (try during runtime)
  + ? tune whether to use scanmatcher\_frame as trajectory source
  + ? tune whether to use base\_link as trajectory source (not applicable, turn black..)
* fix chassis motor problem
* work on beaglebone black
  + wifi connection
  + install ros
  + install hector\_slam
  + install custom packages and nodes
  + run the car!
    - calibration
      * width in between wheels
      * ticks per meter
      * currently motor control is similar to pwm (can control % time on/off)
* navigation
  + test accuracy of odometry input

02/11/15

* work on beaglebone black
  + download image file (ubuntu 12.04
    - <http://rayhightower.com/blog/2014/01/02/beaglebone-black-ubuntu-part-1/>
    - <http://elinux.org/Beagleboard:Ubuntu_On_BeagleBone_Black>
  + partition and expand the disk
    - <http://elinux.org/Beagleboard:Expanding_File_System_Partition_On_A_microSD>
  + share internet connection with mac
    - <https://groups.google.com/forum/#!topic/beagleboard/5yoQWMNfvH4>

02/13/15

* still could share internet connection via USB Serial
  + Linux, WIndows, mac could not detect the board
  + perhaps drivers on BBB is not set up correctly
  + perhaps something outdated with the img
* use a wifi dongle instead
  + upgrade the OS (12.04 precise) hope things work out
  + share wifi connection via laptop (BBBNet; 12345678)
* Install Ubuntu-desktop
* Install ROS hydro
  + “transplant” custom packages
  + install hokuyo\_node
  + install hector\_slam
  + may need to check prerequisite for pyserial?
  + <http://wiki.ros.org/hydro/Installation/UbuntuARM>
  + <https://groups.google.com/forum/#!topic/hbrobotics/Pb_uYDiI7Dk>

√ sudo apt-get install ros-hydro-navigation

√ sudo apt-get install ros-hydro-rosbash

√ sudo apt-get install python-rosinstall

sudo apt-get install python-rosdep  
sudo rosdep init  
rosdep update

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

sudo apt-get install python-rosinstall

* Get to know Navigation
  + run the car!
    - calibration
      * width in between wheels
      * ticks per meter
      * currently motor control is similar to pwm (can control % time on/off)
  + navigation
    - test accuracy of odometry input
    - <http://www.openrobots.org/morse/doc/stable/user/advanced_tutorials/ros_tutorial.html>

02/16/15

* successfully port software on mac to BBB.
* still need to figure out wifi issue, not automatically up during boot, and occasionally offline..
* ros broken package.. happens when install ros packages indivisually

Try to Solve:

<http://answers.ros.org/question/79166/unresolvable-dependency-conflict-in-ros-hydro-desktop-ubuntu-raring/89621%3C/p%3E>

No luck.

Can’t get rviz installed due to package dependency error.

Need to reinstall… Try to remove ros packages from ubuntu and reinstall.

Not all packages are available for armhf, especially the desktop packages

* Solution:

To run rviz on remote end, and data acquisition on BBB

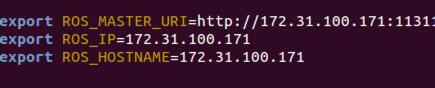
<http://wiki.ros.org/ROS/Tutorials/MultipleMachines>

<http://answers.ros.org/question/115306/how-to-solve-tf-problem-for-hector-slam/>

02/22/15

* get ros ready on BBB
* Start building distributed ROS over laptop VM and BBB
  + problem: connection is not bi-directional
    - VM could ping BBB, but not the other way around
    - try directly using the wifi-dongle, without using the wifi hotspot
  + bootup stuck at “Wait for Network interface configuration”
    - solv: problem specific to ubuntu 12.04, comment out “auto” other than lo in /etc/network/interfaces; reboot
    - use network manager to fix stuff other than local network
      * install java via software center
      * connect to AirPennNet-Help to get certificate
  + Time discrepency
    - <http://wiki.ros.org/ROS/NetworkSetup>
    - <http://wiki.ros.org/ROS/Tutorials/MultipleMachines>

02/24/15

* reason for distributed might not be working on different version of ROS
  + different stub library disallow message unpacking on server
  + lose sync
  + different version of server binary (rosout)
* connection problem solved in another way
  + use AirPennNet-Guest
  + VMWare Network Adapter option choose Bridged over NAT
  + ubuntu VM will get a real IP, seen as an additional computer on the network
* **talker and listener works!**
  + **setting:**
    - **Under AirPennNet-Guest**
    - **host:** 
    - **client:**
      * export ROS\_HOSTNAME=158.130.171.229
      * export ROS\_IP=158.130.171.229
      * export ROS\_MASTER\_URI=<http://172.31.100.171:11311>
  + 2nd try:
    - turtlesim is always a good start :)
    - <http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics>
* H-bridge
  + <http://www.criticalvelocity.com/products/datasheets/cvhb401.pdf>
* LIDAR: <https://xv11hacking.wikispaces.com/LIDAR+Sensor>

02/25/15

* car could run with **mbed connected to VM**
  + ⬆⬇⬅→ controlled by ticker, triggered every 1 s, continue for 0.5s for now
  + encoder is working properly
  + connection problem fixed, so many wires :)
* connect **mbed to BBB,** with roscore running on VM (Mac)
  + met rosserial error
    - <http://answers.ros.org/question/199418/problem-in-running-rosserial-on-raspberry/>
    - sudo apt-get install ros-hydro-common-msgs
* new problem: left wheels won’t fire..
  + always check connection(hardware/software)
  + always check serial port status
* try to go wireless (without adapter cable)
  + <http://www.element14.com/community/community/designcenter/single-board-computers/next-gen_beaglebone/blog/2013/08/10/bbb--rechargeable-on-board-battery-system>
* running experiment on BBB
  + “look up may require exploration to future“
  + update time: ntpdate -u 0.pool.ntp.org

02/26/15

* problem:
  + 1. hokuyo serial port could lose rw access
    - mbed could lose sync too.. don’t know why
    - use crontab to run this rapid check to give rw access
      * serialcheck.sh (effective!)
  + 2. wifi lose sync
    - wait a while will be fine, but need bg code
    - sol:
      * bg script:
      * <http://alexsleat.co.uk/2011/01/09/a-more-elegant-solution-to-ubuntu-wi-fi-reconnecting-issue/>
      * just use crontab to automatically reconnect wifi if lost, call the wirelesscheck.sh
  + 3. driver on mbed seems to have mutex conflict issue
    - ! try using mutex for encoder\_to\_tf
      * **if encoder\_to\_tf program doesn’t run, the control is robust and responsive for either combination of actions**
      * **problem: power on → left wheels go backwards**
        + **solve: connection, p19 unstable, should give in2 low, but disconnected and leave it high, thus left wheels go backwards**
    - ! try simplify the code by applying simple wait instead of ticker
  + duplicate! during ping
    - router will solve this problem?
  + an experiment script is written to boost the speed of program execution
    - experiement\_init.sh
    - running in background: serialcheck.sh, wirelesscheck.sh
* Tomorrow
  + power supply to run on the ground (or long wire)
  + layout of the car (how to make the LIDAR stable)
  + integrate odom + IMU

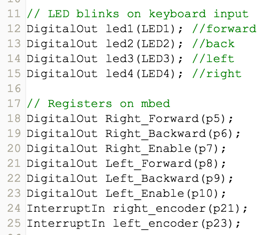
02/28/15

* battery bootup
  + bought <http://elinux.org/AndiceLabs:PowerBar> as instructed
  + might have something to do with uboot.img
* wifi
  + fix boot up wifi to BBBNet
  + <http://www.howtogeek.com/167425/how-to-setup-wi-fi-on-your-raspberry-pi-via-the-command-line/>
  + /etc/network/interface
  + /etc/wpa\_applicant/wpa\_applicant.conf
* gcc 4.8 installation
  + <http://ubuntuhandbook.org/index.php/2013/08/install-gcc-4-8-via-ppa-in-ubuntu-12-04-13-04/>

03/03/2015

* need to use zigbee over wifi to enable control…

03/18/2015

* re-wiring
  + H-bridge
    - <http://www.criticalvelocity.com/products/datasheets/cvhb401.pdf>
    - motorA (left):
      * MA+ → front red
      * front black → back blue
      * back red -->MA-
    - motorB (right):
      * MB+ → back red
      * back blue → front black
      * front red → MB -
    - INA1, INA2, ENA → p8, p9, p10
    - IB1, IB2, ENB → p5, p6, p7
    - 5v out → breadboard → encoder power
  + encoder
    - green: signal; red: power +5V; black: ground
  + mbed
    - 

03/21/15

* test Zigbee communication on VM
  + add serial\_generic package on VM
    - <http://askubuntu.com/questions/269257/how-to-activate-gcc-4-7-version>
  + do not connect motor controller’s Vin to positive
  + ! current problem solved
    - **connect the GND of BBB to the H-bridge**
    - **need to parallel capacitors to motor (10 uF)**
  + ***serial\_check***
    - ***need to handle reconnect issue (to same port)***
    - ***handler: if not responding, chmod again, continue***

03/24/15

* keyboard control + LIDAR test
  + sync time with host
    - ntpdate -u 0.pool.ntp.org
  + test distributed communication
    - setup environment variables
    - remember only run roscore on host, will suffice, can’t have multiple roscores running at the same time
* Problem
  + Mapping
  + could not apply control when Hokuyo module is connected
    - keystroke sent to zigbee, printed out on control-to-mbed program, but couldn’t send
    - without wifi dangler
      * rosparam set hokuyo\_node/port /dev/ttyACM0
      * rosrun hokuyo\_node getID /dev/ttyACM0
    - **with dongler**
      * mbed controller will **not** respond to serial write
        + weird issue: seems that Hokuyo module is.. powering the mbed somehow/ or writing to serial like that… weird though
      * hokuyo node could not establish connection with module
      * possible solution:
        + rosbag record **/scan** topic
        + replay offline
        + hack mbed (separate power supply from USB in), use Vin

*try BBB’s 5V regulated output*

* + - * + **buy BBB wifi cape**
  + line following
    - mbed server down

**03/26/15**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **mbed 1** | **mbed 2** | **LIDAR** | **WIFI** | **Battery** | **USB pw** | **bootable** |
| **X** | **X** | √ | **√** | **√** | **X** | **√** |
| **√** | **√** | **√** | **X** | **√** | **X** | **√** |
| **√** | **√** | **X** | **√** | **√** | **√ (?)** | **√** |
| **X** | **X** | **√** | **√** | **√** | **X** | **√** |

* **power issue**
  + mapping
    - need external power supply for mbeds
    - can’t do online mapping
      * power not enough for wifi and LIDAR together
    - offline mapping
  + line following
    - doable without hokuyo
    - might also need external power supply for mbed
    - wifi works in the beginning, however battery drains too fast and usb power shutdown
* Sol
  + change battery
    - current LIPO battery voltage: 3.7V, 7.4W
  + external power supply for mbed
  + BBB wifi cape?
    - instead of using USB as power supply for wifi

03/31/15

* **get reliable wifi on BBB**
  + <https://groups.google.com/forum/#!topic/beagleboard/9KCIs7yqsa8>
  + <http://embeddedprogrammer.blogspot.com/2013/01/beaglebone-using-usb-wifi-dongle-to.html>
  + now wifi dongle gets configured on boot up

04/01/15

* RPi
  + <https://wiki.ubuntu.com/ARM/RaspberryPi>
  + <http://www.howtogeek.com/167425/how-to-setup-wi-fi-on-your-raspberry-pi-via-the-command-line/>

04/07/15

* usb hub with diode saves the day!
* update driver.cpp
  + apply asynchronous control

04/08/15

* set microSD card as default bootup
  + sudo dd if=/dev/zero of=/dev/mmcblk1 bs=1024 count=1024
  + execute that on Linux at BBB
  + ref: <http://www.twam.info/hardware/beaglebone-black/u-boot-on-beaglebone-black>
* could control car speed (inc/dec)
  + sol1: asynchronous, ticker, analog out
    - mutex
    - ticker + wait
    - “fake” duty cycle
  + √ sol2: synchronous, Pwm
    - real controllable duty cycle
    - could reach 100%
    - Pwm pin 21 on mbed acts weirdly, switch to 22
* get encoder data
  + problem: most of the time get 0
  + possible reason:
    - InterruptIn: <0.8V → 0, >2.0V → 1; not lower than threshold
      * possible solve: add a resistor
    - Software:
      * no printf/wait in ISR (no block call)
      * no mutex in ISR
      * avoid infinite while loop in ISR
* steps to cope with
  + Encoder
    - **? how** lower the voltage input to InterruptIn pin
    - try mbed on another breadboard, only connect the encoders
    - **√** instead of thread, try acquire encoder data as a ticker in driver.cpp
    - √ also works fine in thread
      * **don’t printout l\_count, r\_count.**
  + Straight
    - remove all printfs in ticker\_handler
    - use for loop for adjustment
    - use ticker, but no while loop (just one adaption each time), check periodically
    - Try to use RTOS Timer instead of Ticker
    - **sol1.1:** ticker can not be initialized within thread
    - **sol1.2:** var called outside the script needs to be defined as extern type in header
  + straight version 2
    - 1 ticker for display update
    - 1 ticker for forward\_support
      * within forward\_support, make sure init calibration point is set every time the controller receives a new command
      * getEncoderTicks will record ticks ever since, adjust and then check the accumulate value
  + possible reason for sometimes not getting data:
    - guess: use wait in ticker

04/15/15

* Encoder + Mapping
  + coding part done
  + problem 1:
    - encoder ticks sent from mbed are too large
      * perhaps noisy/fluctuating because of the motor
      * need std::cout to printout left/right measure received
    - need to combine the packages in launch file so that we don’t need to run each one manually
  + problem 2 (solved, using slam\_demo.launch)
    - [ERROR] [1429077125.735688529]: Transform failed during publishing of map\_odom transform: Lookup would require extrapolation into the future. Requested time 1429077124.756175250 but the latest data is at time 1429077124.158656250, when looking up transform from frame [base\_footprint] to frame [odom]
    - seems that tf between odom and base\_footprint is wrong
      * need to check hector\_mapping
  + problem 3
    - encoder\_to\_tf should output x,y,z for distance, not just simple ticks

04/17/15

* 1. use re-chargable battery for motor just for now (for testing)
* 2. fix encoder data overshoot problem/integrate IMU into mapping, enable wifi support using AirPennNet-Guest
* 3. map the corridor with no overlapping
* 4. enable localization using offline map and AMCL/hector\_mapping/move\_base package in ROS
* 5. enable waypoint automation leveraging navigation stack
* (Above is the goal before **Monday night**)
* fix encoder
  + stop using ISR
    - seems like even if left is not connected, a turn in the right wheel will increment the left one (call left ISR)
    - adjacent pin will also cause a problem
  + use AnalogIn to read
    - **the pins could be picky**
    - use 16 and 20, works fine 0.994 vs 0.00024
* publish tf between odom and base\_footprint
  + <http://answers.ros.org/question/44639/how-to-broadcast-a-transform-between-map-and-odom/>
  + <http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom>
  + \* <http://answers.ros.org/question/39132/improve-gmapping-results/>
  + rewrite encoder\_odom\_tf, won’t help
    - still same error of looking into future
  + **rewrite encoder\_from\_mbed to do publishing on receiving msg.**
    - now tf could run together with encoder\_from\_mbed
    - try to use **chrony**
      * **\*\*** [**https://code.google.com/p/rosbee/wiki/chrony**](https://code.google.com/p/rosbee/wiki/chrony)
      * sudo invoke-rc.d chrony restart
    - update frequency of serial generic to be 30 Hz
    - update frequency of encoder output from mbed to be 10Hz
    - remove spinOnce, leave only one spin() at the end.
  + debug tool: rosrun rqt\_tf\_tree, tf tf\_echo /base\_footprint /odom, rviz
* try some AirPennNet MAN!
  + <http://www.seas.upenn.edu/~cis191/hw/pi_internet.html>
  + switch back to BBB
  + <http://embeddedprogrammer.blogspot.com/2013/01/beaglebone-using-usb-wifi-dongle-to.html>

04/20/15

* calibration
  + <http://www.dfrobot.com/index.php?route=product/product&product_id=98>
  + <http://www.robotshop.com/en/dfrobot-4wd-arduino-platform-encoders.html?gclid=CjwKEAjwx9KpBRCAiZ_tgYKWvhQSJABQjGW--Ypt_Migjfc426Nasrbxb6b9wLQvnEr3Rd70H85p_xoCtPTw_wcB>
  + X forward
  + y left
  + Z up
* Hector mapping can’t take in laser data
  + <http://my.phirobot.com/blog/2014-06-hector_mapping_example.html>

**Problem solved during weekend:**

1. getting realtime encoder data will acceptable tick loss.

2. solve time synchronization problem across ROS platforms (will be useful for any sensor input)

3. encoder data is fed into mapping process, but the discrepancy grows as the car moves forward, will perform even worse during turns.

The reason is that during skid steering control process, with one wheel moving forward and another one skidding, the odometry(encoder data) will be ruined. <http://www.robotplatform.com/knowledge/Classification_of_Robots/wheel_control_theory.html> Thus, encoder data may not add much help to the mapping process, which also points out that we really need to leverage the IMU.

4. Gyro and Accelerometer data is fetched from mbed and published to ROS, however it keeps complaining about not feeding in the right data.

next steps:

1. PID control for driving straight smoothly. (encoders can be really useful in this task)

2. Integrate IMU into mapping (this will help with adjustment again roll/pitch), still need to research on how to use IMU for better localization.

3. Since the car could get decent mapping within mlab, could start working on navigation stacks within lab.(easier to do localization)

* PID (P/PI)
* **encoder one last try**
  + disable theta
  + **human made obstacle**

**04/29/15**

* **PID**
  + 1. input/output: I: counts/sec; O: pwm duty cycle
  + 2. setSetPoint: set goal velocity(unit: counts/sec)
  + 3. setProcessValue set realtime feed of velocity(unit: counts/sec)
  + 4. compute calculate pwm for H-bridge control
  + test Kp
    - **duty cycle speed (counts/sec)**
    - **1 19.41**
    - **0.8 14.56**
    - **0.6 12.10**
    - **0.4 6.72**
    - average of dPV = -3.79 dCO = -0.2
    - Kp = dPV/dCO = -3.79/-20% = 0.1895 counts per sec/%
    - Tp = 0.2 s
    - theta(p) = 0.1 s
    - Kc = 2.4585 \* 29/100 = 0.7 (or 0.5)
  + PI
    - normal: Kc = 1.173 \* 29/100 = 0.34; Tc = 0.8
    - agressive: Kc = 5.863 \* 29/100 = 1.7; Tc = 0.08
    - Ti = Tp = 0.2 s
  + tune parameters experimentally
    - avoid straight adjusting in the beginning
    - set offset of right motor’s goal

**05/01**

* **Line following**
  + **purely receiver:** ambient 88
  + scenario handling in ir.cpp
    - straightline implementation (left/right)
    - cheat protocol (move back if move out of lane)
    - keep turning if hasn’t turned enough
    - use constant turn speed (0.9, 0.3 for duty cycle in order to turn faster)

current progress.

* PID control
  + The current implementation is using a PI controller to control actions including left, right and forward.
  + I’ve tuned the constants and now it could move forward and turn much smoother.
  + The car could still change speed in realtime.
  + The implementation of straight line driving(using encoder data) is done and works pretty well.
* Line following
  + I’ve “taped" a track around the pingpong table in mlab, and the bot could finish the full track with no problem. Here’s a link to the video (47 seconds) recorded if you guys are interested:<https://drive.google.com/file/d/0B2jEHYaQEznjSEJTbzNIRnVKZFU/view?usp=sharing>
  + Different scenarios are added to the line following implementation to handle turning scenarios.
  + If you look into the video, you’ll find that when the car could not make the turn, it will move back a little and readjust. It’s basically a temporary cheat protocol, and can be improved by either decreasing the turning angle or adding more sensors. I’ll go with the former solution, and work on close-loop localization leveraging LIDAR and the navigation stack.

Plan next:

* 1. Localization leveraging navigation stack
* 2. Way point navigation (straight line)
* 3. Detect whether the car has closed the loop, and stop when it’s back at the starting point.

Localization:

ref: also real-time mapping + nav (hector\_slam package doesn’t solely support this) <http://answers.ros.org/question/73261/autonomous-navigation-using-navigation-stack-and-hector-slam/>

AMCL: for using existing/static map.

fake laser odom (optional): <http://wiki.ros.org/laser_scan_matcher> can directly use hector\_mapping

trial:

1. AMCL+ Hector\_SLAM <http://answers.ros.org/question/59153/how-to-extract-information-from-hector-slam-for-autonomous-navigation/>
2. √hector\_mapping + move\_base  
   <http://answers.ros.org/question/63360/extrapolation-error-using-hector_mapping-move_base/>

steps:

1. navigation stack setup  
   swap amcl & odom with hector\_mapping(for localization and laser odom)  
    <http://wiki.ros.org/navigation/Tutorials/RobotSetup>  
   focus on the usage of move\_base
2. turtle bot navigation setup (ref)  
   <http://wiki.ros.org/turtlebot_navigation/Tutorials/indigo/Autonomously%20navigate%20in%20a%20known%20map>

ref:

* hector\_navigation using hector\_mapping
  + <http://answers.ros.org/question/112576/how-can-i-run-hector-navigation-for-exploration/>
  + <http://answers.ros.org/question/38575/navigate-in-hector-map/>
  + Chinese: <http://blog.exbot.net/archives/1129>

**05/08/15**

* static IP 192.168.1.109
  + BBB is able to connect machine within local network, however is not able to connect outside machine.
    - fix: /sbin/route add -net 0.0.0.0 gw 192.168.1.1 wlan0
    - reason: Something has deleted the route to the gateway on your server.
  + able to ping ip addresses but not host names
    - add one line dns-nameserver 8.8.8.8 under inet ~ static ~ in /etc/network/interfaces
    - then restart network/interface: sudo /etc/init.d/networking restart
    - ref: <http://askubuntu.com/questions/465729/ping-unknown-host-google-com-in-ubuntu-server>
  + F\*\*\*IT!
    - This guy is doing exactly what I want, without using odometry, instead use hector\_mapping only
    - <https://github.com/DaikiMaekawa/ros-navigation2d-example>
    - adapt param files to fit in my bot scenario
    - add nav\_control.cpp in car\_slam package
      * velocity control:
        + left\_speed\_out = cmd\_vel.linear.x - cmd\_vel.angular.z\*ROBOT\_WIDTH/2  
          right\_speed\_out = cmd\_vel.linear.x + cmd\_vel.angular.z\*ROBOT\_WIDTH/2

plan for tomorrow:

* better control logic in nav\_control
* may need mbed code specifically for navigation (at least cancel drive straight)
* update local\_costmap (tune param for obstacle avoidance using laser)
* use static map

stop when loop closed

* change in ir\_to\_mbed.cpp, add in subscriber on topic slam\_out\_pose, compare initial location to current location.
* if within circular range then consider reached, sending only stop

Other:

might need continuous control in the future (now is by command discrete fashion)

05/09/15

* differential motor control w/o straight driving adjustment
  + try velocity control: √
    - left\_speed\_out = cmd\_vel.linear.x - cmd\_vel.angular.z\*ROBOT\_WIDTH/2  
      right\_speed\_out = cmd\_vel.linear.x + cmd\_vel.angular.z\*ROBOT\_WIDTH/2
* skid steering wheel control (⬆⬇) no need
* larger obstacle edge range (avoidance)
* static map
  + generate a map using bag file (2015-04-19-02-13.bag)
    - <http://wiki.ros.org/slam_gmapping/Tutorials/MappingFromLoggedData>
* close loop detection for IR line following