**TABLE OF CONTENTS**

[SETTING UP UBUNTU 14.04 ON BEAGLEBONE](#h.77aor22oeskr)

[SETTING UP ROS - INDIGO ON BEAGLEBONE](#h.82etm9y6f7ab)

[FIXING SHARED\_PTR BUG IN UBUNTU 12.04](#h.dpk9bso8jazx)

[SOLDERING BATTERY (LI-POL 3.8 Volts)TO BEAGLEBONE](#h.say0zsd7znim)

[imu\_9dof NODE](#h.yi69of79h16a)

[IMU CONNECTION](#h.b0tbpvvlrnvf)

[ZIGBEE MODULE PINOUT SETUP](#h.uapyu4tcgug1)

[REPOSITORY](#h.pjoa2fhhspnf)

[send\_command](#h.gl9hbpeo4y5n)

[SUPPORT FOR BOOST REGEX](#h.vb24liq1qyxf)

[UPDATING THE CODE](#h.wbyj7jcjjvsi)

[ROS::nodeHandle](#h.hmzqn2ty5qtp)

# **SETTING UP UBUNTU 14.04 ON BEAGLEBONE**

I installed the latest version of Ubuntu by following the guidelines given here:

<http://elinux.org/BeagleBoardUbuntu>

* 1. **TO FLASH ON EMMC BLOCK (has less memory)**

I followed the version where you can flash the image on the emmC block, thus not requiring an sd card to run.

Download the image:

***wget*** [***https://rcn-ee.net/rootfs/2015-02-19/flasher/BBB-eMMC-flasher-ubuntu-14.04.2-console-armhf-2015-02-19-2gb.img.xz***](https://rcn-ee.net/rootfs/2015-02-19/flasher/BBB-eMMC-flasher-ubuntu-14.04.2-console-armhf-2015-02-19-2gb.img.xz)

Verify the image :

***md5sum BBB-eMMC-flasher-ubuntu-14.04.2-console-armhf-2015-02-19-2gb.img.xz***

it should output

857d9d8f05f781de63c5d389938d0cf1 BBB-eMMC-flasher-ubuntu-14.04.2-console-armhf-2015-02-19-2gb.img.xz

unzip and put the image on usd card:

***unxz BBB-eMMC-flasher-ubuntu-14.04.2-console-armhf-2015-02-19-2gb.img.xz***

***sudo dd if=./BBB-eMMC-flasher-ubuntu-14.04.2-console-armhf-2015-02-19-2gb.img of=/dev/mmcblk0***

I faced some difficulty in flashing into the sdcard. Make sure you take an empty sdcard , and once flashed verify the image is there using *gparted application****.*** It should be about 1.6~ 1.7gb. Please use at least 4 to 8 gb sd card. Also the 2gb partition did not work . make sure the ***partition is 4gb***.

* 1. **CONNECTING BEAGLEBONE TO PC**

I used putty to connect to the beaglebone, though its not at all required.

Putty Configurations:

ssh: 192.168.7.2

port 22

You can also verify if the bone is connected using the beaglebone website

Note: If you are using the windows you cannot use putty directly. First install the bone drivers.

You can also visualize this by connecting hdmi cable to the bone and connecting a keyboard to the beaglebone should get you working as well.

* 1. **GETTING INTERNET TO THE BEAGLEBONE**

The following details are for setting up internet on beaglebone from a linux PC. I followed the procedure as per the below link : <http://shallowsky.com/blog/hardware/talking-to-beaglebone.html>

Make sure eth1 is configured to 192.168.7.1 if not configure it using

**ifconfig eth1 192.168.7.1**

Make your linux machine as router for beaglebone

on beaglebone

**/sbin/route add default gw 192.168.7.1**

**sudo cp /etc/resolv.conf /etc/resolv.conf.backup  
echo "nameserver 8.8.8.8" >> /etc/resolv.conf**

**Note:**

The above echo command did not work for me instead I did this

**sudo nano /etc/resolv.conf**

add nameserver 8.8.8.8 in the end and save the file.

on linux machine:

**sudo iptables -A POSTROUTING -t nat -j MASQUERADE  
echo 1 | sudo tee /proc/sys/net/ipv4/ip\_forward > /dev/null**

This should setup the internet, you can verfiy by pinging google.

to set proper time to beaglebone you can enter the following:

**/usr/bin/ntpdate -b -s -u pool.ntp.org**

# **SETTING UP ROS - INDIGO ON BEAGLEBONE**

I followed instructions as given here:

<http://wiki.ros.org/indigo/Installation/UbuntuARM>

* 1. **CONFIGURE REPOSITORY IN BEAGLEBONE**

First configure the ubuntu repository, without this I wasn’t able to install ros itself. This is the error I get without configuring the repositories:

***unable to locate ros-indigo-ros-base***

To configure from the command prompt (since desktop is not installed), I followed the instructions given here:

<https://help.ubuntu.com/community/Repositories/CommandLine>

make backup

**sudo cp /etc/apt/sources.list /etc/apt/sources.list.backup**

uncomment the following below lines:

*deb http://us.archive.ubuntu.com/ubuntu/ trusty universe  
deb-src http://us.archive.ubuntu.com/ubuntu/ trusty universe  
deb http://us.archive.ubuntu.com/ubuntu/ trusty-updates universe  
deb-src http://us.archive.ubuntu.com/ubuntu/ trusty-updates universe*

* 1. **SET LOCALE**

**sudo update-locale LANG=C LANGUAGE=C LC\_ALL=C LC\_MESSAGES=POSIX**

* 1. **SETUP THE SOURCE LIST**

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

* 1. **SETUP THE KEYS**

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

The above command did not work for me quite well, I guess I did a dd -. So make sure you type the command properly else you get an error like this

***public key is not available No\_PUBKEY 5523BAEEB01FA116***

If you get a such an error type the command :

**wget http://packages.ros.org/ros.key -O - | sudo apt-key add -**

which is what worked for me.

* 1. **INSTALLATION**

**sudo apt-get update**

**sudo apt-get install ros-indigo-ros-base**

You can install any other ros package you want here.

* 1. **INITIALIZE ROSDEP**

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

* 1. **ENVIRONMENT SETUP**

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

**source /opt/ros/indigo/setup.bash**

**sudo apt-get install python-rosinstall**

* 1. **VERIFY OS NAME**

**cat /etc/lsb-release**

The content should be as below:

DISTRIB\_ID=Ubuntu  
DISTRIB\_RELEASE=14.04  
DISTRIB\_CODENAME=trusty  
DISTRIB\_DESCRIPTION="Ubuntu 14.04"

# **FIXING SHARED\_PTR BUG IN UBUNTU 12.04**

There is a bug in UBUNTU 12.04 and ROS SETUP where the use of sensor\_msgs:imu in my class was creating an issue. There were a number of errors but the most important is :

***error: use of deleted function 'boost::shared\_ptr::shared\_ptr(const boost::shared\_ptr&)'  
In file included from /usr/include/boost/shared\_ptr.hpp:17:0,***

This is apparently a bug which I found out using the link : <http://stackoverflow.com/questions/18900730/boostshared-ptrshared-ptrconst-boostshared-ptr-is-implicitly-declared>

The fix involves updating the move constructor as follows :

***sudo cp /usr/include/boost/smart\_ptr/shared\_ptr.hpp /usr/include/boost/smart\_ptr/shared\_ptr.hpp.backup***

***sudo nano /usr/include/boost/smart\_ptr/shared\_ptr.hpp***

*// Move support*

*#if defined( BOOST\_HAS\_RVALUE\_REFS )*

*shared\_ptr( shared\_ptr && r ): px( r.px ), pn() // never throws*

*{*

*pn.swap( r.pn );*

*r.px = 0;*

*}*

***shared\_ptr(const shared\_ptr&) = default;***

***NOTE: THIS NEEDS TO BE FIXED IN weak\_ptr.hpp and shared\_array.hpp as well. BUT I HAVE NOT TESTED THIS.***

A better fix would have be to reinstall the boost library from 1.49 to 1.54 version. However almost the entire ros is built upon the boost library, and installing 1.54 version, requires that I completely remove the 1.49 version using ***autoremove*** command which will remove all possible libraries associated with 1.49. This is just BAD.

# **SOLDERING BATTERY (LI-POL 3.8 Volts)TO BEAGLEBONE**

I followed instructions as given here: <http://www.element14.com/community/community/designcenter/single-board-computers/next-gen_beaglebone/blog/2013/08/10/bbb--rechargeable-on-board-battery-system>

1. Solder pin tp5 and tp6.
2. Connect +ve of battery to tp8 and negative to either tp6 ot tp5.
3. This will mostly still not boot the battery up. However what I found is, sometimes the board needs to be started by adapter initially then removed. Then plug the Li-Pol battery. Sometimes it works and sometimes it doesn’t.
4. A more sure way to start is to to first connect the Li-Pol battery then connect the Adapter to power the board, once powered completely remove the adapter.
5. ***However this method does not power up the USB. I believe the voltage of 3.8 V and 2A is insufficient to drive the USB.***

# **imu\_9dof NODE**

This code was developed by me.

I made use of the boost:asio libraries to talk to a serial port:

Important files:

1. *Imu\_main.cpp*
2. *Imu\_Parser.cpp*
3. *SensorMsg\_Imu.cpp*
4. *Serial\_Connect.cpp*

***Imu\_main*** starts the node, make sure you give the address of the usb to which the Imu module is connected as an argument and also make sure it’s given write permission.

For eg: if connected to /dev/ttyACM0 then

**chmod a+rw /dev/ttyACM0**

All ros related initialization is done in this code.

***Serial\_Connect*** has a class to connect to the serial port.

***Imu\_Parser*** makes use of regex and extract the Imu related information

***SensorMsg\_Imu*** packs the information and sends as a topic.

the topic to listen to is *imu\_9dof*

The delimiters between the Imu messages I’ve kept is ORIENTB and EORIENT

ORIENTB is for ***B***eginning of ***Orient***ation msg

EORIENT is for ***E***nd of ***Orient***ation msg

to run the ros node, make sure roscore is running. then run using

in catkin workspace

***catkin\_make***

***source devel/setup.bash***

***rosrun imu\_9dof imu\_9dof***

# **IMU CONNECTION**

Connect

GND→ GND

VCC → Vout

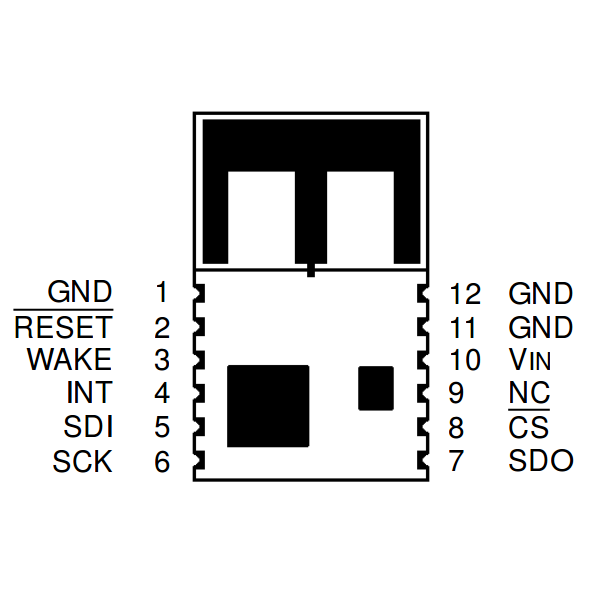
SCL → Pin27

SDA → Pin 28

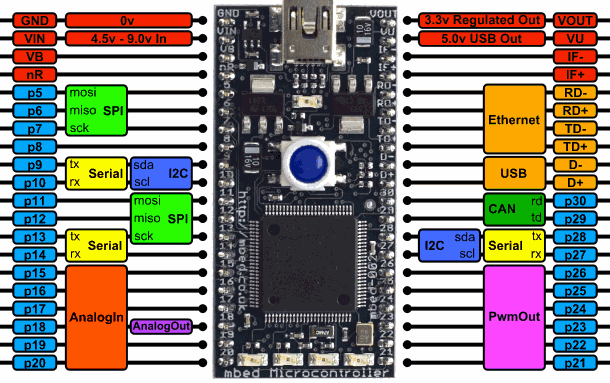
# **ZIGBEE MODULE PINOUT SETUP**

* 1. **PIN-OUTS FOR MODULES**

The pin out of the zigbee MRF24J40MA is as below



The pinout of the mbed module is as below



* 1. **PIN CONNECTIONS**

The connections are as follows:

|  |  |  |
| --- | --- | --- |
| **Zigbee Pin (Slave)** | **Mbed Pin (Master)** | **Functionality** |
| 1 | 1 | GND → GND |
| 6 | 13 | Serial Clock → SCK |
| 5 | 11 | Serial Data Input → MOSI |
| 2 | 21 | Reset Zigbee → Software defined reset |
| 7 | 12 | Serial Data Output → MISO |
| 8 | 14 | Chip Select →Serial Receiver |
| 10 | 40 | Voltage to drive |

# **REPOSITORY**

All of our work is saved in the git repository of mlab.

***git clone*** [***https://github.com/mlab/autobots.git***](https://github.com/mlab/autobots.git)

|  |  |  |  |
| --- | --- | --- | --- |
| **Modules** | **Info** | **Subscribing to** | **Publishing as** |
| get\_control\_info | Reads the ***serial\_generic*** topic for commands and publishes on ***car\_slam/cmd\_vel*** as input for the motors to run | **serial\_generic** | **car\_slam/cmd\_vel** |
| imu\_9dof | reads from the designated serial port directly and send imu as ***imu\_9dof*** topic | **None** (reads directly from port. For eg: /dev/ttyACM0 via **zigbee\_and\_imu** mbed) | **imu\_9dof** |
| imu\_generic | reads from ***serial\_generic*** topic and decodes to find the imu messages | **serial\_generic** | **imu\_generic** |
| serial\_generic | reads anything sent to the designated serial port and sends a ***serial\_generic*** topic | **None** (reads directly from port. For eg: /dev/ttyACM0 via **zigbee\_and\_imu** mbed) | **serial\_generic** |
| send\_command | This is a standalone application to send arrow and space commands | **None** | **None** (Sends data directly to port. For eg:  /dev/ttyACM0) |
| zigbee\_and\_imu | Mbed code to receive imu and control code simultaneously. | **None** | **None** sends data to serial port |
| zigbee\_transmit | Mbed code to transmit control code from PC | **None** receives data from serial port sent from ***send\_command*** | **None** |
|  |  |  |  |

# **send\_command**

Currently there is no Cmake file for it. It needs to be built manually. This is **NOT** a ROS node.

It simply sends the arrow of the space as input to the mbed via the serial port. The serial port needs to be provided with a+rw permissions. This code will **neglect** all the other keyboard inputs. Notice the arrow keys are not standard ascii values. These are system dependent. And this code **will only work for linux**. The function getch function is modified to get inputs from arrow keys. The code for getch is taken from here:

<http://stackoverflow.com/questions/7469139/what-is-equivalent-to-getch-getche-in-linux>

# **SUPPORT FOR BOOST REGEX**

When trying to extract string between delimiters earlier the boost regex was used. However the boost 1.49 version in the beaglebone is not good enough for such regex handling.

Thus I will be using **substring** commands to extract the delimiters out. That said the boost regex can be enabled once the BOOST\_REGEX macro is enabled.

# **UPDATING THE CODE**

**DO NOT USE THE WIFI TO UPDATE THE CODE FROM GIT HUB. I REPEAT DO NOT DO THAT!**

Simply remove the sdcard, store the nodes in the sd card directly in the home/catkin\_ws/ folder from your pc and then boot the beaglebone with the sd card. You will need to compile the nodes. However make sure the time is updated. Else even if you build it will not update the binary.

to update time it is

***sudo data MMDDHHMMYYYY.SS***

# **Things to consider when Building ROS Nodes**

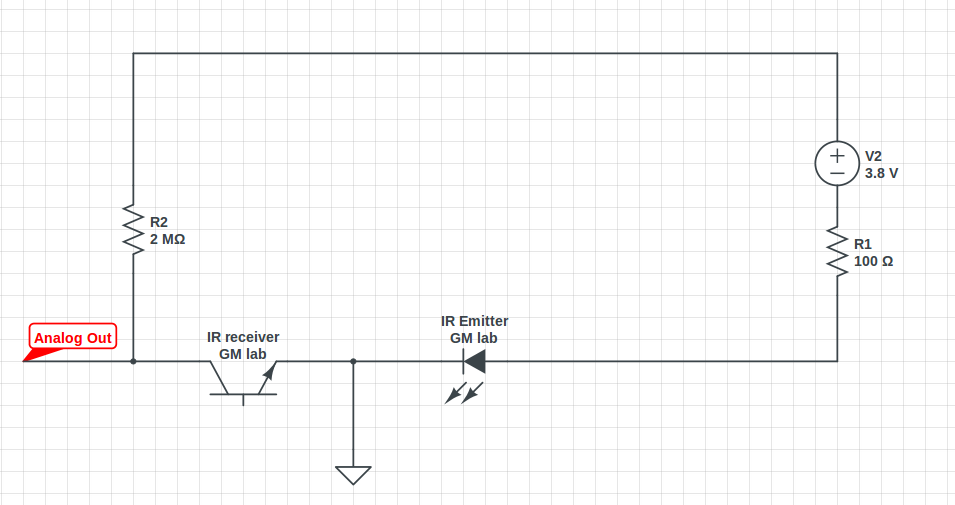
1. In any ros node make sure there is only one instance of the ros::nodeHandle. Else it doesn’t work. Yes Its true, I couldn’t send as a topic.
2. Make sure the dependencies are always there when you create a package. std\_msgs , roscpp rospy are a must dependency.
3. If you have not added them Its better to create a new ros node with these dependencies and copy the source files in these than modifying the existing nodes. Creates lot of problems.
4. When building you might occasionally face this error

***g++: internal compiler error: Killed (program cc1plus)***

***Please submit a full bug report***

If this is the case restart the beaglebone. The problem is due to less swap space.

# **IR sensors**



Emitter: yellow; Receiver: red;

The resistor values chosen are based on the Mechatronics Lab specifications.

The “Analog Out” input needs to be provided to the mbed. (Pin15 to Pin 20)

I have for now used pin 19 and pin 20 as analog inputs for left and right.

If Left and Right is less than 35% they are set to 0.

If Both Left and right set to 0:

stop

else

Left is greater than 1.2\*Right then move left

Else If Right is greater than 1.2\*Left then move right

Else Move forward

Forward is encoded as char ‘1’

Backward is encoded as char ‘2’

Right is encoded as char ‘3’

Left is encoded as char ‘4’

Stop is encoded as char ‘0’

USB generic

<http://stackoverflow.com/questions/15342804/c-linux-detect-all-serial-ports>

<http://stackoverflow.com/questions/7674287/c-checking-the-type-of-a-file-using-lstat-and-macros-doesnt-work>

<http://unix.stackexchange.com/questions/61484/find-the-information-of-usb-devices-in-c>

best example do man ftw (file tree walk) there is a sample code. Build your code from that

# **H Bridge**

We were earlier using critical velocity H-bridge for driving the motors. It is a Dual 2A H-Bridge.

We faced a lot of issues using this and never suspected the H-bridge to be the issue.

Issues faced:

1. The mbeds randomly reset. There are two mbeds, one for the zigbee communication, and IR sensors and another for the motor control. The motor control mbed is connected to the h-bridge. It has pwm pins to control the h-bridge modes (forward , backward, etc). This means there should be a common ground for these pins to work, which is what caused us a lot of problems.
2. When the motors run, the ground pin at the H-bridge (connected to the ground pin of the mbed) is not exactly 0 Volts. Sometimes its +ve .7voltage and sometimes negative. Thus putting a diode at the ground resolves the issue for sometime. However this is not always stable, since the direction of the diodes need to be reversed based on the voltage comming out at this pin.
3. Also putting an Inductor of 1mH at the power rails is stable. However voltage supplied to motors are now reduced making it to rotate slowly.
4. Tried a variety of capacitors inductors and optocouplers. None helped.

What worked is changing the H-bridge itself : TB6612FNG, this is a small hbridge and works flawlessly.

# **IR Sensors**

We initially bought the [Zumo Reflectance sensor array](https://www.pololu.com/product/1419). But could not get it to working, since it was a digital Zumo Reflectance sensor primarily designed for zumo robot which runs using arduino. I did change the library, but that did not work. The chip had been spoilt by then. We then bought the analog sensors [QTR-3A Reflectance sensor array](https://www.pololu.com/product/2456) and the [QRT-L-1A Reflectance sensor array](https://www.pololu.com/product/2454). These sensor arrays work well but the sensors should be less than 3mm from the surface to work well!