Testing Hardware

This tutorial assumes that you have completed the hardware connections, configured xbee modules, able to program the microcontroller. In this tutorial, you are expected to install libraries required, interface MPU6050, Motor Encoders with ATmega2560.

To use Arduino with ROS, some libraries have to be installed. Using the rosserial_arduino package, you can use ROS directly with the Arduino IDE. *rosserial* provides a ROS communication protocol that works over your Arduino's UART. It allows your Arduino to be a full-fledged ROS node which can directly publish and subscribe to ROS messages, publish TF transforms, and get the ROS system time.

ROS connections are implemented as an Arduino library. Libraries are a collection of code that makes it easy for you to coonect to a sensor, display, module etc. For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays.

In order to use the rosserial libraries in your own code, you must first put "***#include <ros.h>***" prior to including any other header files otherwise the Arduino IDE will not be able to locate them.

Type the following commands to generate the required libraries:

```
cd catkin_ws/src
git clone https://github.com/ros-drivers/rosserial.git
cd catkin_ws
catkin_make
```

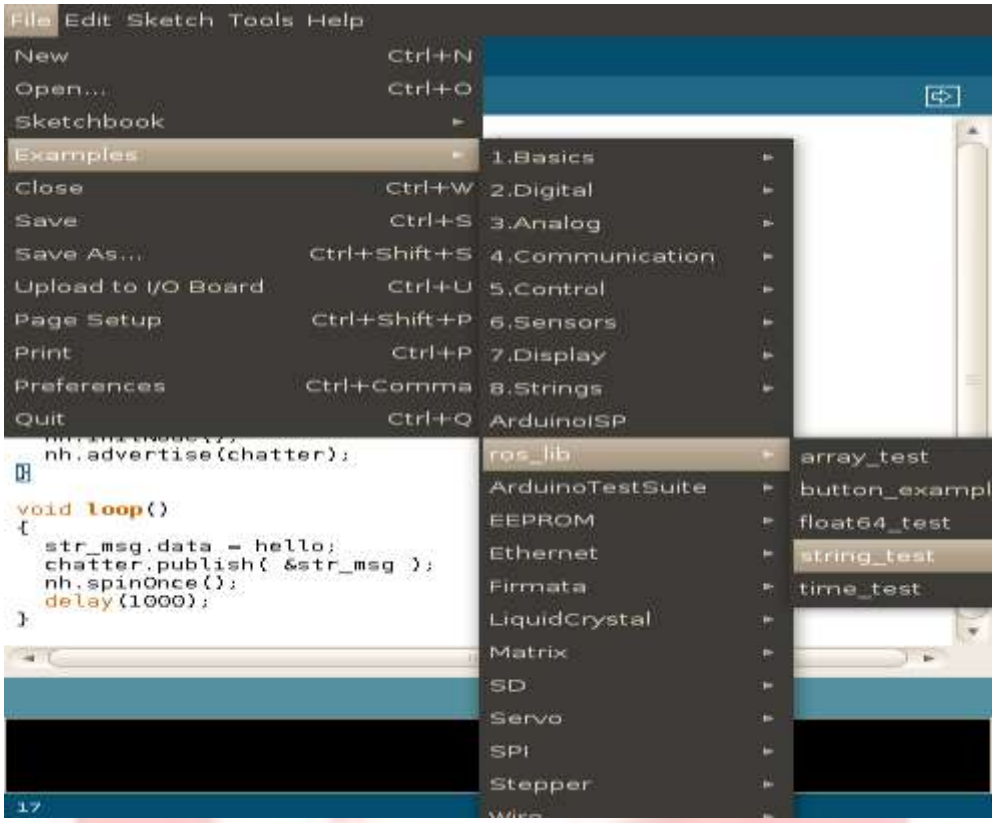
These commands clone rosserial from the github repository, generate the rosserial_msgs needed for communication, and make the ros_lib library in the catkin_ws directory.

Type the following commands to generate the required libraries:

```
cd catkin_ws
sudo devel/setup.bash
cd catkin_ws/src/rosserial/roserial_arduino/src/roserial_arduino/
chmod a+x make_libraries.py
rosrun roserial_arduino make_libraries.py
```

Now you can see that ros_lib library got created in catkin_ws folder. Copy this folder to the path ~/Arduino/libraries folder.

Now if you restart your IDE, you can see **ros_lib** added in Examples menu.



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

Check if everything is going fine:

Open **HelloWorld** example from File->Examples->ros_lib->HelloWorld. Compile it and program ATmega with the .hex file generated.

After successful programming, launch the roscore in a new terminal window:

roscore

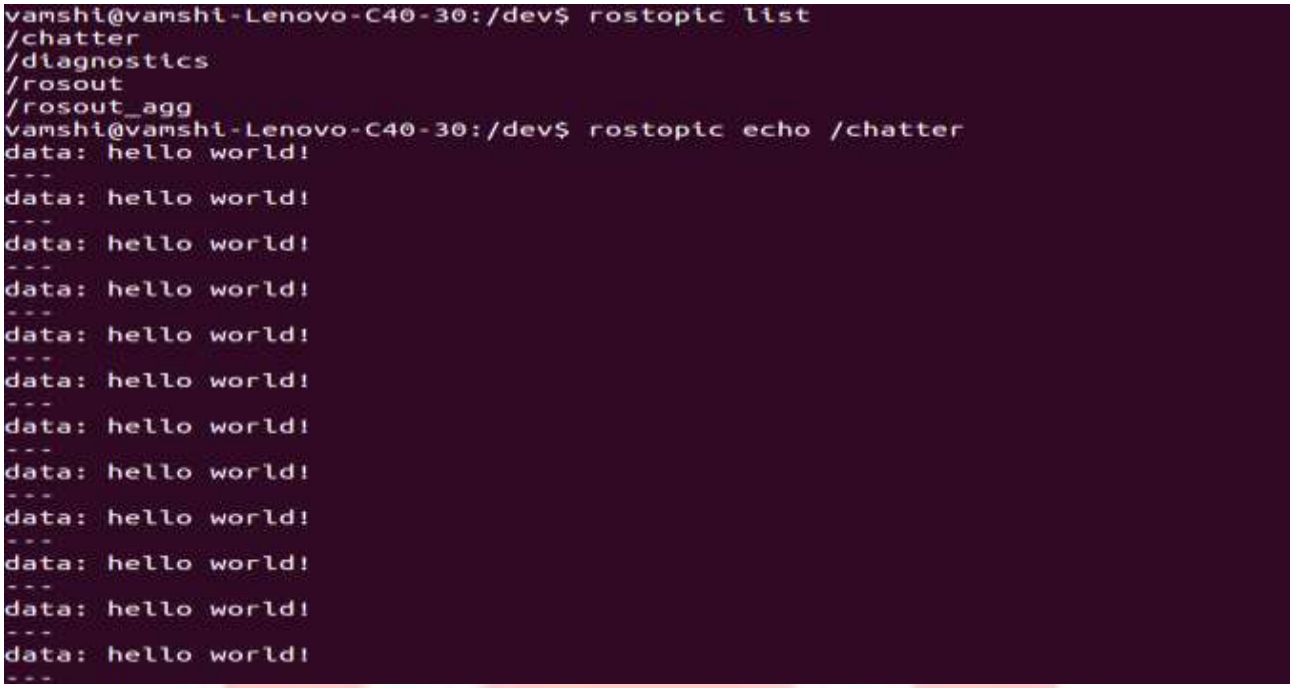
Next, run the rosserial client application that forwards your Arduino messages to the rest of ROS. Make sure to use the correct serial port:

rosrun rosserial_python serial_node.py /dev/ttyUSB0

Finally, watch the greetings come in from your Arduino by launching a new terminal window and running the following command:

rostopic echo chatter

The output should be as shown in figure below:



Code explanation is given here.

MPU6050 testing:

- Install gtkterm by the following command:

  sudo apt-get install gtkterm

- Copy **i2dev.zip** and **MPU6050.zip** (provided in ##give path##) in the **~/Arduino/libraries** folder. Extract them.

- Now restart the Arduino IDE. Now you can see that **MPU6050** also got added in the examples menu. Now open **MPU6050_raw**. Compile using verify icon.

- Burn the hex code into the microcontroller.

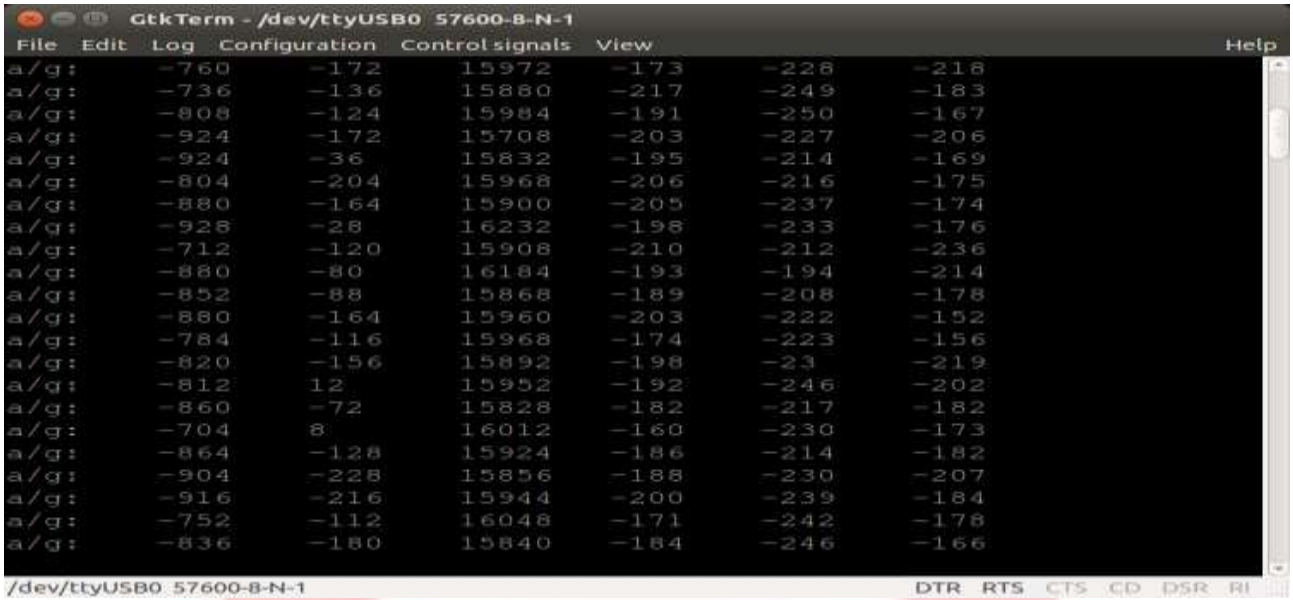
- Now open gtkterm by following command:

  sudo gtkterm

  Enter password if it prompts for it.

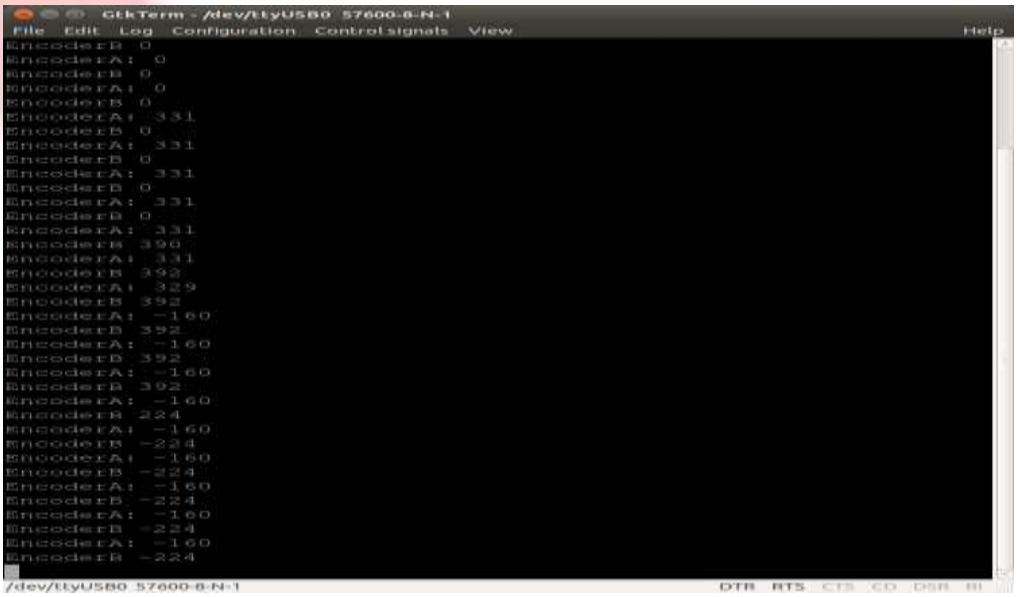
Open port under configuration tab. Set the port as **ttyUSB0** and baud rate as **57600** and leave the remaining untouched. Press apply. Reset the microcontroller.

Then you can see the accelerometer and gyroscope raw data. Check the data by changing the orientation and tilting the sensor. Your output may be as shown in figure below:



Check encoders' data:

- Burn ***motor_testing_encoder.hex*** code into the microcontroller (provided in "test file" folder of hardware testing part).
- Open gtkterm and connect through XBee as you have done previously.
- The output on the serial console should be as shown in figure below:



Check that the values are zeros initially. They increase if wheels are rotated in clockwise and decrease if wheels are rotated in anti-clockwise direction.