

## 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 `roserial_arduino` [package](#), you can use ROS directly with the Arduino IDE. `roserial` 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 connect 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 `roserial` 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/roserial.git
cd catkin_ws
catkin_make
```

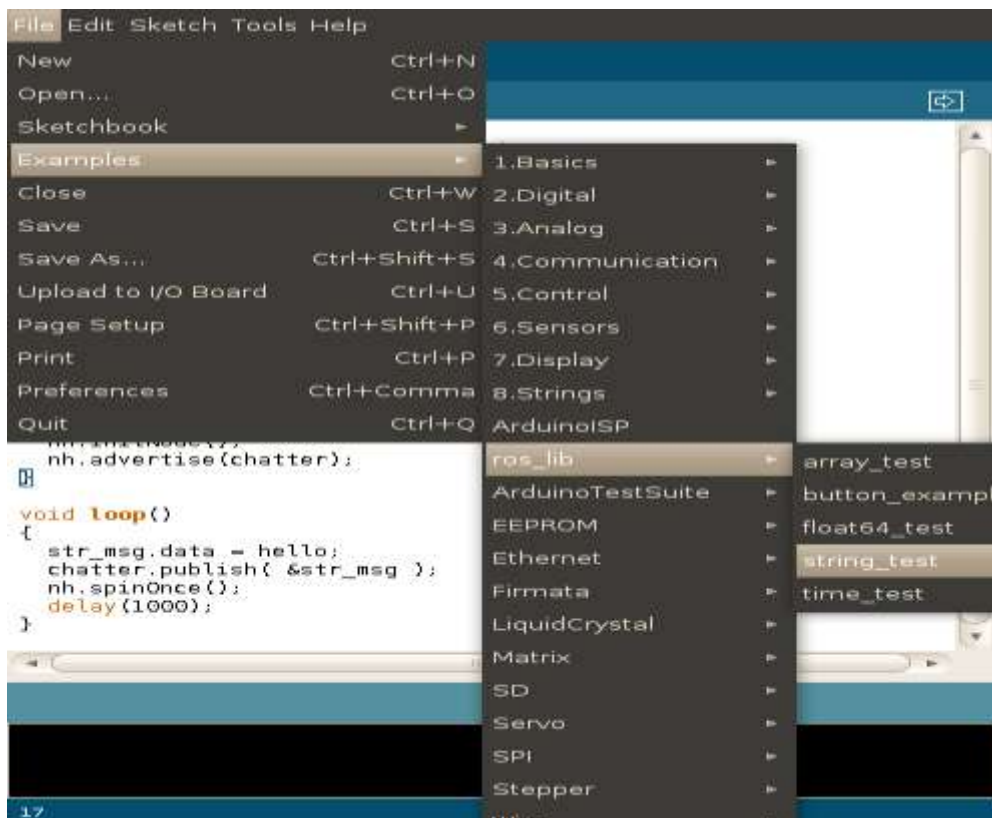
These commands clone `roserial` from the github repository, generate the `roserial_msgs` needed for communication, and make the `ros_lib` library in the `catkin_ws` directory.

Type the following commands to generate the required libraries:

```
cd catkin_ws
sudo devel/setup.bash
cd catkin_ws/src/roserial/roserial_arduino/src/roserial_arduino/
chmod a+x make_libraries.py
roscpp run 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/rosterial\\_arduino/Tutorials/Arduino%20IDE%20Setup](http://wiki.ros.org/rosterial_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 roserial client application that forwards your Arduino messages to the rest of ROS. Make sure to use the correct serial port:

***rostrun roserial\_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:

```
vamshi@vamshi-Lenovo-C40-30:/dev$ rostopic list
/chatter
/diagnostics
/rosout
/rosout_agg
vamshi@vamshi-Lenovo-C40-30:/dev$ rostopic echo /chatter
data: hello world!
---
data: hello world!
---
data: hello world!
---
data: hello world!
---
data: hello world!
---
data: hello world!
---
data: hello world!
---
data: hello world!
---
data: hello world!
---
data: hello world!
---
data: hello world!
---
data: hello world!
```

Code explanation is given [here](#).

## MPU6050 testing:

- Install gkterm by the following command:  
  
***sudo apt-get install gkterm***
- 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 gkterm by following command:

***sudo gkterm***

Enter password if it prompts for it.

Open port under configuration tab. Set the port as **ttUSB0** 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:

```

File Edit Log Configuration Control signals View Help
a/g: -760 -172 15972 -173 -226 -216
a/g: -736 -136 15880 -217 -249 -183
a/g: -808 -124 15984 -191 -250 -167
a/g: -924 -172 15708 -203 -227 -206
a/g: -924 -36 15832 -195 -214 -169
a/g: -804 -204 15968 -206 -216 -175
a/g: -880 -164 15900 -205 -237 -174
a/g: -928 -28 16232 -198 -233 -176
a/g: -712 -120 15908 -210 -212 -236
a/g: -880 -80 16184 -193 -194 -214
a/g: -852 -88 15868 -189 -208 -178
a/g: -880 -164 15960 -203 -222 -152
a/g: -784 -116 15968 -174 -223 -156
a/g: -820 -156 15892 -198 -23 -219
a/q: -812 12 15952 -192 -246 -202
a/g: -860 -72 15828 -182 -217 -182
a/g: -704 8 16012 -160 -230 -173
a/g: -864 -128 15924 -186 -214 -182
a/g: -904 -228 15856 -188 -230 -207
a/g: -916 -216 15944 -200 -239 -184
a/g: -752 -112 16048 -171 -242 -178
a/g: -836 -180 15840 -184 -246 -166

/dev/ttyUSB0_57600-8-N-1 DTR RTS CTS CD DSR RI
```

### 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:

```

GltTerm - /dev/ttyUSB0 57600-8-N-1
File Edit Log Configuration Control signals View Help
EncoderA: 0
EncoderB: 0
EncoderA: 0
EncoderB: 0
EncoderA: 331
EncoderB: 0
EncoderA: 331
EncoderB: 0
EncoderA: 331
EncoderB: 0
EncoderA: 331
EncoderB: 0
EncoderA: 331
EncoderB: 390
EncoderA: 331
EncoderB: 392
EncoderA: 329
EncoderB: 392
EncoderA: -160
EncoderB: 392
EncoderA: -160
EncoderB: 392
EncoderA: -160
EncoderB: 392
EncoderA: -160
EncoderB: 224
EncoderA: -160
EncoderB: -224
EncoderA: -160
EncoderB: -224
EncoderA: -160
EncoderB: -224
EncoderA: -160
EncoderB: -224
EncoderA: -160
EncoderB: -224
EncoderA: -160
EncoderB: -224
/dev/ttyUSB0 57600-8-N-1
DTR RTS CTS CD CRSD BI

```

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.

