Hello World: Creating a Publisher

The Code

We'll start our exploration into rosserial by creating a "hello world" program for our Arduino. (Note: the Arduino community often calls source code for programs a "sketch", we will use the same convention below). If you have followed the [Arduino IDE Setup](http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup) tutorial, you'll be able to open the sketch below by choosing ros\_lib -> HelloWorld from the Arduinoexamples menu.

This should open the following code in your IDE:

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World)

[1](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_1) /\*

[2](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_2) \* rosserial Publisher Example

[3](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_3) \* Prints "hello world!"

[4](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_4) \*/

[5](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_5)

[6](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_6) #include <ros.h>

[7](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_7) #include <std\_msgs/String.h>

[8](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_8)

[9](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_9) ros::NodeHandle nh;

[10](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_10)

[11](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_11) std\_msgs::String str\_msg;

[12](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_12) ros::Publisher chatter("chatter", &str\_msg);

[13](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_13)

[14](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_14) char hello[13] = "hello world!";

[15](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_15)

[16](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_16) void setup()

[17](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_17) {

[18](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_18) nh.initNode();

[19](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_19) nh.advertise(chatter);

[20](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_20) }

[21](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_21)

[22](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_22) void loop()

[23](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_23) {

[24](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_24) str\_msg.data = hello;

[25](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_25) chatter.publish( &str\_msg );

[26](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_26) nh.spinOnce();

[27](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_27) delay(1000);

[28](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-f52c3186d145d0ab359f23e0b05b5ea7cee62c79_28) }

The Code Explained

Now, let's break the code down.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World)

[6](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-5826d3601828e55baddc97302302a2bbdf6a5f62_6) #include <ros.h>

[7](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-5826d3601828e55baddc97302302a2bbdf6a5f62_7) #include <std\_msgs/String.h>

[8](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-5826d3601828e55baddc97302302a2bbdf6a5f62_8)

As a part of every ROS Arduino program, you need to include the ros.h header file and header files for any messages that you will be using.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World)

[9](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-0931df68a0576b0231e71d81c449487199e1dd1f_9) ros::NodeHandle nh;

Next, we need to instantiate the node handle, which allows our program to create publishers and subscribers. The node handle also takes care of serial port communications.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World)

[11](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-6ce4c2b4ee7163df2d1f0e9cb529d4b8a40ef044_11) std\_msgs::String str\_msg;

[12](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-6ce4c2b4ee7163df2d1f0e9cb529d4b8a40ef044_12) ros::Publisher chatter("chatter", &str\_msg);

We need to instantiate the publishers and subscribers that we will be using. Here we instantiate a Publisher with a topic name of "chatter". The second parameter to Publisher is a reference to the message instance to be used for publishing.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World)

[16](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-96f01e519593a568c0a728d452aa9670d4b5a2ce_16) void setup()

[17](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-96f01e519593a568c0a728d452aa9670d4b5a2ce_17) {

[18](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-96f01e519593a568c0a728d452aa9670d4b5a2ce_18) nh.initNode();

[19](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-96f01e519593a568c0a728d452aa9670d4b5a2ce_19) nh.advertise(chatter);

[20](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-96f01e519593a568c0a728d452aa9670d4b5a2ce_20) }

In the Arduino setup function you then need to initialize your ROS node handle, advertise any topics being published, and subscribe to any topics you wish to listen to.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World)

[22](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-afeb2edcb33158d9707af3fc02fbdf6bd1bf878f_22) void loop()

[23](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-afeb2edcb33158d9707af3fc02fbdf6bd1bf878f_23) {

[24](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-afeb2edcb33158d9707af3fc02fbdf6bd1bf878f_24) str\_msg.data = hello;

[25](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-afeb2edcb33158d9707af3fc02fbdf6bd1bf878f_25) chatter.publish( &str\_msg );

[26](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-afeb2edcb33158d9707af3fc02fbdf6bd1bf878f_26) nh.spinOnce();

[27](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-afeb2edcb33158d9707af3fc02fbdf6bd1bf878f_27) delay(1000);

[28](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World#CA-afeb2edcb33158d9707af3fc02fbdf6bd1bf878f_28) }

Finally, in the loop function, the node publishes "Hello World" and calls ros::spinOnce() where all of the ROS communication callbacks are handled.

Uploading the Code

To upload the code to your Arduino, use the upload function within the Arduino IDE. This is no different from uploading any other sketch.

Running the Code

Now, launch the [roscore](http://wiki.ros.org/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

rosrun rosserial\_python serial\_node.py \_port:=/dev/ttyUSB0

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

rostopic echo chatter

Blink: Creating a Subscriber

The Code

Now that we've created a ROS publisher in the [previous tutorial](http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World), we'll create a subscriber. If you have followed the [Arduino IDE Setup tutorial](http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup), you'll be able to open the sketch below by choosing ros\_lib -> Blink from the Arduino examples menu.

This should open the following code in your IDE:

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink)

[1](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_1) /\*

[2](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_2) \* rosserial Subscriber Example

[3](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_3) \* Blinks an LED on callback

[4](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_4) \*/

[5](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_5)

[6](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_6) #include <ros.h>

[7](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_7) #include <std\_msgs/Empty.h>

[8](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_8)

[9](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_9) ros::NodeHandle nh;

[10](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_10)

[11](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_11) void messageCb( const std\_msgs::Empty& toggle\_msg){

[12](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_12) digitalWrite(13, HIGH-digitalRead(13)); // blink the led

[13](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_13) }

[14](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_14)

[15](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_15) ros::Subscriber<std\_msgs::Empty> sub("toggle\_led", &messageCb );

[16](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_16)

[17](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_17) void setup()

[18](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_18) {

[19](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_19) pinMode(13, OUTPUT);

[20](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_20) nh.initNode();

[21](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_21) nh.subscribe(sub);

[22](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_22) }

[23](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_23)

[24](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_24) void loop()

[25](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_25) {

[26](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_26) nh.spinOnce();

[27](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_27) delay(1);

[28](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-7f48e62c93848c28c1e367f430d0051ad1546342_28) }

The Code Explained

Now, let's break the code down.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink)

[6](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-4624f3d6209a34554955fb46c11c585bb0e12a56_6) #include <ros.h>

[7](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-4624f3d6209a34554955fb46c11c585bb0e12a56_7) #include <std\_msgs/Empty.h>

[8](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-4624f3d6209a34554955fb46c11c585bb0e12a56_8)

As before, we need to include the ros.h as with any other ROS Arduino program. We also include the header files for messages, in this case, the Empty message.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink)

[9](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-0931df68a0576b0231e71d81c449487199e1dd1f_9) ros::NodeHandle nh;

Next, we need to instantiate the node handle, which allows our program to create publishers and subscribers. The node handle also takes care of serial port communications.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink)

[11](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-d2be4794fdc0cb75c46bdbb0a2bef854f261056d_11) void messageCb( const std\_msgs::Empty& toggle\_msg){

[12](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-d2be4794fdc0cb75c46bdbb0a2bef854f261056d_12) digitalWrite(13, HIGH-digitalRead(13)); // blink the led

[13](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-d2be4794fdc0cb75c46bdbb0a2bef854f261056d_13) }

We then create the callback function for our subscriber. The call back function must take a constant reference of a message as its argument. In our callback messageCb, the type of message is std\_msgs::Empty and the message name will betoggle\_msg.

Inside our callback, we could reference toggle\_msg, but since it is empty, there is no need to. We just blink the LED on the Arduino every time we receive a message.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink)

[15](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-15b3bf02ab3052e2642f972a6c600782a9196ca4_15) ros::Subscriber<std\_msgs::Empty> sub("toggle\_led", &messageCb );

We need to instantiate the publishers and subscribers that we will be using. Here we instantiate a Subscriber with a topic name of "toggle\_led" and type std\_msgs::Empty. With Subscribers, you must remember to template the subscriber upon the message. Its two arguments are the topic it will be subscribing to and the callback function it will be using.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink)

[17](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-6ae9cd358bd0ac5073c1712539920fc931f10fb1_17) void setup()

[18](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-6ae9cd358bd0ac5073c1712539920fc931f10fb1_18) {

[19](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-6ae9cd358bd0ac5073c1712539920fc931f10fb1_19) pinMode(13, OUTPUT);

[20](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-6ae9cd358bd0ac5073c1712539920fc931f10fb1_20) nh.initNode();

[21](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-6ae9cd358bd0ac5073c1712539920fc931f10fb1_21) nh.subscribe(sub);

[22](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-6ae9cd358bd0ac5073c1712539920fc931f10fb1_22) }

In the Arduino setup function you then need to initialize your ROS node handle, advertise any topics being published, and subscribe to any topics you wish to listen to.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink)

[24](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-3c13d57378bfddd71f741800140388b782c15c8c_24) void loop()

[25](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-3c13d57378bfddd71f741800140388b782c15c8c_25) {

[26](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-3c13d57378bfddd71f741800140388b782c15c8c_26) nh.spinOnce();

[27](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-3c13d57378bfddd71f741800140388b782c15c8c_27) delay(1);

[28](http://wiki.ros.org/rosserial_arduino/Tutorials/Blink#CA-3c13d57378bfddd71f741800140388b782c15c8c_28) }

Finally, in the loop function we call ros::spinOnce() where all of the ROS communication callbacks are handled. We don't need to do any additional processing in the loop(), since ros::spinOnce() will handle passing messages to the subscriber callback.

Uploading the Code

To upload the code to your Arduino, use the upload function within the Arduino IDE. This is no different from uploading any other sketch.

Running the Code

Now, launch the [roscore](http://wiki.ros.org/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

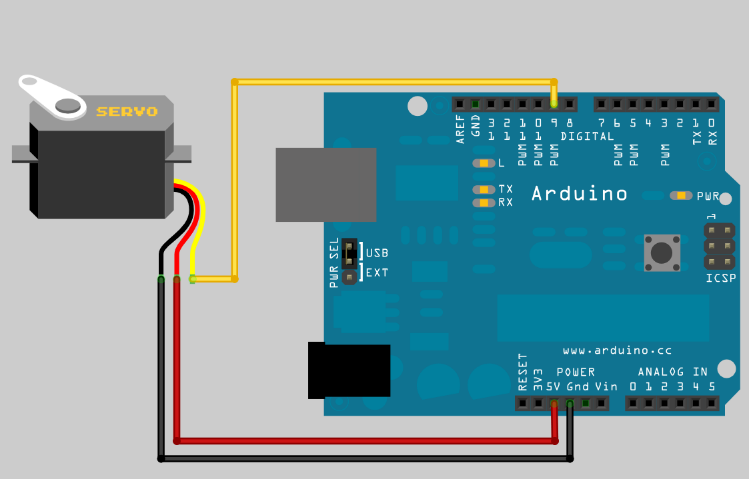
Finally, you can toggle the LED using [rostopic](http://wiki.ros.org/rostopic):

rostopic pub toggle\_led std\_msgs/Empty --once

Servo Controller Example

**Description:** Tutorial for controlling an R/C servo with rosserial and an Arduino

Hardware



This example assumes that you have an Arduino and a hobby r/c servo. The r/c servo can be purchased from your local hobby shop, [Towerhobbies](http://www3.towerhobbies.com/cgi-bin/wti0001p?&I=LXUK84&P=ML), [Sparkfun](http://www.sparkfun.com/products/9064), etc.

The hobby servo r/c are great little actuators because they are relatively cheap (as low as $10) but contain a gear box and motor control electronics. They are controlled by sending a squarewave pulse of 1-2 milliseconds in width every 20 milliseconds. This typically moves the servo arm from 0-180 degrees. Hobby servo's come in a huge variety of sizes, torques, and angular precision.

Code

The code for this tutorial is made extremely simple through the use of the Arduino Servo library. The [Servo Library](http://www.arduino.cc/en/Reference/Servo)handles all of the low level control to generate and maintain the servo pulses. All your code needs to do is specify the pin the servo is attached to and then write the angle to the servo object. Underneath, the Servo library uses the Arduino's built in timer interrupts to generate the correct pulses. In this example, we only control one servo, but the same library can be used to control up to 12 servos on most Arduino boards and 48 on the Arduino Mega.

[Dis/Attivare numerazione delle righe](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller)

[1](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_1) /\*

[2](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_2) \* rosserial Servo Control Example

[3](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_3) \*

[4](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_4) \* This sketch demonstrates the control of hobby R/C servos

[5](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_5) \* using ROS and the arduiono

[6](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_6) \*

[7](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_7) \* For the full tutorial write up, visit

[8](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_8) \* www.ros.org/wiki/rosserial\_arduino\_demos

[9](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_9) \*

[10](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_10) \* For more information on the Arduino Servo Library

[11](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_11) \* Checkout :

[12](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_12) \* http://www.arduino.cc/en/Reference/Servo

[13](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_13) \*/

[14](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_14)

[15](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_15) #if defined(ARDUINO) && ARDUINO >= 100

[16](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_16) #include "Arduino.h"

[17](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_17) #else

[18](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_18) #include <WProgram.h>

[19](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_19) #endif

[20](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_20)

[21](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_21) #include <Servo.h>

[22](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_22) #include <ros.h>

[23](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_23) #include <std\_msgs/UInt16.h>

[24](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_24)

[25](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_25) ros::NodeHandle nh;

[26](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_26)

[27](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_27) Servo servo;

[28](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_28)

[29](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_29) void servo\_cb( const std\_msgs::UInt16& cmd\_msg){

[30](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_30) servo.write(cmd\_msg.data); //set servo angle, should be from 0-180

[31](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_31) digitalWrite(13, HIGH-digitalRead(13)); //toggle led

[32](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_32) }

[33](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_33)

[34](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_34)

[35](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_35) ros::Subscriber<std\_msgs::UInt16> sub("servo", servo\_cb);

[36](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_36)

[37](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_37) void setup(){

[38](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_38) pinMode(13, OUTPUT);

[39](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_39)

[40](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_40) nh.initNode();

[41](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_41) nh.subscribe(sub);

[42](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_42)

[43](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_43) servo.attach(9); //attach it to pin 9

[44](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_44) }

[45](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_45)

[46](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_46) void loop(){

[47](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_47) nh.spinOnce();

[48](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_48) delay(1);

[49](http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller#CA-e8097364ea8467e706b5b6a3e568070b9087926f_49) }

The key servo specific areas here are the fact that we made a global Servo object, attached to the correct arduino pin, and then on every servo topic call back, we write the servos new angle to the servo object.

Testing

First, startup your roscore and the rosserial python node in their own terminal windows.

roscore

rosrun rosserial\_python serial\_node.py \_port:=/dev/ttyUSB0

Now, in a new terminal window, use rostopic pub to control your servo! Simply specify the angle from 0-180 and watch it move.

rostopic pub servo std\_msgs/UInt16 <angle>