Note: This tutorial assumes that you have completed the previous tutorials: Hello World (example Publisher) (/rosserial_arduino/Tutorials/Hello%20World).

Please ask about problems and questions regarding this tutorial on answers.ros.org (http://answers.ros.org). Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Blink (example subscriber)

Description: This tutorial shows step by step how to use rosserial with subscribers.

Tutorial Level: BEGINNER

Next Tutorial: Using Time and TF (/rosserial_arduino/Tutorials/Time%20and%20TF)

Contents

1. Blink: Creating a Subscriber

1.1 The Code

Now that we've created a ROS publisher in the previous tutorial (/rosserial_arduino/Tutorials/Hello%20World), we'll create a subscriber. If you have followed the Arduino IDE Setup tutorial (/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:

Toggle line numbers

```
 1 /*
 2  * rosserial Subscriber Example
 3  * Blinks an LED on callback
 4  */
 5
 6 #include <ros.h>
 7 #include <std_msgs/Empty.h>
 8
 9 ros::NodeHandle nh;
10
11 void messageCb( const std_msgs::Empty& toggle_msg){
12   digitalWrite(13, HIGH-digitalRead(13));   // blink the led
13 }
14
15 ros::Subscriber<std_msgs::Empty> sub("toggle_led", &messageCb );
16
17 void setup()
18 {
19   pinMode(13, OUTPUT);
20   nh.initNode();
21   nh.subscribe(sub);
22 }
23
24 void loop()
25 {
26   nh.spinOnce();
27   delay(1);
28 }
```

1.2 The Code Explained

Now, let's break the code down.

Toggle line numbers

```
6 #include <ros.h>
7 #include <std_msgs/Empty.h>
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.

Toggle line numbers

```
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.

Toggle line numbers

```
11 void messageCb( const std_msgs::Empty& toggle_msg){
12   digitalWrite(13, HIGH-digitalRead(13));   // blink the led
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 be `toggle_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.

Toggle line numbers

```
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.

Toggle line numbers

```
17 void setup()
18 {
19   pinMode(13, OUTPUT);
20   nh.initNode();
21   nh.subscribe(sub);
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.

Toggle line numbers

```
24 void loop()
25 {
26   nh.spinOnce();
27   delay(1);
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.

1.3 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.

1.4 Running the Code

Now, launch the roscore (/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 (/rostopic):

```
rostopic pub toggle_led std_msgs/Empty --once
```

2. Further Reading

Please see rosserial/Overview (/rosserial/Overview/Publishers%20and%20Subscribers) for more information on publishers and subscribers. Also see limitations (/rosserial/Overview/Limitations) for information about more complex data types.

Wiki: rosserial_arduino/Tutorials/Blink (last edited 2015-07-31 23:43:38 by ChrisDickson (/ChrisDickson))