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.

Using Time and TF

Description: This tutorial shows how to use ros::Time and TF to create a tf publisher on the Arduino.

Tutorial Level: INTERMEDIATE

Next Tutorial: Measuring Temperature (/rosserial_arduino/Tutorials/Measuring%20Temperature)

electric	fuerte	groovy	hydro	indigo	jade	kinetic

Contents

1. Using Time and TF on the Arduino
    1. The Code
    2. The Code Explained
    3. Uploading the Code
    4. Running the Code
2. Further Reading

1. Using Time and TF on the Arduino

The `rosserial_arduino` package contains libraries for generating timestamps on the Arduino which are synchronized with the PC/Tablet on which the roscore instance is running. This tutorial shows how to access time, using an example of publishing a `tf` transform.

1.1 The Code

If you have followed the Arduino IDE Setup tutorial, you'll be able to open the sketch below by choosing `ros_lib -> TimeTf` from the Arduino examples menu.

This should open the following code in your IDE:

Toggle line numbers

```
/*
 * rosserial Time and TF Example
 * Publishes a transform at current time
 */

#include <ros.h>
#include <ros/time.h>
#include <tf/transform_broadcaster.h>

ros::NodeHandle  nh;

geometry_msgs::TransformStamped t;
tf::TransformBroadcaster broadcaster;

char base_link[] = "/base_link";
char odom[] = "/odom";

void setup()
{
  nh.initNode();
  broadcaster.init(nh);
}

void loop()
{
  t.header.frame_id = odom;
  t.child_frame_id = base_link;
  t.transform.translation.x = 1.0;
  t.transform.rotation.x = 0.0;
  t.transform.rotation.y = 0.0;
  t.transform.rotation.z = 0.0;
  t.transform.rotation.w = 1.0;
  t.header.stamp = nh.now();
  broadcaster.sendTransform(t);
  nh.spinOnce();
  delay(10);
}
```

1.2 The Code Explained

Now, let's break the code down.

Toggle line numbers

```
#include <ros.h>
#include <ros/time.h>
#include <tf/transform_broadcaster.h>

```

We need to include our typical ROS stuff, as well as the transform broadcaster.

Toggle line numbers

```
geometry_msgs::TransformStamped t;
tf::TransformBroadcaster broadcaster;

char base_link[] = "/base_link";
char odom[] = "/odom";
```

Next, we instantiate a `TransformStamped` message to use, and a broadcaster. We also need to specify the names of the frames we are publishing a transform for.

Toggle line numbers

```
  broadcaster.init(nh);
```

Inside the `setup()` function, we have to call `init()` on the TransformBroadcaster with the node handle as a parameter. Without doing this, the broadcaster will not publish correctly.

Toggle line numbers

```
  t.header.frame_id = odom;
  t.child_frame_id = base_link;
  t.transform.translation.x = 1.0;
  t.transform.rotation.x = 0.0;
  t.transform.rotation.y = 0.0;
  t.transform.rotation.z = 0.0;
  t.transform.rotation.w = 1.0;
```

Inside the `loop()` function, we fill in the fields of our transform. The frame IDs are set to the correct string names, and the values of the translation and rotation are set.

Toggle line numbers

```
  t.header.stamp = nh.now();
```

Calling `nh.now()` returns the current time, as a `ros::Time` instance, just like when using roscpp's ros::Time::now() (/roscpp).

Toggle line numbers

```
  broadcaster.sendTransform(t);
  nh.spinOnce();
  delay(10);
}
```

Finally, we publish the transform, and then `spinOnce` and wait a bit before doing it again. Transforms should always be published at a regular rate, although usually the data fields will be filled in from real data.

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:

Finally, you can check the transform using:

```
rosrun tf tf_echo odom base_link
```

Or, by running:

```
rosrun tf view_frames
```

2. Further Reading

Please see rosserial/Overview (/rosserial/Overview/Time) for more information using Time instances.