Sistema de monitorización inercial del movimiento de las extremidades superiores

Daniel Fernández Villanueva

21 de junio de 2013

Índice general

Parte I

Memoria

Capítulo 1

ANTECEDENTES

Capítulo 2

OBJETIVO DEL PROYECTO

Este proyecto tiene como objetivo la implementación de un sistema de monitorización en tiempo real del movimiento de las extremidades superiores del cuerpo humano. Para la realización de este sistema se tendrá que dar solución a los siguientes

1. **Aplicación para la adquisición de datos de los sensores inerciales**:
   Realización de un sistema que permitirá obtener en tiempo real los datos que proporcionan los sensores. En concreto se desarrollará un driver para un sensor xsens o una red de sensores xsens conectados mediante un master xbus. Este driver permitirá leer los datos que proporcionan los acelerómetros, giróscopos, magnetómetros y sensores de temperatura, además de la orientación de cada uno de los sensores conectados al PC. Este driver se encargará también de crear una interfaz para la posterior utilización de los datos en otros programas de foma sencilla.

2. **Tratamiento de los datos para obtener los ángulos de rotación entre cada sensor**:
   Una vez sea posible la adquisición de los datos con el driver anterior, se creará otro programa con el que se obtendrán los ángulos de rotación entre cada sensor teniendo en cuenta además la geometría de las articulaciones del brazo o modelo sobre las que se situarán los sensores.

3. **Utilización de los datos para el objetivo deseado**:
   En esta última fase se crearán los sistemas necesarios para la utilización de los datos con el objetivo deseado:

   - Para la visualización de la posición del brazo en el simulador 3D Gazebo, se creará un modelo del brazo y una interfaz etre ROS y el simulador que permitirá la visualización de la posición del brazo en tiempo real.
   - Para el control del robot mediante el movimiento del brazo o modelo del robot físico, se creará otra interfaz entre ROS y el driver del propio robot.

Capítulo 3

ESPECIFICACIONES DE DISEÑO

Capítulo 4

DISEÑO DEL SISTEMA

4.1. Estudio de soluciones

4.2. Simulación

Capítulo 5

IMPLEMENTACIÓN FÍSICA

5.1. Selección de componentes

5.2. Montaje

5.3. Ajuste

Capítulo 6

PROTOCOLO DE PRUEBAS. REDISEÑO

Capítulo 7

RESULTADOS OBTENIDOS

Capítulo 8

HERRAMIENTAS UTILIZADAS

En este capítulo se realizará una breve descripción de los elementos utilizados en el proyecto, tanto de hardware como de software.

8.1. Hardware

8.1.1. Sensores XSENS

Sensor MTi-G

XBUS Master

8.1.2. Brazo humano

8.1.3. Robot Youbot

8.1.4. Modelo del robot Youbot

8.2. Software

8.2.1. ROS

¿Qué es ROS?

ROS (del inglés *Robot Operating System* - Sistema Operativo Robótico) es una plataforma de desarrollo de software que incluye conjunto de utilidades centradas en ayudar al desarrollador en la creación de programas para el control de robots. Esta herramienta incorpora abstracción del hardware, drivers para dispositivos, librerías, visualizadores, utilidades para el intercambio de mensajes entre programas y administradores de paquetes de software, entre otras muchas cosas. ROS es además software abierto, bajo una licencia BSD, por lo que cualquier persona puede ver su código fuente y modificarlo.

¿Por qué usar ROS?

ROS proporciona solución a diversos problemas que vienen dados inherentemente al objetivo de este proyecto:

- **Creación y compilación de programas**
  - **Gestor de paquetes**
- **Comunicación entre programas:** ROS incluye:
  - **Máster**
  - **Topics**
  - **Servicios**
  - **Servidor de parámetros**

- **Visualización de datos**
- **Otras herramientas**
  - **Bag**
  - **rxplot**

8.2.2. El sistema operativo Ubuntu

¿Qué es Ubuntu?

Ubuntu es un sistema operativo con núcleo Linux. Es gratuito y es distribuido como software *open source*. Se trata de la distribución GNU/Linux más popular en equipos personales

¿Por qué usar Ubuntu?

8.2.3. El lenguaje de programación C++

8.2.4. Simulador Gazebo

Gazebo es programa para simulación en 3D de un robot o una población de robots interactuando entre sí y el ambiente. Incorpora simulación de la física de sólidos rígidos y de la respuesta de sensores. Además incorpora un visualizador 3D bastante potente que permite ver en tiempo real la posición de los robots de forma realista.

La versión de Gazebo que se va a utilizar proporciona también una serie de herramientas para la comunicación con ROS.

8.2.5. Visualizador RViz

8.2.6. Control de versiones: git

Capítulo 9

PROCESO DE REALIZACIÓN

En este capítulo se detallará el proceso de realización de cada una de las fases del proyecto.

9.1. Creación del driver para la adquisición de datos de los sensores xsens

En está primera fase se tratará de encontrar un método para la toma de datos de la red de sensores inerciales. Estos sensores estarán conectados a un máster, que irá conectado al PC mediante conexión USB. Los datos así obtenidos se publicarán en *topics* de ROS.

Método seguido

Para la realización del driver se ha partido del código incluido en la documentación de los sensores

9.1.1. El paquete xsens_driver

9.2. Creación de una librería matemática en C++ que permita trabajar con posiciones y orientaciones

9.2.1. La librería dfv

La clase Quaternion

La clase Vector3

La clase Matrix

9.3. Cálculo de los ángulos formados entre cada sensor

9.3.1. Formas de representar orientaciones espaciales

Se puede configurar el sensor xsens para que proporcione una de las siguientes representaciones de su orientación:

- Ángulos de Euler.
- Matriz de rotación.
- Cuaternión.

Ángulos de Euler

Matrices de rotación

Cuaterniones

9.3.2. Utilización de cuaterniones para la representación de rotaciones

Sea un vector unitario:

$$\vec{e} = e_x i + e_y j + e_z k, \quad \|e\| = \sqrt{e_x^2 + e_y^2 + e_z^2} = 1$$

Se definirá el cuaternión de rotación con ángulo θ sobre el eje dado por el vector $\vec{e}$:

$$q(\theta, \vec{e}) = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) e_x i + \sin\left(\frac{\theta}{2}\right) e_y j + \sin\left(\frac{\theta}{2}\right) e_z k$$

Se puede descomponer el cuaternión como suma de un número real y un cuaternión imaginario puro multiplicado por otro número real:

$$q(\theta, \vec{e}) = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) e$$

Donde e es el cuaternión imaginario puro (parte real nula) cuyas componentes se corresponden a las del vector unitario $\vec{e}$ que define el eje de rotación.

Por cuestión de comodidad:

$$c_i = \cos\left(\frac{\theta_i}{2}\right)$$

$$s_i = \sin\left(\frac{\theta_i}{2}\right)$$

De tal forma que ahora el cuaternión se escribirá de la siguiente manera:

$$q(\theta_i, \vec{e_i}) = c_i + s_i e_i$$

El producto de dos cuaterniones se puede expresar de esta forma:

$$q_1 q_2 = (c_1 + s_1 e_1)(c_2 + s_2 e_2) = c_1 c_2 + c_1 s_2 e_2 + c_2 s_1 e_1 + s_1 s_2 e_1 e_2$$

El conjugado del cuaternión se representará de la siguiente manera:

$$conj(q) = q^*$$

Dado a que se va a trabajar siempre con cuaterniones unitarios (también llamados *versores*), se podrá asumir lo siguiente:

$$\|q\| = 1, \quad q^{-1} = \frac{q^*}{\|q\|^2} = q^*$$

De tal forma que:

$$qq^* = q^* q = 1$$

Rotación de un vector alrededor de un eje y un ángulo dados

Se puede realizar la rotación de un vector alrededor de un eje $\vec{e}$ y un ángulo θ mediante la siguiente operación:

$$p' = q_1 p q_1^*, \quad q_1 = q(\theta_1, \vec{e_1})$$

Si multiplicamos a la izquierda por q_1^* y por la derecha por q_1:

$$q_1^* p' q_1 = q_1^* (q_1 p q_1) q_1^*$$

$$q_1^* p' q_1 = p$$

Obtenemos que el conjugado representa una rotación inversa a la del cuaternión original:

$$q^* = q\left(\theta, -\vec{e}\right) = q\left(-\theta, \vec{e}\right)$$

Composición de rotaciones en coordenadas extrínsecas

El cuaternión de rotación definido por $q\left(\theta, \vec{e}\right)$ representa una rotación alrededor de un eje $\vec{e}$ fijo al sistema de referencia global.

Realizando una nueva rotación al vector obtenido anteriormente, se obtendrá una rotación compuesta por una primera rotación definida por el cuaternión q_1 seguida de otra rotación definida por q_2:

$$p'' = q_2 p' q_2^*, \quad q_2 = q(\theta_2, \vec{e_2})$$

$$p'' = q_2(q_1 p q_1^*)q_2^* = (q_2 q_1)p(q_1^* q_2^*) = q_{12} p q_{12}^*$$

Se puede expresar la composición de dos rotaciones como un nuevo cuaternión que resulta de la multiplicación en orden inverso de los cuaterniones que definen las dos rotaciones:

$$q_{12} = q_2 q_1$$

De aquí se obtiene que el conjugado del producto de dos cuaterniones es el producto de los conjugados en orden inverso:

$$q_{12} = (q_2 q_1)^* = q_1^* q_2^*$$

De forma análoga, para n cuaterniones:

$$q_{12\ldots(n-1)n} = q_n q_{n-1} \cdots q_2 q_1$$
$$q_{12\ldots(n-1)n} = (q_n q_{n-1} \cdots q_2 q_1)^* = q_1^* q_2^* \cdots q_{n-1}^* q_n^*$$

Composición de rotaciones en coordenadas intrínsecas

Relación entre rotaciones intrínsecas y extrínsecas

A continuación se demostrará que una rotación compuesta por varias rotaciones en el sistema de coordenadas intrínseco del sólido rígido se corresponde a la composición de rotaciones en el sistema extrínseco realizadas en orden inverso.

Se define un cuaternión de rotación asociado al eje intrínseco $\vec{e_2}$ después de haber sufrido una rotación definida por el cuaternión $q_1 = q\left(\theta_1, \vec{e_1}\right)$ como:

$$q_2^I = q\left(\theta_2, \vec{e_2}^I\right), \quad \vec{e_2}^I = (q_1 e_2 q_1^*)_{\vec{v}}$$

La orientación de partida es la misma para el sistema intrínseco y el extrínseco, por lo tanto:

$$q_1^E = q_1^I = q_1$$

Para demostrar la afirmación de partida se tendrá que demostrar la veracidad de la siguiente igualdad:

$$q_2^I q_1 = q_1 q_2^E$$

Se reordenará la igualdad para que los cálculos sean más sencillos:

$$q_2^I q_1 = q_1 q_2^E \iff q_2^I = q_1 q_2^E q_1^*$$

Desarrollo del lado izquierdo de la igualdad:

$$q_2^I = c_2 + s_2(q_1 e_2 q_1^*)$$

$$q_1 e_2 q_1^* = (c_1 + s_1 e_1) e_2 (c_1 - s_1 e_1) = (c_1 e_2 + s_1 e_1 e_2)(c_1 - s_1 e_1) =$$

$$= c_1^2 e_2 + s_1 c_1 e_1 e_2 - c_1 s_1 e_2 e_1 - s_1^2 e_1 e_2 e_1 = c_1^2 e_2 + s_1 c_1 (e_1 e_2 - e_2 e_1) - s_1^2 e_1 e_2 e_1$$

9.4. Incorporación de las herramientas creadas

9.4.1. Visualizador de la posición del brazo

Obtención de los ángulos de rotación entre cada segmento del brazo

Cálculo de las posiciones de cada segmento del brazo

Implementación

9.4.2. Controlador de un simulador del brazo robótico del robot Youbot

9.4.3. Controlador del brazo robótico del robot Youbot real

Capítulo 10

RESULTADOS EXPERIMENTALES

Capítulo 11

CONCLUSIONES

Capítulo 12

BIBLIOGRAFÍA

Parte II

Anexos

Apéndice A

INSTALACIÓN Y PUESTA EN MARCHA DEL SOFTWARE

Apéndice B

SOLUCIÓN DE PROBLEMAS

B.1. Error iniciando Gazebo

```
Msg Waiting for master
Msg Connected to gazebo master @ http://localhost:11345
Exception [Master.cc:69] Unable to start server[Address already in use]


terminate called after throwing an instance of 'gazebo::common::Exception'
Aborted (core dumped)
[gazebo-1] process has died [pid 2795, exit code 134, cmd /opt/ros/fuerte/stacks/simulator_gaze
/opt/ros/fuerte/stacks/simulator_gazebo/gazebo_worlds/worlds/empty.world __name:=gazebo
__log:=/home/daniel/.ros/log/772c2f96-ab75-11e2-a2fc-001de05009b5/gazebo-1.log].
log file: /home/daniel/.ros/log/772c2f96-ab75-11e2-a2fc-001de05009b5/gazebo-1*.log
LightListWidget::OnLightMsg
```

Solución: Ejecutar comando:

```
$ ps ax | grep [g]z
```

Ver si hay un proceso gzserver

```
 3118 ?        Sl    12:47
 /opt/ros/fuerte/stacks/simulator_gazebo/gazebo/gazebo/bin/gzserver
 /opt/ros/fuerte/stacks/simulator_gazebo/gazebo_worlds/worlds/empty.world __name:=gazebo
__log:=/home/daniel/.ros/log/8188cc76-ab73-11e2-a4ec-001de05009b5/gazebo-1.log -s /opt/ros/fuer
-s /opt/ros/fuerte/stacks/simulator_gazebo/gazebo/lib/libgazebo_ros_api_plugin.so
```

Si lo hay, ejecutar *System Monitor* y matar el proceso *gzserver*.

Apéndice C

Instalación y configuración del software necesario

C.1. Instalación de ROS Fuerte

La versión de ROS que se utilizará es ROS Fuerte. Esta elección se debe a que dicha versión es compatible con el simulador Gazebo, con el que posteriormente se realizará la visualización en 3D del modelo.

Para realizar la instalación de ROS se partirá de una instalación previa de Ubuntu, pudiendo ser éste de cualquiera de estas *releases*:

- 10.04 LTS (Lucid Lynx)
- 11.04 (Oneiric Ocelot)
- 12.04 LTS (Precise Pangolin)

C.1.1. Configuración de los repositorios de Ubuntu

Se procederá a abrir el Centro de Sofware de Ubuntu y en la barra de menú de dicho programa, se seleccionará en el menú Edit la opción Software Sources. En la pestaña Ubuntu Software se comprobará que están seleccionados los repositorios restricted, universe y multiverse:

C.1.2. Configuración del archivo sources.list

El archivo sorces.list le dice al gestor de paquetes de Ubuntu de dónde puede obtener cada paquete de ROS. Se abrirá un terminal y se ejecutará el siguiente comando que dependerá de la versión de Ubuntu que tengamos instalada:

Ubuntu 10.04 (Lucid)

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu lucid main"
> /etc/apt/sources.list.d/ros-latest.list'
```

Ubuntu 11.10 (Oneiric)

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu oneiric main"
> /etc/apt/sources.list.d/ros-latest.list'
```

Ubuntu 12.04 (Precise)

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main"
> /etc/apt/sources.list.d/ros-latest.list'
```

Este comando lo que hace es crear un archivo de texto en la ruta especificada como parámetro, que contiene la dirección de donde descargar los paquetes para la versión específica de ROS que tengamos.

C.1.3. Configuración de la keys

En el terminal se ejecutará el siguiente comando:

```
$ wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
```

C.1.4. Descarga e instalación

Se actualizará el índice de paquetes de Ubuntu para tener la seguridad de que el servidor de ROS.org está indexado:

```
$ sudo apt-get update
```

A continuación se procederá a descargar e instalar la versión completa de ROS. El el terminal se ejecutará el siguiente comando:

```
$ sudo apt-get install ros-fuerte-desktop-full
```

Esta instalación traerá consigo las siguientes herramientas, entre otras:

- ROS
- rx (herramientas para interfaz gráfica: rxbag, rxgraph, rxplot, ...)
- rviz (herramienta de visualización 3D)
- librerías genéricas para robots
- Simuladores 2D/3D (entre ellos Gazebo)
- Navegación y percepción 2D y 3D

C.1.5. Configuración del entorno

Cada vez que se inicie un nuevo terminal es necesario añadir las variables de entorno. Si se quisiera, se puede automatizar dicha tarea ejecutando el comando:

```
$ echo \source /opt/ros/fuerte/setup.bash" >> ~/.bashrc
```

Este comando añade la línea source /opt/ros/fuerte/setup.bash al archivo /.bashrc. Este archivo contiene la configuración inicial del terminal, y se ejecuta cada vez que abrimos un nuevo terminal. Posteriormente se ejecutará el archivo anterior para actualizar el terminal. De esta forma reconocerá los nuevos comandos de ROS:

```
$ . ~/.bashrc
```

C.1.6. Otras herramientas

Se instalarán dos herramientas que permitirán obtener los paquetes necesarios para obtener los datos de los sensores XSENS MTi-G. Para ello, en el terminal se ejecutará el siguiente comando:

```
$ sudo apt-get install python-rosinstall python-rosdep
```

C.2. Instalación del simulador Gazebo

Para instalar la versión de Gazebo preparada para comunicarse con ROS se ejecutará el siguiente comando:

```
$ sudo apt-get install ros-fuerte-simulator-gazebo
```

Apéndice D

CÓDIGO FUENTE

D.1. Driver Xsens

D.1.1. xsens_node.cpp

Parte III

Otros documentos

D.2. Manual del sensor MTi-G

AAAAAAAAAAAAAAAAAAAAAAa