Crazyflie 2.1 - ROS (`uned_crazyflie_ros_pkg`)

Francisco J. Mañas-Álvarez

20 de abril de 2021

Índice

1. Introducción

Este repositorio[1] contiene los paquetes de ROS y ficheros de configuración para la teleoperación y simulación del dron crazyflie 2.1 en ROS, Gazebo y Matlab. Se busca obtener una herramienta Hardware-in-the-Loop que sea facilmente escalable y mantenible. A partir de esta herramienta, se pretende integrar estos robots en un sistema distribuido mayor con más variedad de robots.

1.1. Estructura

- **doc.** Contiene un fichero *.tex* que aborda más en detalle toda la información relacionada con el repositorio: esquemas de ROS, búsquedas bibliográficas, enlaces de interés, etc.
- **scripts.** Contiene aquellos ficheros auxiliares que no forman parte de ningún paquete de ROS. Por ejemplo, ficheros *.sh* para automatizar procesos repetitivos como la conversión de los ficheros *.bag* a txt o los scripts de Matlab para representar datasets.
- **submodules.** En este directorio están vinculados otros repositorios que se reutilizan, o se toman de base, para tareas ya abordadas por otros usuarios.
- **uned_crazyflie_config.** Paquete de ROS. Contiene aquellos elementos auxiliares para la configuración del entorno, así como los *.launch* para la ejecución en bloque de las diferentes estructuras del sistema.
- **uned_crazyflie_drone.** Paquete de ROS. Comprende los nodos propios desarrollados para la integración de drones en el sistema. Incluye toda la información asociada para su correcta puesta en marcha.
- **uned_crazyflie_test.** Paquete de ROS. Paquete en el que se incluyen todos los elementos destinados a realizar comprobaciones en el sistema de forma rápida. Por ejemplo los nodos *talker* y *listener* que se desarrollan al empezar a usar ROS, que en este caso se usan para comprobar la correcta comunicación entre máquinas en el sistema distribuido.

1.2. Crazyflie

El dron crazyflie 2.1, figura 1, se trata de un quadrotor que por su reducido tamaño entra dentro de la categoría de "nanoquadrotor". Su diseño está especialmente pensado para vuelo en interiores. A continuación se detallan sus especificaciones técnicas, disponibles también en la web del dron[2].

Figura 1: Crazyflie 2.1.

[1] Github: `https://github.com/FranciscoJManasAlvarez/uned_crazyflie_ros_pkg`
[2] Bitcraze: `https://store.bitcraze.io/products/crazyflie-2-1`

1.2.1. Especificaciones mecánicas

- Peso: $27\,[g]$
- Dimensiones: $92x92x29\,[mm]$

1.2.2. Radio

- 2.4GHz ISM band radio
- Increased range with 20 dBm radio amplifier, tested to ¿1 km range LOS with Crazyradio PA (environmentally dependent)
- Bluetooth Low Energy support with iOS and Android clients available
- Dual antenna support with both on board chip antenna and U.FL connector

1.2.3. Microcontroladores

- STM32F405 main application MCU (Cortex-M4, 168MHz, 192kb SRAM, 1Mb flash)
- nRF51822 radio and power management MCU (Cortex-M0, 32Mhz, 16kb SRAM, 128kb flash)
- uUSB connector
- On-board LiPo charger with 100mA, 500mA and 980mA modes available
- Full speed USB device interface
- Partial USB OTG capability (USB OTG present but no 5V output)
- 8KB EEPROM

1.2.4. IMU

- 3 axis accelerometer / gyroscope (BMI088)
- high precision pressure sensor (BMP388)

1.2.5. Especificaciones de vuelo

- Tiempo de vuelo con batería completa: 7 minutos
- Tiempo de carga hasta batería completa: 40 minutos
- Máxima carga recomendada: 15g

1.2.6. Conectores

- VCC (3.0V, max 100mA)
- GND
- VCOM (unregulated VBAT or VUSB, max 1A)
- VUSB (both for input and output)
- I2C (400kHz)
- SPI

- 2 x UART
- 4 x GPIO/CS for SPI
- 1-wire bus for expansion identification
- 2 x GPIO connected to nRF51

1.2.7. Modelado

El modelado se ha basado en el capítulo 4 del trabajo [1]. Para el modelado, se comprenderá la configuración del crazyflie en "X", ya que presenta menos inconvenientes en el caso de querer agregar sensores, como cámaras.

Para la obtención del modelo dinámico del sistema, se asumen tres hipótesis, que simplifican el modelo pero dan una buena aproximación para trabajar. Las simplificaciones sobre el dron son:

- Se comporta como un sólido-rígido sin deformaciones.
- Es simétrico, tanto en masa como en el sistema de propulsión (todos los motores presentan el mismo comportamiento).
- La masa es constante.

La transformación desde el centro de referencia del sistema global al dron está determinada por tres rotaciones sucesivas: Yaw (ψ) $\rightarrow$ Pitch (θ) $\rightarrow$ Roll (ϕ). La matriz de rotación resultante sería: COMPROBAR

$$R_i^b = \begin{pmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{pmatrix} \quad (1)$$

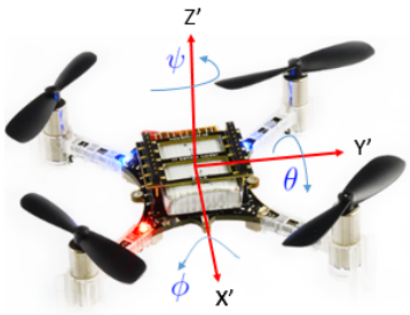

Figura 2: Ejes y ángulos.

En la siguiente tabla se representan las variables de estado que se emplean en este modelado

A continuación se realiza el modelado basado en espacios de estado para un quadrotor genérico. Los parámetros empleados en el proceso se muestran en la tabla **??**. El vector de estados está compuesto por cuatro vectores asociados a los 6 grados de libertad del dron y sus respectivas velocidades tal como se describe en la tabla 1.

1.3. Ecuaciones del equilibrio de fuerzas

1.4. Proyectos anteriores

En este apartado se describen los trabajos ya realizados sobre crazyflie que se han consultado y reutilizado parcialmente para la herramienta desarrollada.

Vector	Estado	Descripción
p	x	Posición X
	y	Posición Y
	z	Posición Z
Φ	ϕ	Ángulo Roll
	θ	Ángulo Pitch
	ψ	Ángulo Yaw
V	u	Velocidad Lineal X
	v	Velocidad Lineal Y
	w	Velocidad Lineal Z
ω	p	Velocidad Angular X
	q	Velocidad Angular Y
	r	Velocidad Angular Z

Tabla 1: Vector de estados.

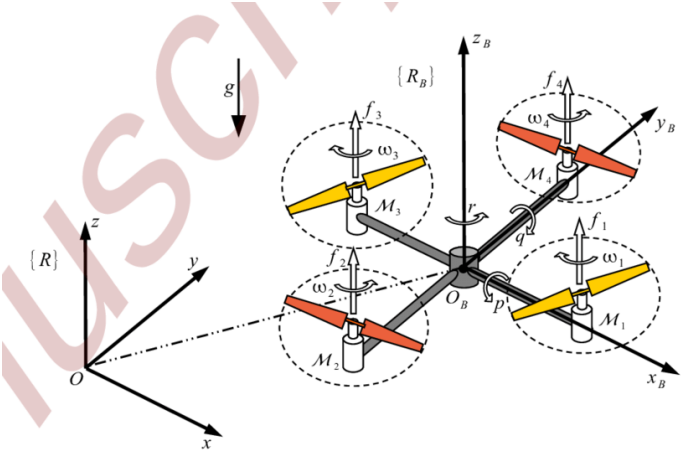

Figura 3: Esquema de un quadrotor.

CrazyS [3]

TO-DO

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod, erat ultrices hendrerit consequat, ligula nisi semper felis, id euismod ex erat eget diam. Vestibulum ut leo condimentum, ullamcorper orci id, suscipit arcu. Suspendisse suscipit purus tincidunt ex eleifend, sed tristique eros maximus. Donec vitae nisl congue, gravida orci vitae, consectetur sapien. Interdum et malesuada fames ac ante ipsum primis in faucibus. Duis ante orci, blandit rutrum urna in, mollis commodo lorem. Donec blandit, velit sed aliquam auctor, mi nunc lobortis arcu, in imperdiet sapien est vitae libero. Donec tincidunt quam ipsum, vitae rhoncus dolor efficitur id. Proin laoreet, ipsum quis vehicula condimentum, ex ipsum tincidunt est, bibendum iaculis enim est vel risus.

Crazyflie_ros [4]

TO-DO

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod, erat ultrices hendrerit consequat, ligula nisi semper felis, id euismod ex erat eget diam. Vestibulum ut leo condimentum,

[3] Github: `https://github.com/gsilano/CrazyS`
[4] Github: `https://github.com/whoenig/crazyflie_ros`

ullamcorper orci id, suscipit arcu. Suspendisse suscipit purus tincidunt ex eleifend, sed tristique eros maximus. Donec vitae nisl congue, gravida orci vitae, consectetur sapien. Interdum et malesuada fames ac ante ipsum primis in faucibus. Duis ante orci, blandit rutrum urna in, mollis commodo lorem. Donec blandit, velit sed aliquam auctor, mi nunc lobortis arcu, in imperdiet sapien est vitae libero. Donec tincidunt quam ipsum, vitae rhoncus dolor efficitur id. Proin laoreet, ipsum quis vehicula condimentum, ex ipsum tincidunt est, bibendum iaculis enim est vel risus.

2. Instalación

Se plantea inicialmente el desarrollo del sistema distribuido en ROS. El objetivo de desarrollo de la plataforma es implementar todo el sistema en ROS Noetic Ninjemys y Ubuntu 20.04 LTS (Focal Fossa) a fin de prolongar el mantenimiento y vigencia de la plataforma. Ambos tienen el mantenimiento previsto de 5 años. No obstante, el lanzamiento de estas versiones se ha realizado en el año 2020, por lo que, junto con el fin del mantenimiento de la versión 2.7 de Python, implica que mucho del material ya desarrollado para versiones anteriores, no se puede migrar con facilidad. Por tanto, se plantea la reutilización de gran parte del material ya disponible en la web, trabajando con parte del sistema en la actual y otra parte en la versión anterior, ROS Melodic Morenia y Ubuntu 18.04 LTS (Bionic Beaver).

2.1. ROS

Lo primero debe ser tener instalada la correspondiente versión de ROS para el sistema operativo del dispositivo (Noetic, Melodic). La máquina donde se ejecuten los paquetes reutilizados debe trabajar con ROS Melodic. No hay problema de compatibilidad en la interconexión de distintas máquinas siempre que los topics no presenten incompatibilidades entre versiones.

2.2. rosbridge_suite

La conexión común de todos los componentes de la red de ROS se realiza a través del paquete rosbridge_suite, instalado mediante el comando sudo apt-get install ros-¡rosdistro¿-rosbridge-suite (debe estar instalado previamente ROS en el dispositivo). Para la correcta identificación y conexión de cada máquina, se debe configurar en cada una los parámetros *ROS_MASTER_URI* y *ROS_HOSTNAME*, dados por la ip de cada dispositivo.

```
sudo nano ~/.bashrc
...
commentstyleexport ROS_MASTER_URI = http://xxx.xxx.x.xx:11311
commentstyleexport ROS_HOSTNAME = xxx.xxx.x.xx
```

Para el lanzamiento del paquete, se emplea el comando

```
roslaunch rosbridge_server rosbridge_websocket.launch
```

2.3. Matlab

Matlab debe disponer del toolbox de ROS instalado. En este caso, se trabaja con la versión de Matlab 2020b.

Se deben configurar los parámetros *ROS_MASTER_URI* y *ROS_HOSTNAME* para que pueda conectarse a la red que se ejecute en el dispositivo principal (ROS_MASTER_URI) y sea identificado dentro de la red. Estas acciones se llevan a cabo en línea de comandos mediante las instrucciones:

```
setenv('ROS_MASTER_URI','http://192.168.1.xx:11311')
setenv('ROS_HOSTNAME','192.168.1.xx')
```

La versión de Python que emplea el toolbox de Matlab es la 2.7. En principio no supone un problema porque no influye en el desempeño del resto de la red de dispositivos. Se puede descargar

esta versión desde la web oficial. Se debe configurar la versión de Python Matlab mediante el comando

```
pyversion folder
```

donde *folder* es el directorio donde se ha instalado previamente la versión de python. Esto se emplea para integrar posteriormente los mensajes no estándares que se emplean en el proyecto. El compilador que debe estar fijado en Matlab debe ser Microsoft Visual C++ 2017. Para realizar esta comprobación se puede ejecutar el comando

```
mex -setup cpp
```

Una vez asegurada la versión de python y del compilador se deben agregar las nuevas tipologías de mensajes al directorio de Matlab. Para ello, se deben ejecutar los siguientes comandos:

```
folderpath = 'C:\folder con los nuevos mensajes\'
rosgenmsg(folderpath)
addpath('C:\folder con los nuevos mensajes\matlab_msg_gen_ros1\win64\install\m')
savepath
clear classes
rehash toolboxcache
rosmsg list
```

Recordar ejecutar siempre el comando rosshutdown al final del script para evitar dejar el nodo en el aire y al proncipio por si se nos ha olvidado cerrarlo anteriormente, que no de problemas.

2.4. Dependencias

- **Octomap**. TO-DO: Especificar los paquetes que dependen

  ```
  sudo apt-get install ros-<rosdistro>-octomap
  ```

- **Xacro**. TO-DO: Especificar los paquetes que dependen

  ```
  sudo apt-get install ros-<rosdistro>-xacro
  ```

- **Joy**. Paquete para realizar la lectura del joystick para teleoperación.

  ```
  sudo apt-get install ros-<rosdistro>-joystick-drivers
  ```

- **mav_comm**. TO-DO. Este paquete se emplea como complemento a gran parte de los paquetes ya desarrollados de Crazyflie y es compatible con ambas versiones de ROS y Ubuntu por lo que se puede alojar en el espacio de trabajo del dispositivo y compilarlo como cualquier otro paquete.

2.5. Ubuntu 18.04 - ROS Melodic Morenia

A continuación se detalla la instalación en el entorno de trabajo de ROS para el paquete CrazyS, de donde se reutiliza gran parte de la arquitectura de simulación.

```
mkdir -p catkin_ws/src
cd crazyflie/src
catkin_init_workspace
cd ..
catkin init
cd src
git clone https://github.com/gsilano/CrazyS.git
git clone https://github.com/gsilano/mav_comm.git

rosdep install --from-paths src -i
sudo apt install ros-melodic-rqt-rotors ros-melodic-rotors-comm \
ros-melodic-mav-msgs ros-melodic-rotors-control

sudo apt install ros-melodic-rotors-gazebo ros-melodic-rotors-evaluation \
```

```
ros-melodic-rotors-joy-interface

sudo apt install ros-melodic-rotors-gazebo-plugins ros-melodic-mav-planning-msgs \
ros-melodic-rotors-description ros-melodic-rotors-hil-interface

rosdep update
catkin build

echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

En la sección de instalación de CrazyS se detalla este proceso y posibles soluciones en caso de fallos con gazebo (como que no se inicie la simulación).

2.6. Ubuntu 20.04 - ROS Noetic Ninjemys

La configuración del entorno de trabajo para el paquete desarrollado se muestra a continuación.

```
mkdir -p crazyflie_ws/src
cd crazyflie/src
git clone https://github.com/FranciscoJManasAlvarez/uned_crazyflie_ros_pkg
cd uned_crazyflie_ros_pkg
git submodules init
git submodules update
cd ..
git clone https://github.com/ethz-asl/mav_comm.git
cd ../..
catkin build
echo "source devel/setup.bash" >> ~/.bashrc
```

Este paquete compila correctamente en ambas versiones de ROS y Ubuntu.

3. Simulador

La simulación se puede hacer tanto de forma exclusiva en c++ como empleando Matlab.

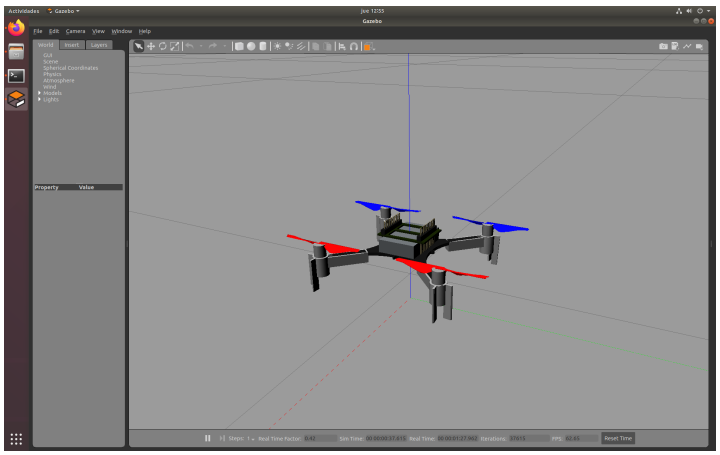

Figura 4: Crazyflie 2.1 en Gazebo

3.1. C++

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod, erat ultrices hendrerit consequat, ligula nisi semper felis, id euismod ex erat eget diam. Vestibulum ut leo condimentum, ullamcorper orci id, suscipit arcu. Suspendisse suscipit purus tincidunt ex eleifend, sed tristique

eros maximus. Donec vitae nisl congue, gravida orci vitae, consectetur sapien. Interdum et malesuada fames ac ante ipsum primis in faucibus. Duis ante orci, blandit rutrum urna in, mollis commodo lorem. Donec blandit, velit sed aliquam auctor, mi nunc lobortis arcu, in imperdiet sapien est vitae libero. Donec tincidunt quam ipsum, vitae rhoncus dolor efficitur id. Proin laoreet, ipsum quis vehicula condimentum, ex ipsum tincidunt est, bibendum iaculis enim est vel risus.

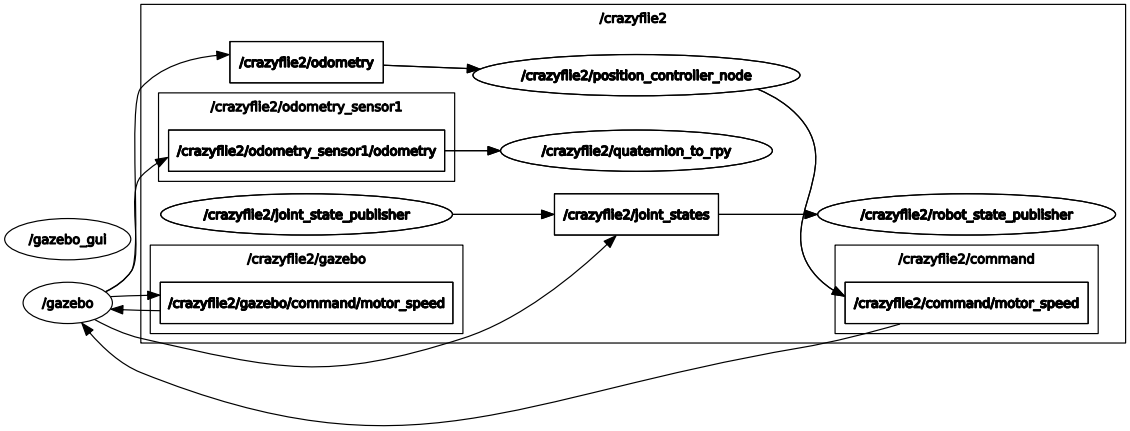

Figura 5: Captura de rqt_graph durante la ejecución de la simulación C++ (simple).

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod, erat ultrices hendrerit consequat, ligula nisi semper felis, id euismod ex erat eget diam. Vestibulum ut leo condimentum, ullamcorper orci id, suscipit arcu. Suspendisse suscipit purus tincidunt ex eleifend, sed tristique eros maximus. Donec vitae nisl congue, gravida orci vitae, consectetur sapien. Interdum et malesuada fames ac ante ipsum primis in faucibus. Duis ante orci, blandit rutrum urna in, mollis commodo lorem. Donec blandit, velit sed aliquam auctor, mi nunc lobortis arcu, in imperdiet sapien est vitae libero. Donec tincidunt quam ipsum, vitae rhoncus dolor efficitur id. Proin laoreet, ipsum quis vehicula condimentum, ex ipsum tincidunt est, bibendum iaculis enim est vel risus.

In hac habitasse platea dictumst. Aenean justo tortor, congue non congue ut, egestas a sem. Mauris egestas diam id nulla dictum eleifend. Nulla aliquam vulputate dapibus. Morbi tellus dolor, ornare sed iaculis ac, sodales sed nisl. Sed elementum, ligula eget venenatis congue, nisi nulla finibus mauris, sed hendrerit dui odio et massa. Cras nec enim ut nunc rhoncus cursus. Vestibulum ex ipsum, commodo a dolor a, bibendum fermentum lacus. Praesent et ultricies sem. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. In iaculis elit eget mattis auctor. Duis vel gravida magna. Cras egestas cursus rutrum. Duis consectetur et ex et commodo. Suspendisse dignissim magna ac mi porta bibendum. Praesent sollicitudin aliquet malesuada.

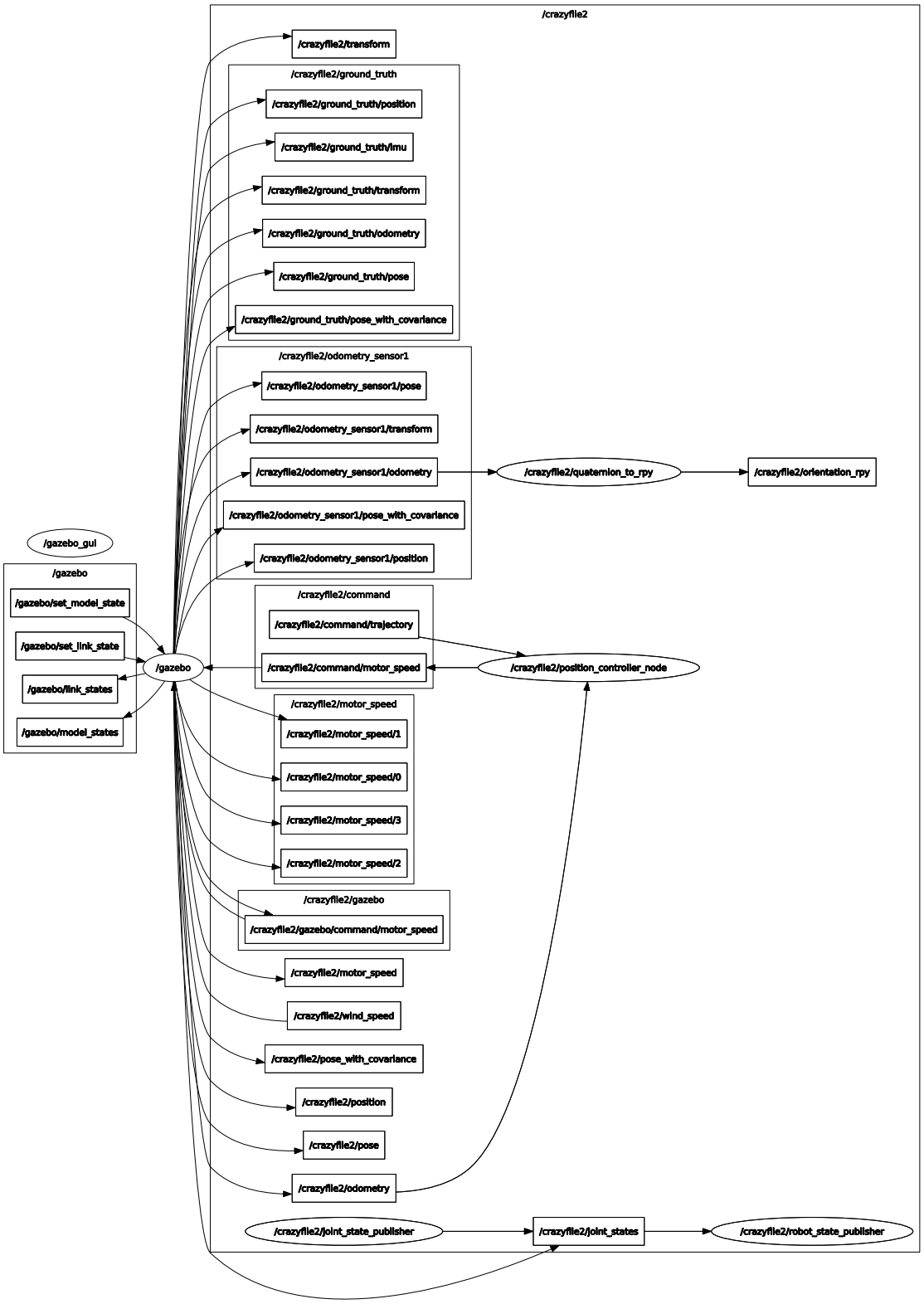

Figura 6: Captura de rqt_graph durante la ejecución de la simulación C++.

3.2. Matlab

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod, erat ultrices hendrerit consequat, ligula nisi semper felis, id euismod ex erat eget diam. Vestibulum ut leo condimentum, ullamcorper orci id, suscipit arcu. Suspendisse suscipit purus tincidunt ex eleifend, sed tristique eros maximus. Donec vitae nisl congue, gravida orci vitae, consectetur sapien. Interdum et malesuada fames ac ante ipsum primis in faucibus. Duis ante orci, blandit rutrum urna in, mollis commodo lorem. Donec blandit, velit sed aliquam auctor, mi nunc lobortis arcu, in imperdiet sapien est vitae libero. Donec tincidunt quam ipsum, vitae rhoncus dolor efficitur id. Proin laoreet, ipsum quis vehicula condimentum, ex ipsum tincidunt est, bibendum iaculis enim est vel risus.

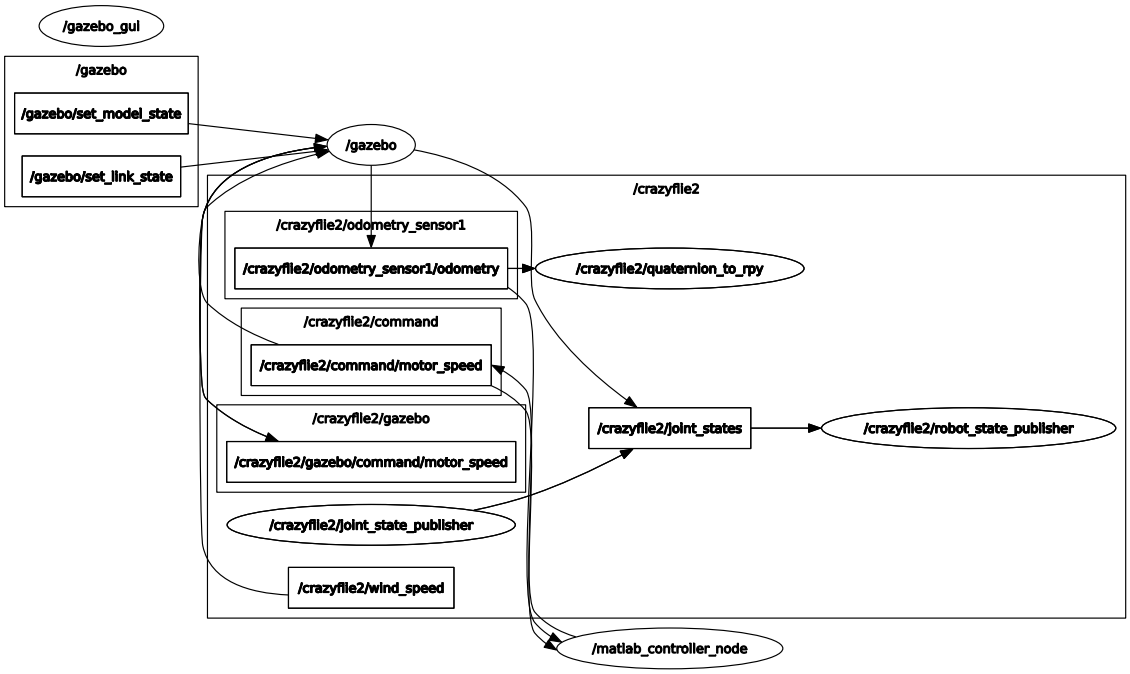

Figura 7: Captura de rqt_graph durante la ejecución de la simulación Matlab (simple).

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod, erat ultrices hendrerit consequat, ligula nisi semper felis, id euismod ex erat eget diam. Vestibulum ut leo condimentum, ullamcorper orci id, suscipit arcu. Suspendisse suscipit purus tincidunt ex eleifend, sed tristique eros maximus. Donec vitae nisl congue, gravida orci vitae, consectetur sapien. Interdum et malesuada fames ac ante ipsum primis in faucibus. Duis ante orci, blandit rutrum urna in, mollis commodo lorem. Donec blandit, velit sed aliquam auctor, mi nunc lobortis arcu, in imperdiet sapien est vitae libero. Donec tincidunt quam ipsum, vitae rhoncus dolor efficitur id. Proin laoreet, ipsum quis vehicula condimentum, ex ipsum tincidunt est, bibendum iaculis enim est vel risus.

In hac habitasse platea dictumst. Aenean justo tortor, congue non congue ut, egestas a sem. Mauris egestas diam id nulla dictum eleifend. Nulla aliquam vulputate dapibus. Morbi tellus dolor, ornare sed iaculis ac, sodales sed nisl. Sed elementum, ligula eget venenatis congue, nisi nulla finibus mauris, sed hendrerit dui odio et massa. Cras nec enim ut nunc rhoncus cursus. Vestibulum ex ipsum, commodo a dolor a, bibendum fermentum lacus. Praesent et ultricies sem. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. In iaculis elit eget mattis auctor. Duis vel gravida magna. Cras egestas cursus rutrum. Duis consectetur et ex et commodo. Suspendisse dignissim magna ac mi porta bibendum. Praesent sollicitudin aliquet malesuada.

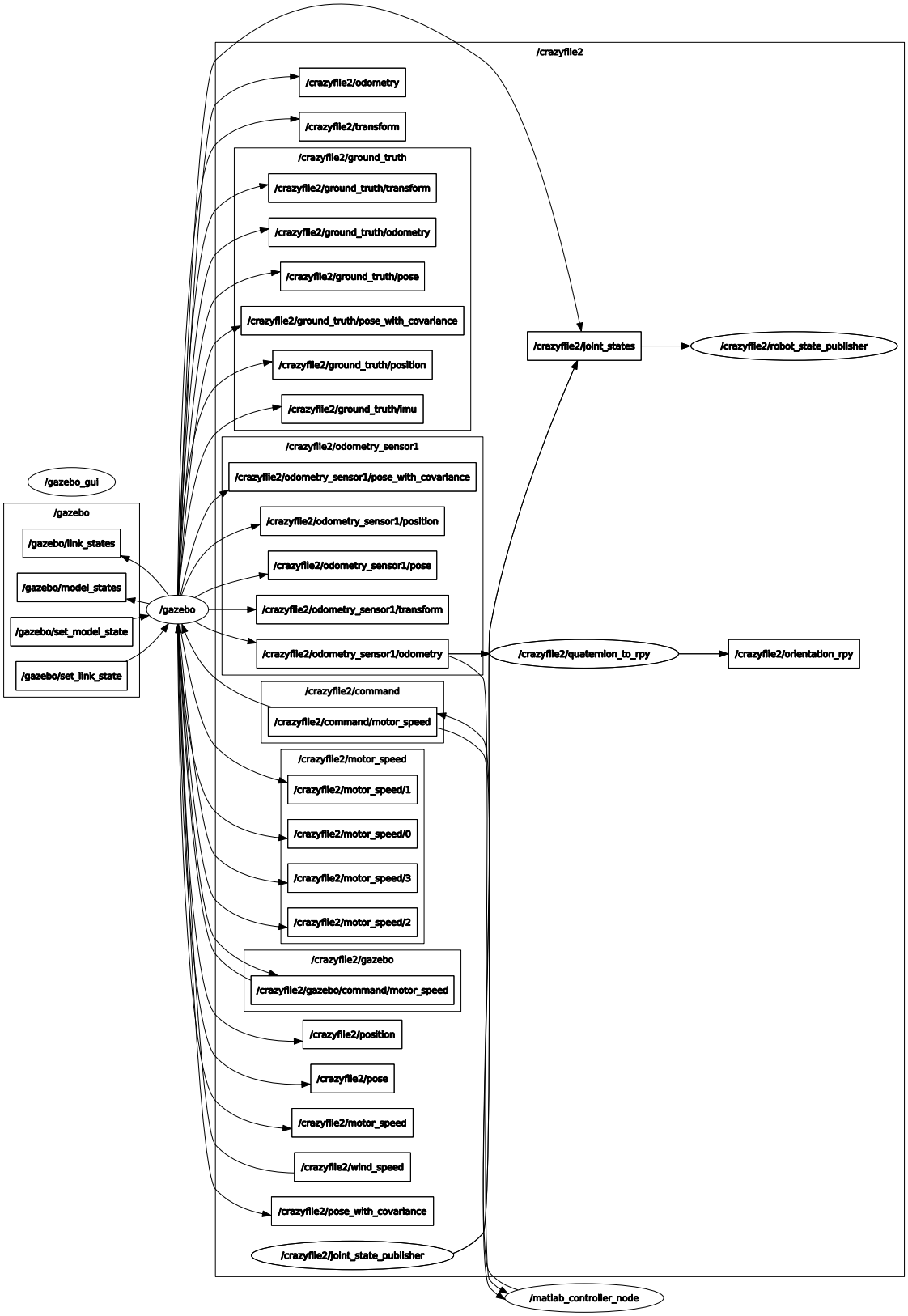

Figura 8: Captura de rqt_graph durante la ejecución de la simulación Matlab.

4. Hardware-in-the-Loop

4.1. Crazyflie_ros

TO-DO

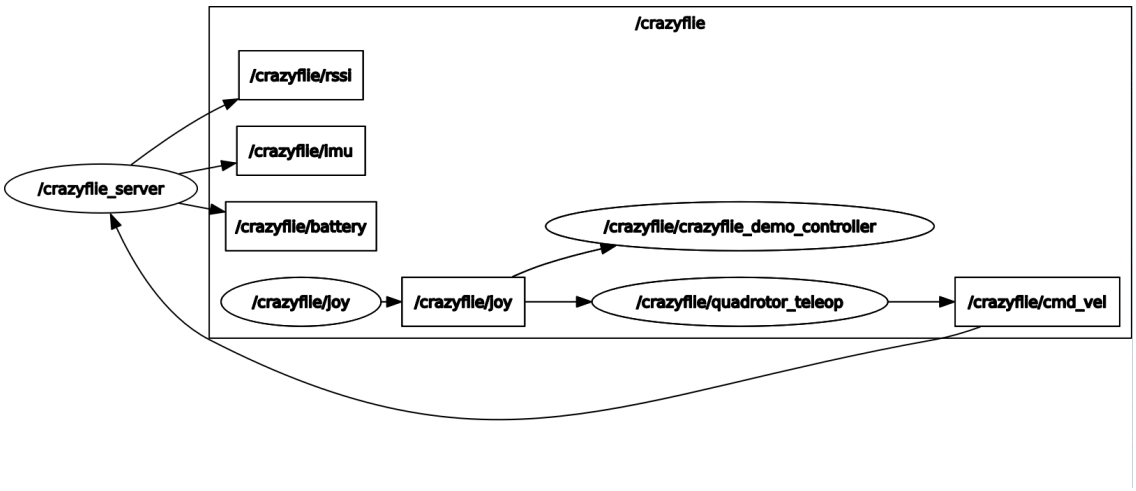

Figura 9: rosgraph.

Referencias

[1] Carlos Luis and Jérôme Le Ny. Design of a trajectory tracking controller for a nanoquadcopter. *arXiv preprint arXiv:1608.05786*, 2016.