Khepera IV - ROS (`uned_kheperaIV_ros_pkg`)

Francisco J. Mañas-Álvarez

13 de mayo 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.

2. Khepera IV

El Khepera IV, figura 1, se trata de un robot diferencial compacto. Su diseño está especialmente pensado para vuelo en interiores. A continuación se detallan sus especificaciones técnicas, disponibles también en la web del fabricante K-team[2]. También se encuentra disponible online el manual de usuario[3].

Figura 1: Khepera IV.

2.1. Características

2.1.1. Especificaciones mecánicas

- Peso: 540$[g]$
- Carga Máxima: 2000$[g]$
- Dimensiones: 140$\varnothing x$58$[mm]$

[1]Github: `https://github.com/FranciscoJManasAlvarez/uned_crazyflie_ros_pkg`
[2]Khepera. K-team: `https://www.k-team.com/khepera-iv`
[3]Manual Khepera: `http://ftp.k-team.com/KheperaIV/software/Gumstix%20COM%20Y/UserManual/Khepera%20IV%20User%20Manual%204.x.pdf`

2.1.2. Hardware

- Núcleo de Linux que se ejecuta en un procesador ARM Cortex-A8 de 800 MHz con núcleo DSP de punto fijo C64x y un microcontrolador adicional para la gestión de periféricos.
- RAM 256 MB
- Flash 512MB más 8GB adicionales para datos
- Compilador GNU C / C ++, para aplicaciones nativas integradas.
- Admite Python 2.7.9

2.1.3. Sensores

- 12 sensores infrarrojos (8 de proximidad y luz ambiental y 4 de proximidad al suelo para aplicaciones de seguimiento de línea y prevención de caídas) con un alcance desde 2 mm hasta 25 cm. Modelo: TCRT5000 de Vishay Telefunken[4].
- 5 sensores de ultrasonido con rango de 25 cm a 2 metros. Tiempo de lectura 10ms. Modelo: 400PT12B de Prowave[5].
- Acelerómetro y Giroscopio de 3 ejes. Modelo: LSM330DLC de STMicroelectronics[6].
- Micrófono integrado $(100 - 10000[Hz])$. Modelo: PU0414HR5H-SB de Knowles[7].
- Cámara a color integrada (752x480 píxeles, 30FPS). Distancia focal de 2.1mm. Campo de visión: 131^{o} horizontal, 101^{o} vertical.

2.1.4. Alimentación y batería

- Alimentación: 9 V a 2,5 A
- Autonomía: 7 horas aprox.
- Batería: polímero de litio de 7,4 V, 3400 mAh

2.1.5. Comunicaciones

- 1x host USB 2.0 (500mA)
- 1x dispositivo USB 2.0
- WiFi 802.11 b / g
- Bluetooth 2.0 EDR
- Se pueden agregar módulos de expansión al robot usando el bus KB-250.

2.1.6. Movimiento

- 2 motores DC con escobillas con codificadores incrementales (aproximadamente 147 pulsos por mm de movimiento del robot) y caja de cambios
- Máx. 1 m / s en lazo abierto y 0,8 m / s con controlador de velocidad PID predeterminado de fábrica. Mínimo 0,003 m / s con controlador de velocidad PID predeterminado de fábrica

[4]TCRT5000 Datasheet: `https://datasheet.octopart.com/TCRT5000-Vishay-datasheet-547230.pdf`
[5]400PT12B Datasheet: `http://www.prowave.com.tw/pdf/T400PT12.PDF`
[6]LSM330DLC Datasheet: `https://www.mouser.es/datasheet/2/389/en_DM00037200-1920927.pdf`
[7]PU0414HR5H-SB Datasheet: `https://datasheetspdf.com/pdf-file/916686/KnowlesElectronics/SPU0414HR5H-SB/1`

2.1.7. Otros

- 3 imanes situados en la parte superior
- 3 LEDs RGB

2.2. Proyectos anteriores

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

Proyectos ...

TO-DO

2.3. Modelado

El modelado de los Khepera IV se realiza atendiendo tanto al enfoque dinámico como cinemático. El texto de referencia para determinar el modelado genérico de los robots de tipo diferencial se encuentra en el capítulo 2 de la tesis de Rafael Socas Gutiérrez [2]. Los Khepera, son robots móviles diferenciales con ruedas. Para mantener la estabilidad, disponen de dos esferas pasivas.

Modelo cinemático

Este modelo determina el movimiento del robot a partir de sus características geométricas y sus restricciones. Se establecen las siguientes suposiciones consideradas en el modelado:

- Las ruedas giran sobre la superficie sin efectos de deslizamiento.
- El eje de giro del robot es perpendicular al plano X-Y
- El centro de gravedad se sitúa en el punto Q de la figura X

TO-DO. Figura del esquema

Considerando v_r como la velocidad lineal de la rueda derecha, v_l como la velocidad lineal de la rueda izquierda y L la distancia entre las ruedas, las velocidades lineales y angulares del robot se puede calcular según la ecuación 1.

$$v_Q = \frac{1}{2}\left(v_r + v_l\right) \quad ; \quad \dot{\varphi} = \frac{\left(v_r - v_l\right)}{L} \tag{1}$$

Al partir del supuesto de ausencia de deslizamiento en las ruedas, las velocidades de cada rueda cumplen la relación:

$$v_r = r \cdot \dot{\theta}_r \quad ; \quad v_l = r \cdot \dot{\theta}_l \tag{2}$$

Conocidas estas relaciones, se puede determinar la velocidad en el plano X-Y.

$$\dot{x}_Q = \frac{r}{2}\left(\dot{\theta}_r \cdot \cos\left(\varphi\right) + \dot{\theta}_l \cdot \cos\left(\varphi\right)\right) \tag{3}$$

$$\dot{y}_Q = \frac{r}{2}\left(\dot{\theta}_r \cdot \sin\left(\varphi\right) + \dot{\theta}_l \cdot \sin\left(\varphi\right)\right) \tag{4}$$

$$\dot{\varphi} = \frac{r}{L}\left(\dot{\theta}_r - \dot{\theta}_l\right) \tag{5}$$

De esta forma, el **modelo cinemático diferencial directo** se puede expresar como:

$$\dot{p} = J\dot{q} \rightarrow \dot{p} = \begin{pmatrix} \frac{r}{2}\cos\left(\varphi\right) \\ \frac{r}{2}\sin\left(\varphi\right) \\ \frac{r}{L} \end{pmatrix} \cdot \dot{\theta}_r + \begin{pmatrix} \frac{r}{2}\cos\left(\varphi\right) \\ \frac{r}{2}\sin\left(\varphi\right) \\ -\frac{r}{L} \end{pmatrix} \cdot \dot{\theta}_l \tag{6}$$

donde $\dot{p} = \begin{pmatrix} \dot{x}_Q \\ \dot{y}_Q \\ \dot{\varphi} \end{pmatrix}$, $\dot{q} = \begin{pmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{pmatrix}$ y J es el jacobiano del robot diferencial.

Puesto que la matriz Jacobiana no es invertible, para obtener $\dot{q}$ a partir de modelo cinemático diferencial directo, se debe emplear la matriz pseudoinversa (de Moore-Penrose) que establece su valor según la siguiente expresión:

$$J^+ = \left(J^T J\right)^{-1} J^T \tag{7}$$

Otra forma de calcular J^+ se puede hacer despejando debidamente de las ecuaciones de que relacionan la velocidad lineal de cada rueda con la velocidad del dron.

$$\begin{aligned} r\dot{\theta}_r &= \dot{x}_Q \cos(\varphi) + \dot{y}_Q \sin(\varphi) + \tfrac{L}{2}\dot{\varphi} \\ r\dot{\theta}_l &= \dot{x}_Q \cos(\varphi) + \dot{y}_Q \sin(\varphi) - \tfrac{L}{2}\dot{\varphi} \end{aligned} \tag{8}$$

De esta forma, el **método cinemático diferencial inverso** se obtiene según la siguiente expresión.

$$\dot{q} = J^+ \dot{p} \rightarrow \begin{pmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{pmatrix} = \frac{1}{r} \begin{pmatrix} \cos(\varphi) & \sin(\varphi) & \frac{L}{2} \\ \cos(\varphi) & \sin(\varphi) & -\frac{L}{2} \end{pmatrix} \begin{pmatrix} \dot{x}_Q \\ \dot{y}_Q \\ \dot{\varphi} \end{pmatrix} \tag{9}$$

Finalmente, el radio de curvatura instantáneo R se obtiene mediante la siguiente expresión.

$$R = \frac{v_Q}{\dot{\varphi}} = \frac{L}{2}\left(\frac{v_r + v_l}{v_r - v_l}\right) \tag{10}$$

2.3.1. Modelo dinámico

TO-DO

3. Control

En esta sección se documentarán los controladores propuestos y operativos que se han implementado, así como sus resultados en Matlab, Gazebo y experimentos reales. A continuación se muestra el esquema de control genérico implementado.

3.1. PID continuo

TO-DO

$$C(s) = \frac{U(s)}{E(s)} = K_p + K_i \cdot \frac{1}{s} + K_d \cdot \frac{N}{1 + N \cdot \frac{1}{s}} \tag{11}$$

3.2. PID discreto

En el caso del controlador PID discreto, se toma como referencia los valores del caso continuo. TO-DO. Para la discretización de los controladores, se ha estimado que la aproximación más adecuada es la trapezoidal (*Tustin's approximation* ó *the bilinear transformation*) [3], ecuación 13.

$$U(s) = \left(K_p + K_i \cdot \frac{1}{s} + K_d \cdot \frac{N}{1 + N \cdot \frac{1}{s}}\right) E(s) \rightarrow u(t) = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{de(t)}{dt} \tag{12}$$

$$s = \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} \rightarrow z = e^{sT} \approx \frac{1 + (sT/2)}{1 - (sT/2)} \tag{13}$$

$$C(z) = \frac{U(z)}{E(z)} = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-1}} \begin{cases} q_0 &= K_p + \frac{K_i T}{2} + \frac{K_d}{T} \\ q_1 &= -K_p + \frac{K_i T}{2} - 2\frac{K_d}{T} \\ q_2 &= \frac{K_d}{T} \end{cases} \tag{14}$$

3.3. MPC

TO-DO

3.4. Control basado en eventos

TO-DO

4. Instalación

TO-DO

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).

4.1. 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.

4.2. 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
```

4.3. 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.

4.4. Dependencias

TO-DO

- **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.

4.5. Ubuntu 20.04 - ROS Noetic Ninjemys

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

```
mkdir -p khepera_ws/src
cd khepera_ws/src
git clone https://github.com/FranciscoJManasAlvarez/uned_kheperaIV_ros_pkg
cd uned_kheperaIV_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.

5. Simulador

TO-DO

5.1. C++

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.

5.2. Matlab

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.

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.

6. Hardware-in-the-Loop

TO-DO

Referencias

[1] A. Barrientos, L. Felipe Peñín, C. Balaguer, and R. Aracil. Fundamentos de robótica, 2007.

[2] Rafael Socas Gutiérrez. Estrategias de control basadas en eventos aplicadas a robots móviles. 2017.

[3] Björn Wittenmark, Karl Johan Åström, and Karl-Erik Årzén. Computer control: An overview. *IFAC Professional Brief*, 1:2, 2002.