TMuLE: Un multiplexor de terminal para ROS2

Manuel Fernández Carmona

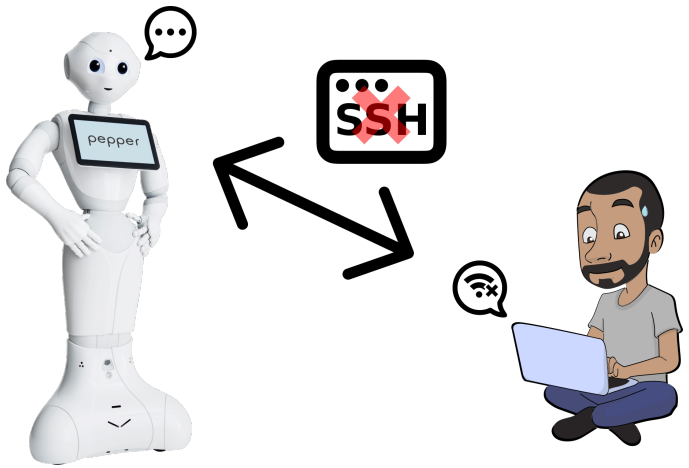
Universidad de Málaga
mfcarmona@uma.es

ROSCon Madrid,
Fuenlabrada, Madrid, España
28 Sept. 2023

El problema

La persistencia de las sesiones:

- Es habitual que cuando se trabaja con un robot nos conectemos a él.
- Pero la red inalámbrica no siempre es perfecta.
- ¿Cómo garantizar la persistencia?



TMuLE

Qué es

- Gestor de configuración de sesiones de `tmux`.
- `tmux` es un multiplexor de terminal

Por qué usarlo

- permite configuraciones complejas de las sesiones.
- mínima huella de memoria
- ”casi” sin dependencias

Cómo instalar TMuLE

Prerrequisitos

```
sudo apt install tmux
```

Que depende de: `libc6, libevent-core, libtinfo6, libutempter0`

Instalación

```
pip install tmule
```

Que depende de: `autobahn, libtmux, psutil, pyyaml, twisted, web.py`

Los fuentes están disponibles en
`https://github.com/marc-hanheide/TMuLE`

Control básico

TMuLE usa un archivo `yaml` para configurar la sesión de `tmux`. Posee la siguiente sintaxis básica:

```
tmule --config ARCHIVO.yaml COMANDO
```

Hay cuatro comandos básicos:

- `launch`: lanzar sesión
- `stop`: detener sesión
- `relaunch`: reactivar sesión
- `terminate`: cerrar sesión

Manipulando la sesión: tmux

Una vez ejecutado `launch`, tendremos una sesión de `tmux`.
Para acceder a ella:

```
tmux a -t NOMBRE_SESION
```

Y una vez dentro, podremos:

- [C-b c] crear ventana
- [C-b NUM] movernos a la ventanta NUM
- [C-b ”] crear un panel debajo del actual
- [C-b %] crear un panel a la derecha del actual
- [C-b $\leftarrow\rightarrow$] movernos entre paneles
- [C-b z] maximizar/restaurar un panel
- [C-b d] desconectarnos de la sesión
- ...

El libro ”The Tao of tmux” lo explica en profundidad.

Configurando la sesión: el archivo yaml

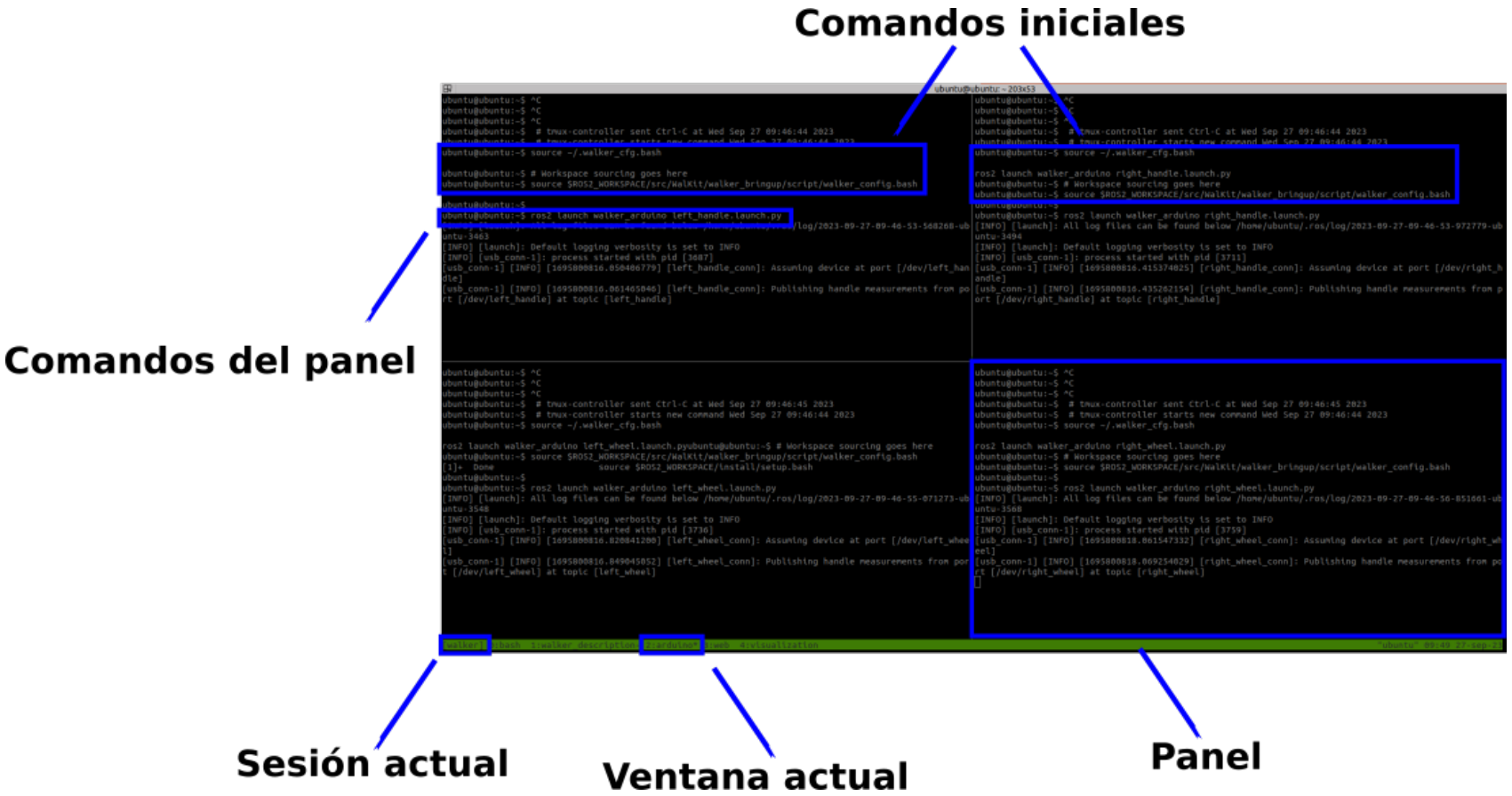
```yaml
session: walker
init_cmd: |
  source ~/.walker_cfg.bash
  # Workspace sourcing goes here
  source $ROS2_WORKSPACE/src/WalKit/walker_bringup/script/walker_config.bash
windows:
  - name: walker_description
    tags: ['core']
    panes:
      - ros2 launch walker_description walker_state_publisher.launch.py
  - name: arduino
    tags: ['core','sensors']
    panes:
      - ros2 launch walker_arduino left_handle.launch.py
      - ros2 launch walker_arduino right_handle.launch.py
      - ros2 launch walker_arduino left_wheel.launch.py
      - ros2 launch walker_arduino right_wheel.launch.py
  - name: web
    tags: ['web']
    panes:
      - ros2 launch rosbridge_server rosbridge_websocket_launch.xml
      - cd ~/workspace/walker_ws/src/WalKit/walker_web_gui/gui; python3 -m http.server 8080
  - name: visualization
    tags: ['visualization']
    panes:
      - ros2 launch walker_step_detector plot_steps.launch.py
      - ros2 run walker_plot plot_forces
```

Configurando la sesión: el archivo yaml

Tenemos los siguientes elementos en el archivo:

- `session`: nombre de la sesión de `tmux`.
- `init_cmd`: comandos ejecutados al iniciar los paneles.
- `windows`: lista de ventanas que tendra la sesión.
  Tiene tres secciones:
  - `name`: nombre descriptivo para tmux.
  - `tags`: etiquetas para administrar las ventanas.
  - `panes`: cada línea de comandos es un panel diferente.

TMuLE en acción



Las etiquetas

Permiten un control fino sobre las ventanas que se ejecutan cuando ejecutamos el comando `launch`.
P.e. el comando:

```
tmule --config walker.yaml launch --tag core
```

Solo lanzaría aquellas ventanas etiquetadas como `''core''`.

Otros comandos

Existen otros comandos muy interesantes:

```
tmule --config ARCHIVO.yaml COMANDO_AVANZADO
```

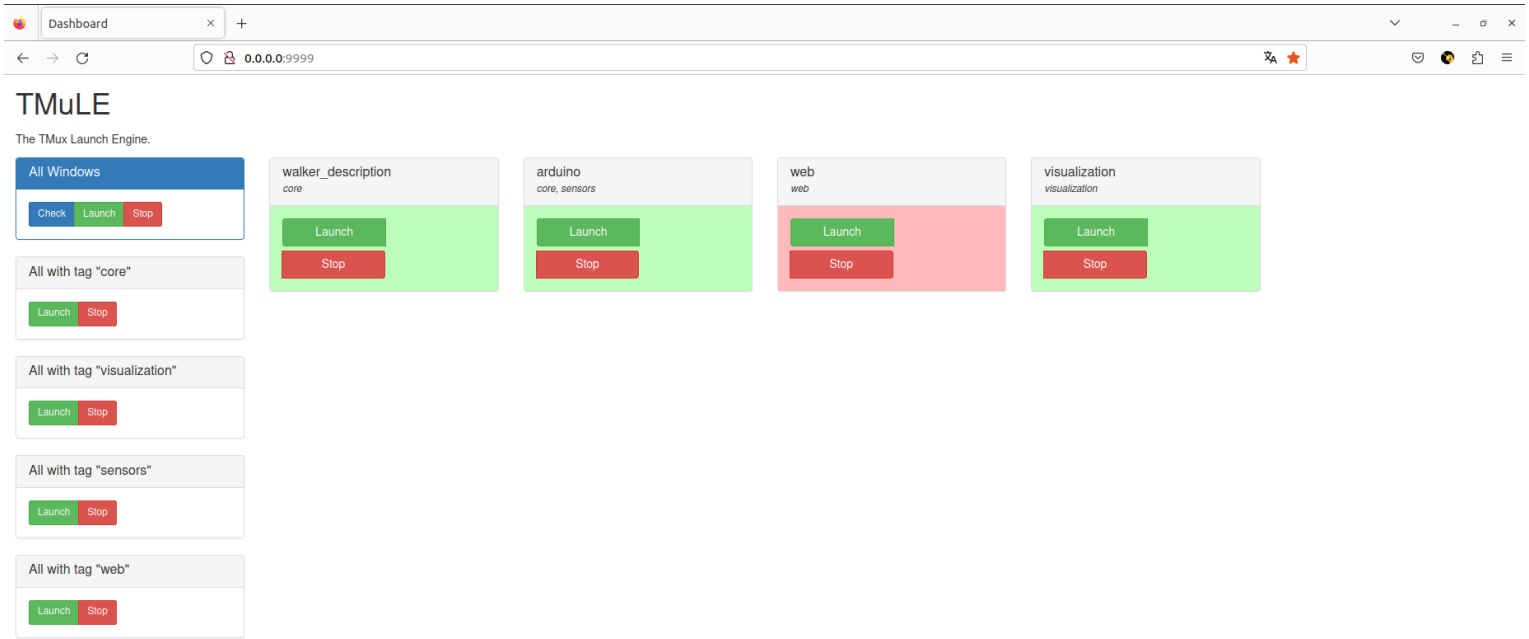
Comandos avanzados:

- `server`: crea un servidor web.
- `pids`: devuelve los pid de los procesos lanzados por TMuLE.
- `running`: devuelve `true` si están corriento los procesos de una ventana específica (`-w NOMBRE_VENTANA`).

comando `server`

Crea un servidor web en el puerto **9999**. Con él podemos:

- Lanzar o detener las ventanas individualmente, por etiquetas o globalmente.
- Comprobar su estado de ejecución.

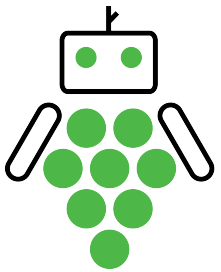


Este comando es ideal para un script de arranque.

Proyectos

Proyectos Europeos

- Iliad, project ID 732737: logística colaborativa.
- Bacchus, project ID 871704: agricultura de precisión.



Proyectos Nacionales

- SAVIA: Sistema de Autonomía Variable para movIlidad Asistida, Ref: RTI2018-096701-B-C21: robótica asistiva.
- Control colaborativo para interacción física empática entre robot y humano, Ref:PID2021-127221OB-I00.

Competiciones de robótica

RoboCup@Home:

- SPQReL: Social Standard Platform League[1].
- LCASTOR: Open Platform League.

[1] y en la European Robotics League Service Robots

Gracias por vuestra atención.
¿Preguntas?

TMuLE: Un multiplexor de terminal para ROS2

Manuel Fernández Carmona

Universidad de Málaga
mfcarmona@uma.es

ROSCon Madrid,
Fuenlabrada, Madrid, España
28 Sept. 2023

Este trabajo ha sido financiado por el ministerio de Ciencia e Innovación a través del proyecto de Investigación PID2021-127221OB-I00.

UNIVERSIDAD DE MÁLAGA | uma.es