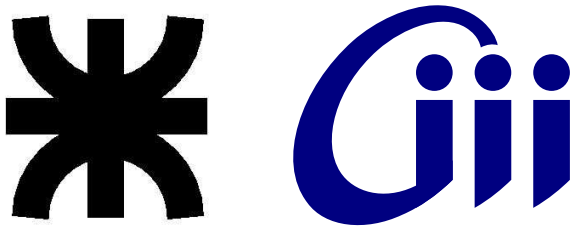
Entorno de desarrollo Player/Stage en el robot RoMAA

Gonzalo F. Perez Paina

Centro de Investigación en Informática para la Ingeniería
Universidad Tecnológica Nacional, F.R.C.
http://ciii.frc.utn.edu.ar
Córdoba, Argentina



Noviembre 2010

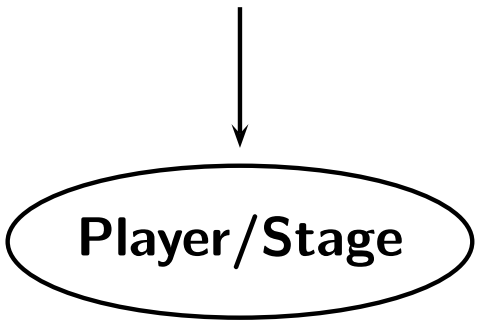
Contenido

Necesidad de un entorno de desarrollo de robótica

Herramientas y librerías flexibles de programación y simulación...

... no solo de la plataforma móvil...

... sino también de diferentes sensores y actuadores utilizados en robótica móvil



Los Entornos de Desarrollo de Robótica permiten:

- Acceso abstracto a dispositivos → Flexibilidad con el hardware
- Agregar nuevos algoritmos → Reutilización de código
- Simulación

Entorno de desarrollo Player/Stage

Originalmente

Desarrollado en el laboratorio de investigación de robótica de la *University of Southern California* (USC, Robotics Research Lab) por Brian P. Gerkey y Richard T. Vaughan.

En la actualidad

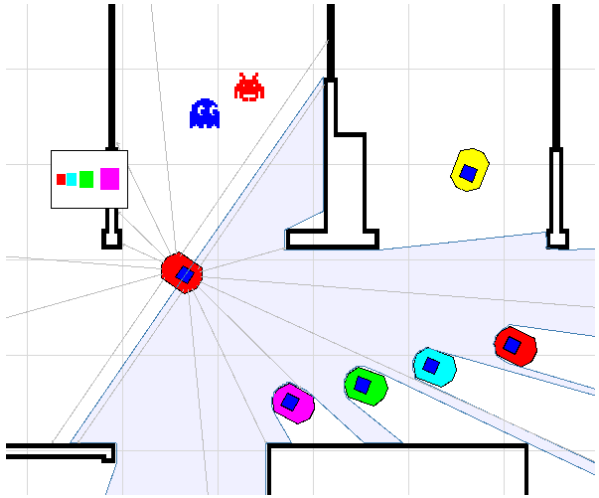
Es un proyecto activo de `sourceforge.net` usado por un gran número de investigadores alrededor del mundo - `http://playerstage.sourceforge.net/`

All the world's a stage; and all the men and women merely players, from As You Like It by William Shakespeare

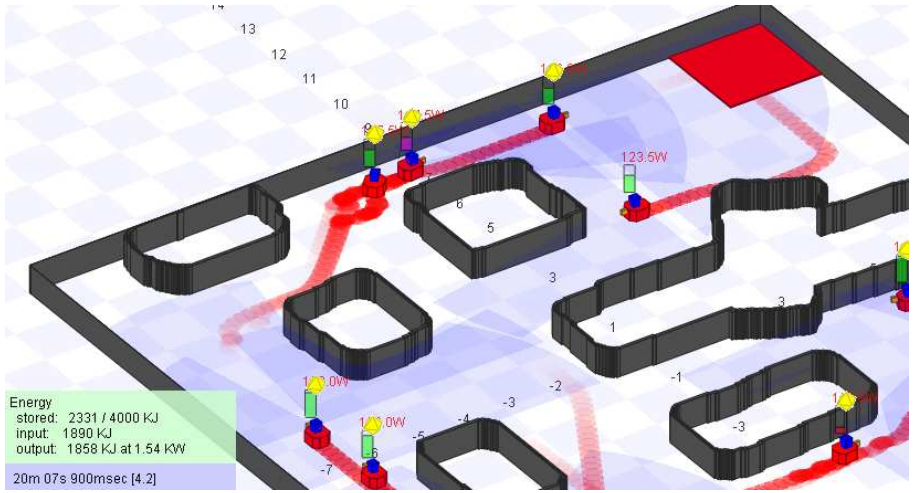
Usado en 27 países, en >150 laboratorios. Ej. China(6), Japón(2), Alemania(5), España(10), UK(6), USA(30), Australia(4), Brazil(2), etc.

Player y el simulador Stage

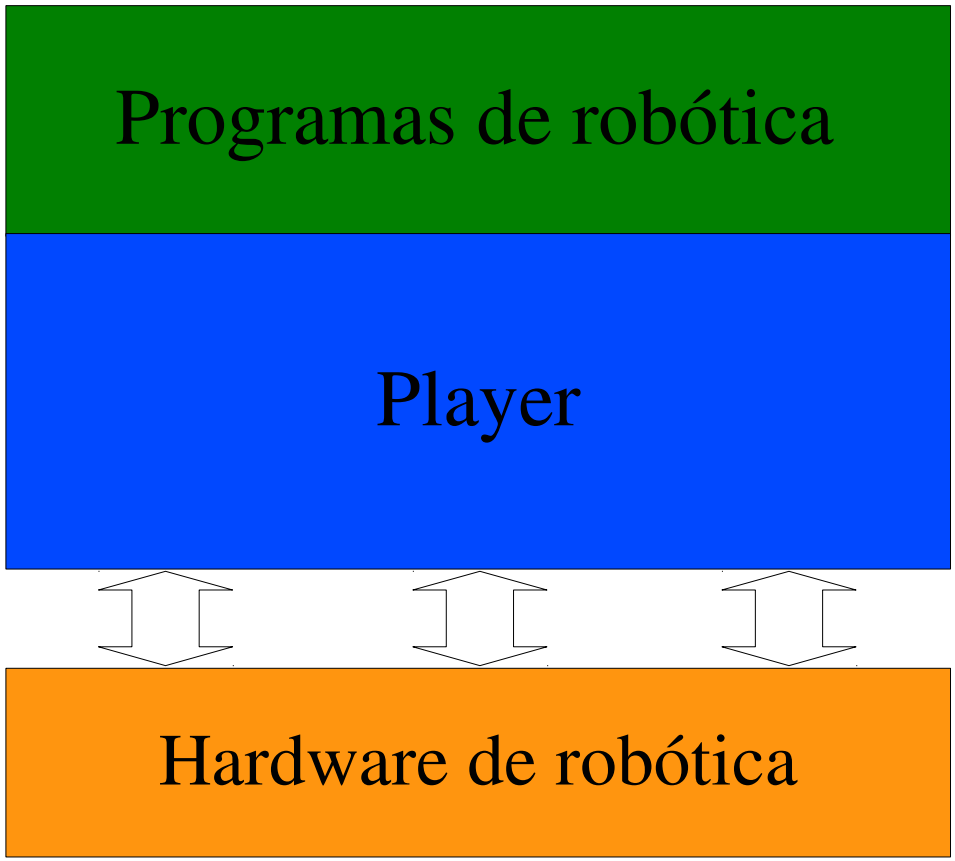
- **Player**
  Es una capa de abstracción de hardware (HAL) para dispositivos de robótica. Trabaja en arquitectura cliente/servidor.



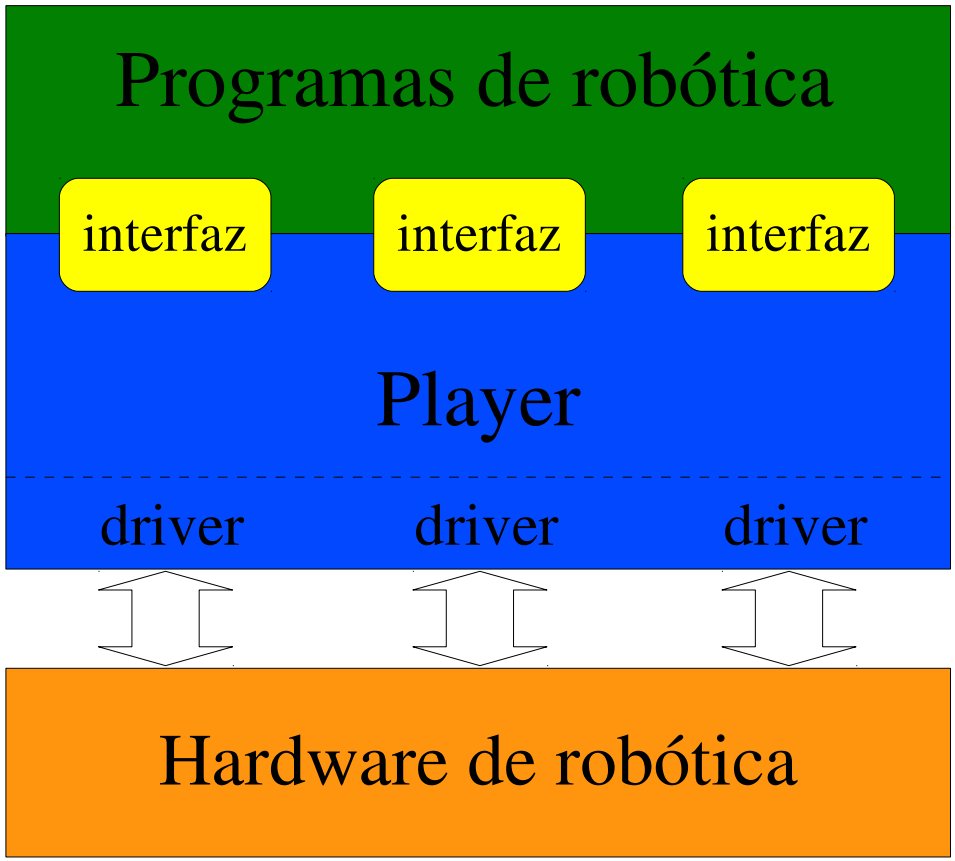
- **Stage**
  Simulador 2.5D. Dispone de robots virtuales y modelos de sensores, como sonares, laser rangefinder, odometría, etc.



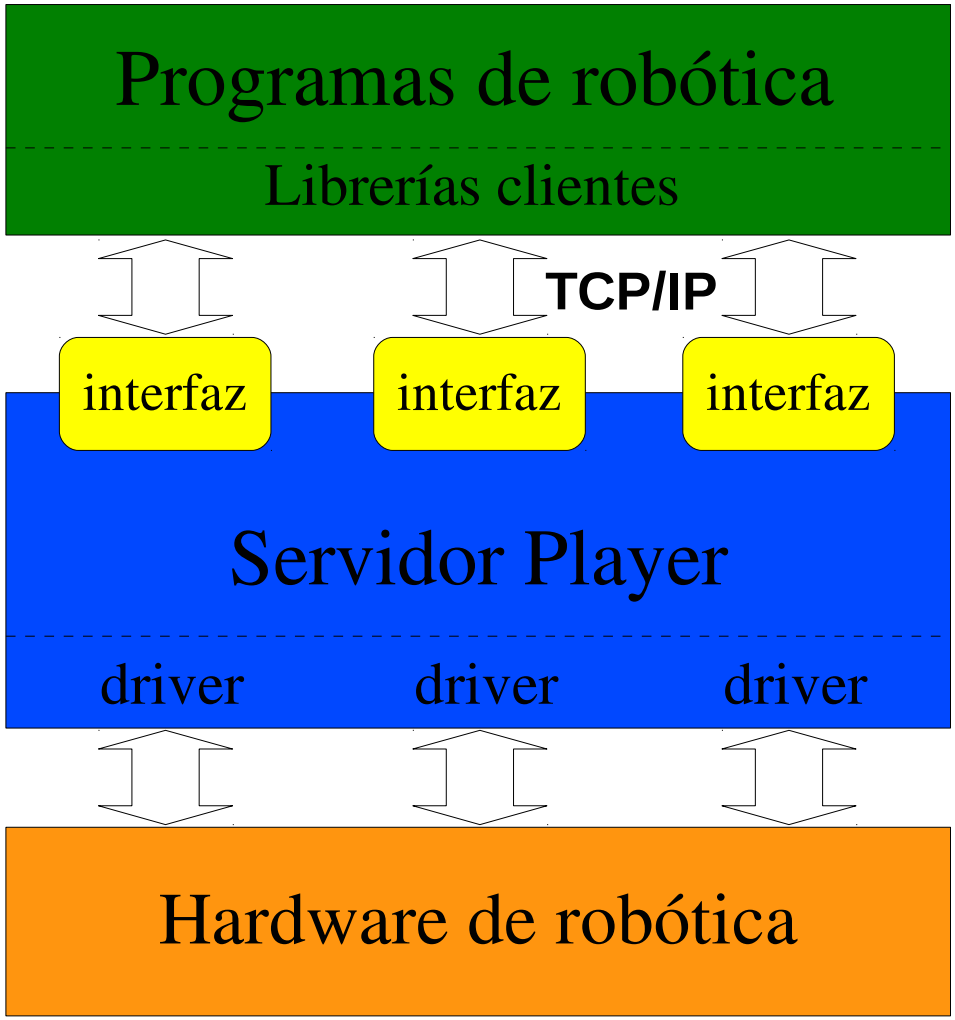
Player/Stage, descripción



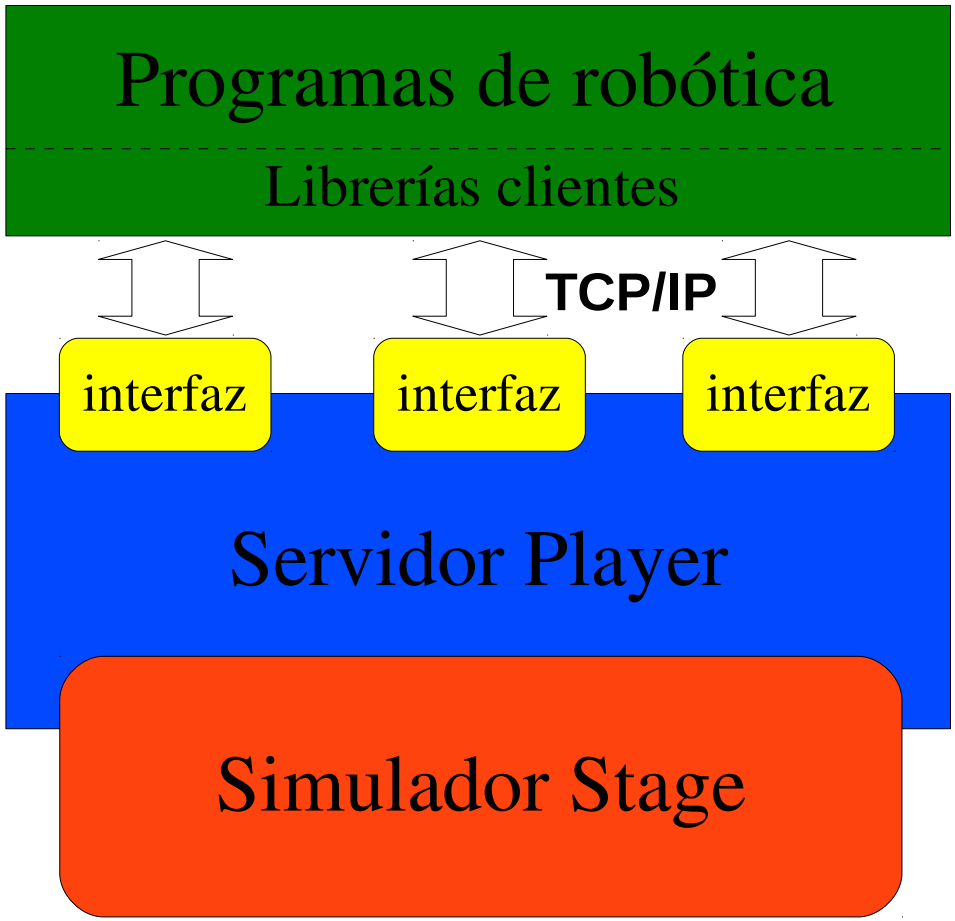
Player/Stage, descripción



Player/Stage, descripción



Player/Stage, descripción



Interfaces y drivers

Intefaces

- **position2d** Planar mobile robot
- **laser** Laser range-finder
- **camera** Camera imagery
- **imu** Inertial measurement unit

- **localize** Planar localization system
- **planner** A planar path-planner
- **map** Access maps
- **log** Log read/write control

Drivers

- Pionner/AmigoBot, Mobilerobots
- Segway
- RoMAA

- PTU-D46 pan-tilt, DirectPerception
- URG laser rangefinder, Hokuyo
- LMS200 laser rangefinder, SICK
- Mtx/Mti IMU, XSens

Servidor Player y librerías clientes

Servidor Player

Ejecución

```
$> player cfgfile.cfg
```

Independientemente si es sobre hardware real o simulado por Stage.

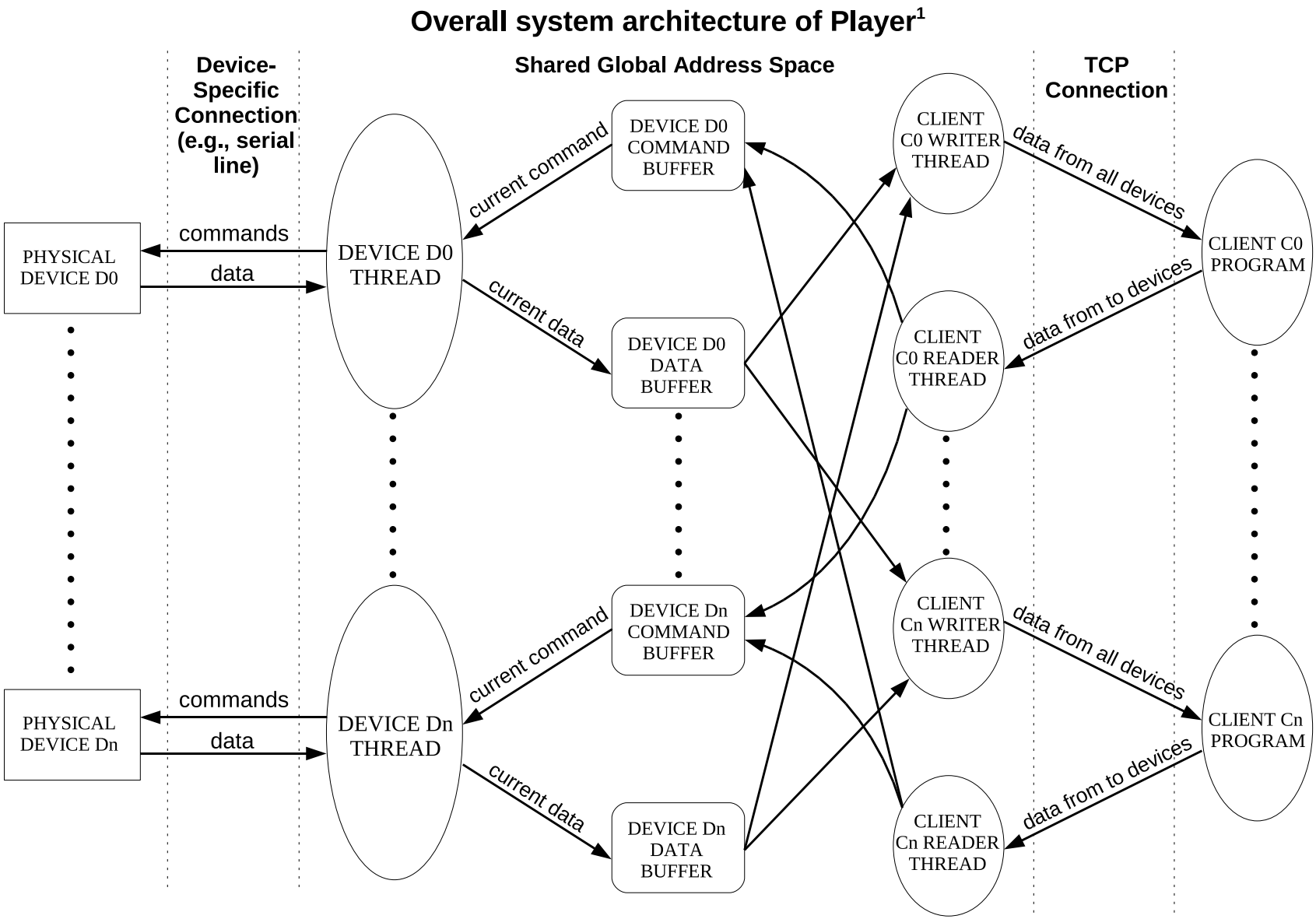
Archivo de configuración `.cfg`

- Indica que driver utilizar o bien el simulador Stage
- Indica que interfaces utilizan los dispositivos

Librerías clientes

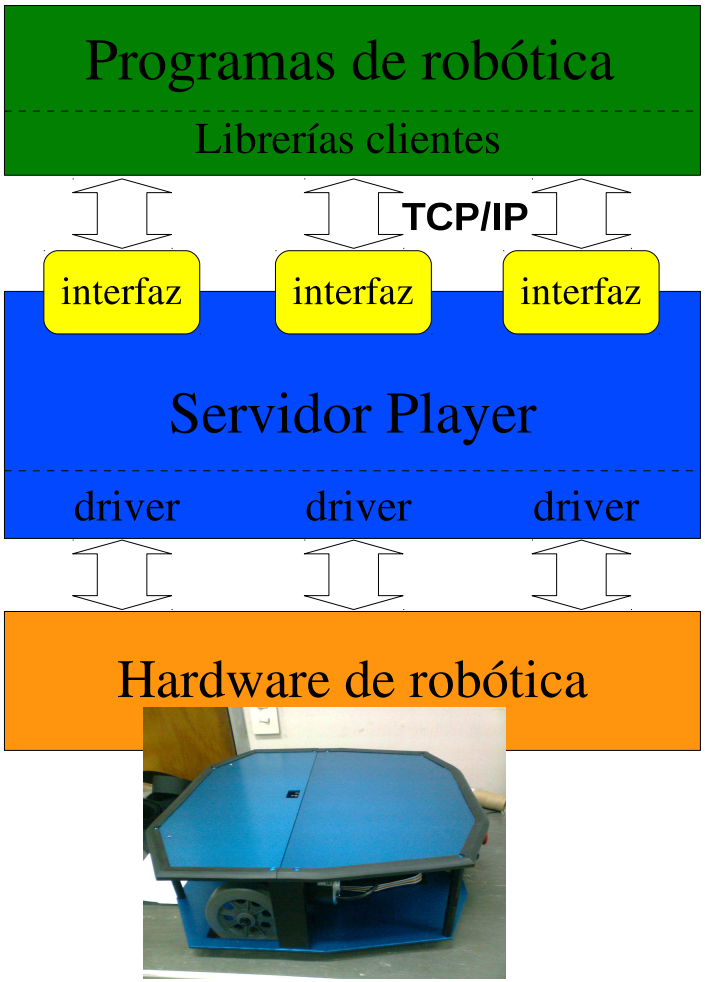
- Del proyecto, C,C++, python
- Contribuciones, como `MATLAB`, `Java`, `GNU/Octave`, etc.

Estructura del servidor Player

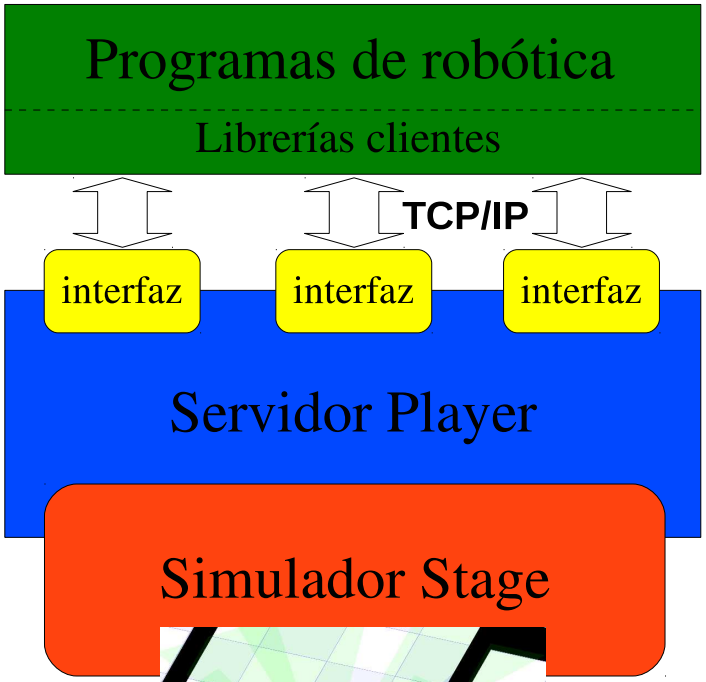


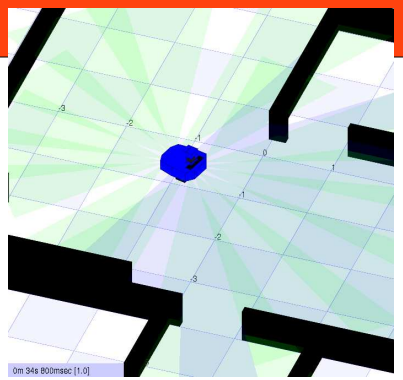
Player con dispositivos reales vs. simulado

```
$>player romaareal.cfg
```



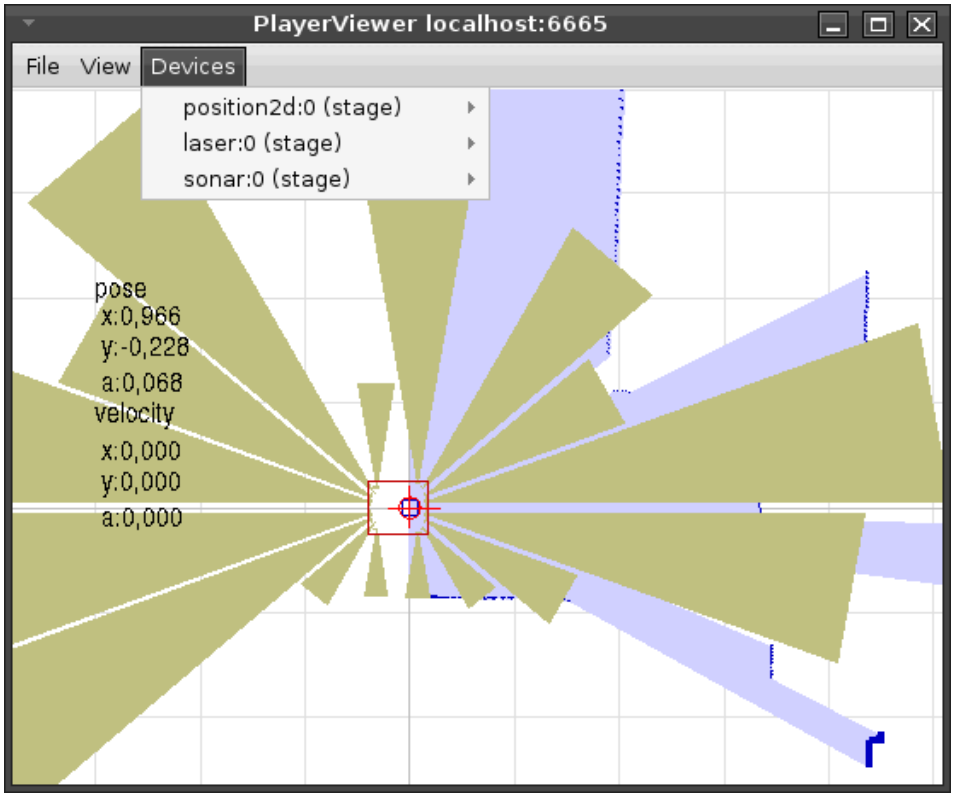

```
$>player romaasim.cfg
```



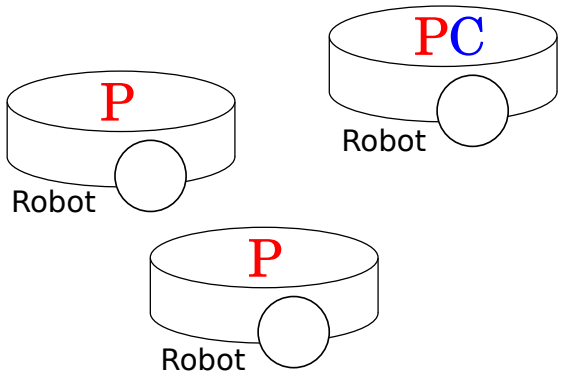


Algunas herramientas de Player

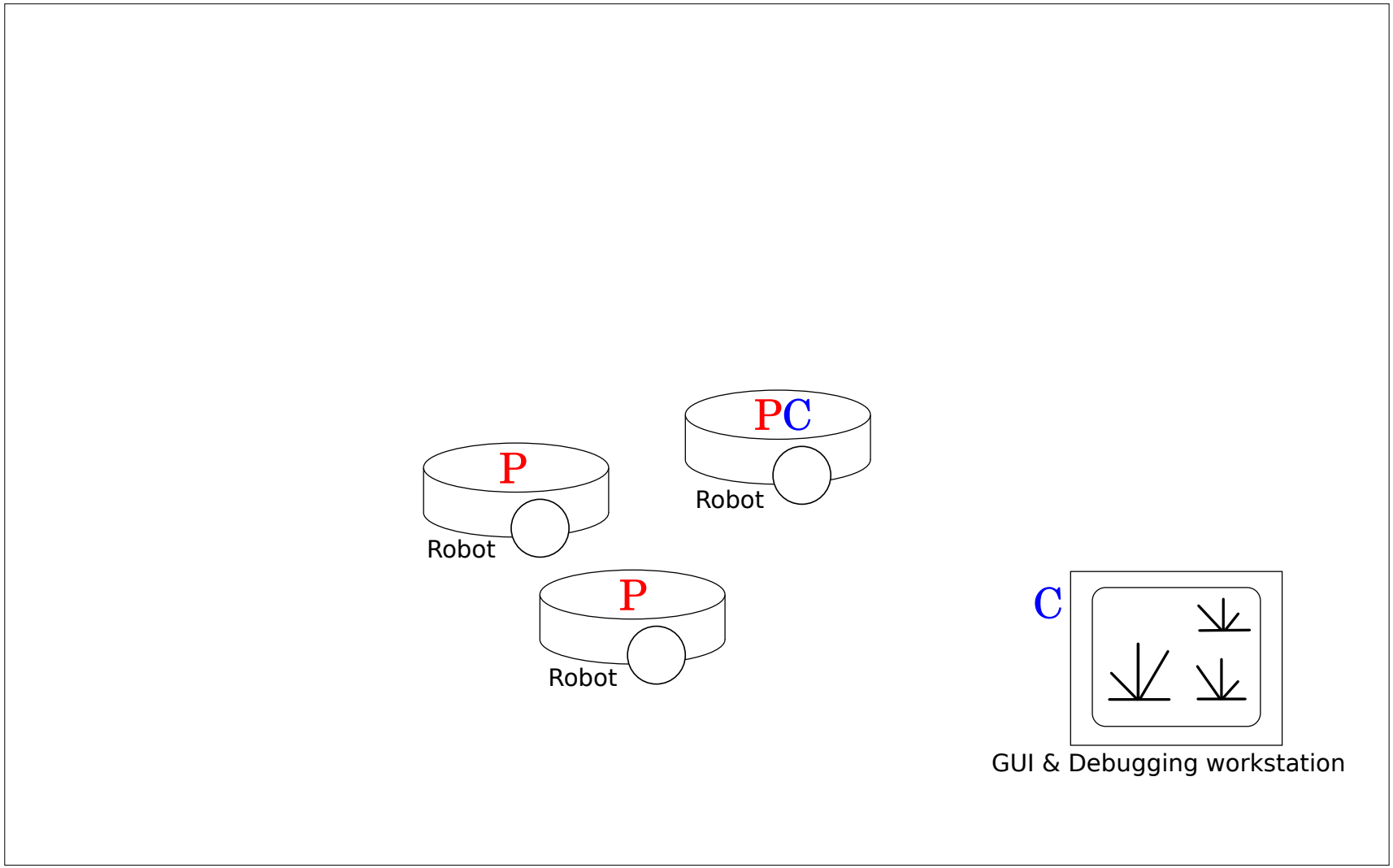
- **playerprint** imprime datos de sensores
- **playerv** muestra datos gráficamente
- **playerjoy** teleoperación
- **playervcr** control de logging
- **playercam** muestra imagen de video



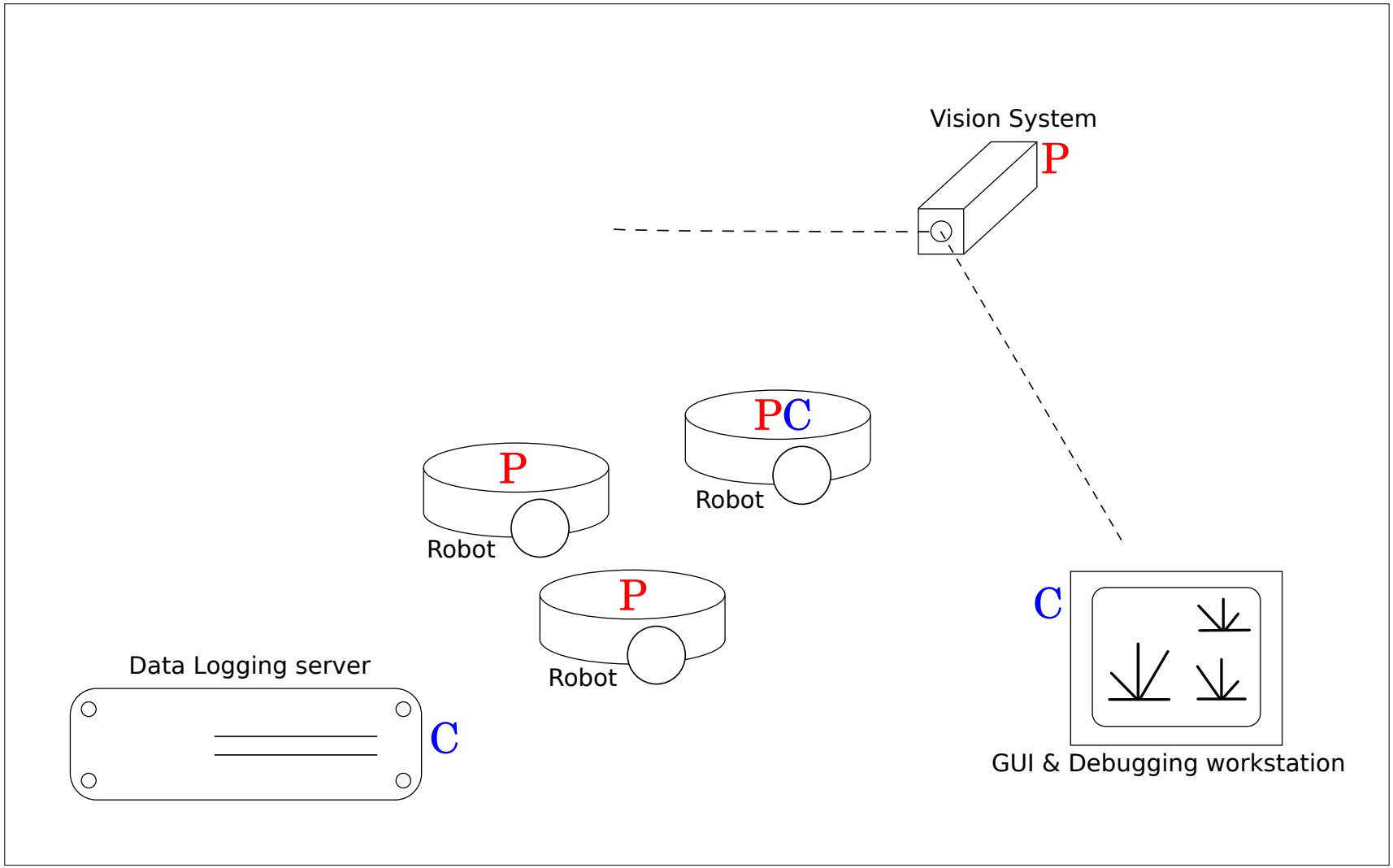
Ejemplo de la flexibilidad de Player



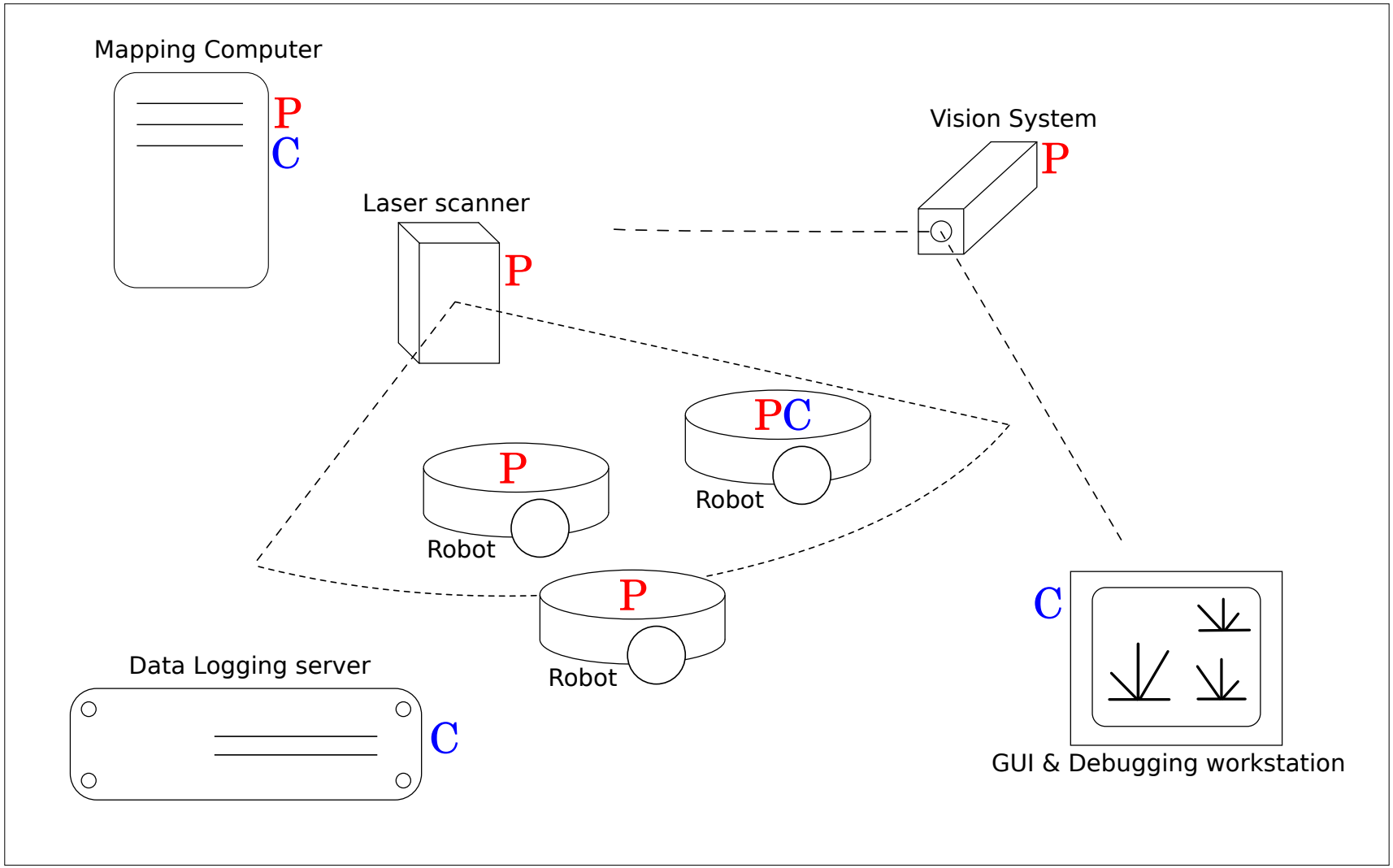
Ejemplo de la flexibilidad de Player



Ejemplo de la flexibilidad de Player



Ejemplo de la flexibilidad de Player



Driver del robot RoMAA para Player

Programación del driver

Se programa en C++ utilizando las librerías provistas por Player (`playercore`).
Se hereda una clase de `ThreadedDriver`.

Generación del driver

La compilación del driver genera una librería dinámica `libromaa.so`

Archivo de configuración

Para cargar el driver y definir la interfaz que utiliza, además de otros parámetros específicos

Archivo de configuración

Sección driver

```
driver
(
  name "driver_name"
  provides [device_address]
  # other parameters...
)
```

Tipos de drivers

- driver normal
- driver plugin

Dirección de dispositivo

`host:robot:interface:index`

Otros parámetros pueden ser `requires` y `plugin`

`$>player romaareal.cfg`

```
driver
(
  name          "romaa"
  plugin        "libromaa"
  provides      [ "position2d:0" ]
  port          "/dev/ttyUSB0"
  baudrate      115200
  motor_pid     [ 1300.0 5000.0 1.0 ]
  vw_pid        [ 0.0 25.0 0.0 ]
  wheel_control 0
  t_odometry    25
  t_loop        20
)
```

```
driver
(
  name       "camera1394"
  provides   [ "camera:0" ]
  framerate  15
  mode       "640x480_mono"
)
```

Archivo de configuración (cont.)

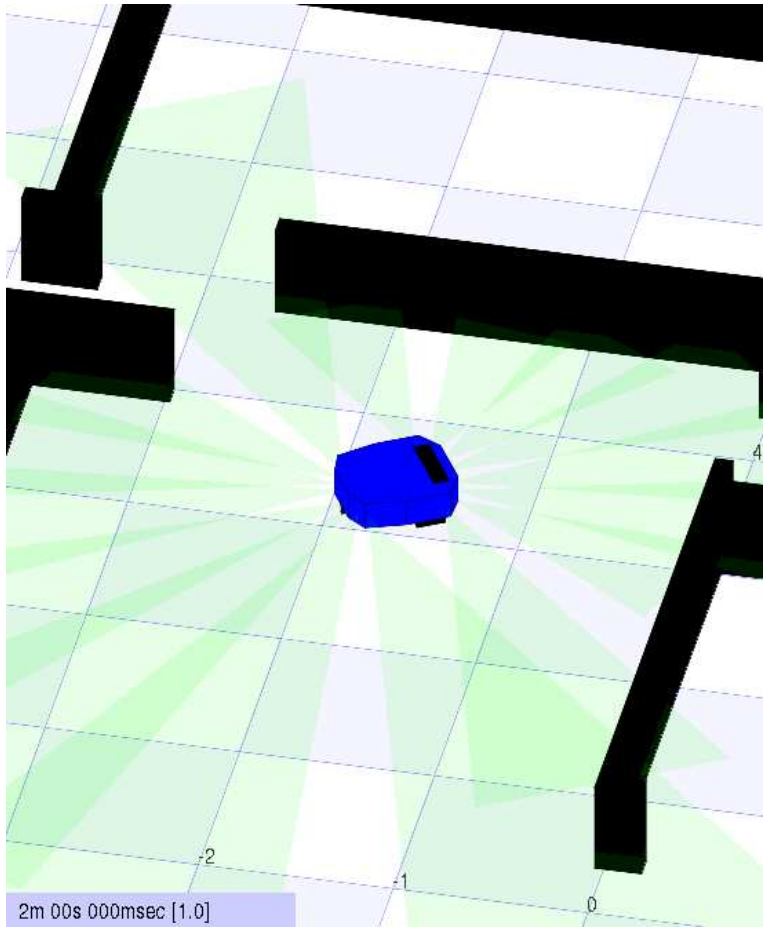
```
$>player romaasim.cfg
```

```
# Desc: Player configuration file for
# controlling RoMAA Stage simulator
# Author: Gonzalo F Perez Paina
# Date: 1 July 2009

# load the Stage plugin simulation driver
driver
(
  name       "stage"
  provides   [ "simulation:0" ]
  plugin     "libstageplugin"
  # load the named file into the simulator
  worldfile "lab.world"
)

# Create a Stage driver and attach
# position2d to the model "romaarobot"
driver
(
  name      "stage"
  provides  [ "position2d:0" "sonar:0" ]
  model     "romaarobot"
)
```

Stage utiliza el archivo `.world`



Archivos de descripción del entorno

Archivo .world

```
# lab.world
# Author: Gonzalo F. Perez Paina

include "map.inc"
include "romaa.inc"

interval_sim 100
interval_real 100

paused 0

# size of the whole simulation
size [15 15]

# configure the GUI window
window
(
  size [ 700.0 700.0 ]
  # 700/46 rounded up a bit
  scale 46
  show_data 1
)
```

```
# load an environment bitmap
map
(
  bitmap "lab.png"
  size [15 15 0.5]
)

# robot model
romaa_sonar
(
  name "romaarobot"
  pose [-2 -2 0 90]
)
```

Ejemplo de programa cliente

```
#include <iostream>
#include <libplayerc++/playerc++.h>

int main(int argc, char *argv[])
{
  // Connect to the local player process on port 6665
  PlayerCc::PlayerClient romaa_robot("localhost", 6665);

  // Create a position2d proxy
  PlayerCc::Position2dProxy romaa_pos2d(&romaa_robot, 0);

  romaa_pos2d.ResetOdometry();
  romaa_pos2d.SetSpeed(1.0, 0.0);

  for( int i = 0; i < 200; i++ )
  {
   romaa_robot.Read();
   std::cout << "i: "  << i << " ---> ";
   std::cout << "px: " << romaa_pos2d.GetXPos() << " - ";
   std::cout << "py: " << romaa_pos2d.GetYPos() << " - ";
   std::cout << "pa: " << romaa_pos2d.GetYaw() << std::endl;
   usleep(50000);
  }
  romaa_pos2d.SetSpeed(0.0, 0.0);
  return 0;
}
```

Ejemplo de programa cliente

```
#include <iostream>
#include <libplayerc++/playerc++.h>

int main(int argc, char* argv[])
{
  // Connect to the local player process on port 6665
  PlayerCc::PlayerClient romaa_robot("localhost", 6665);
  // Create a laser proxy
  PlayerCc::LaserProxy romaa_laser(&romaa_robot, 0);

  romaa_robot.Read();

  std::cout << "Laser data..." << std::endl;
  for( int i = 0; i < romaa_laser.GetCount(); i++ )
  {
    std::cout << romaa_laser.GetRange(i) << " ";
  }
  std::cout << std::endl;

  std::cout << "Laser bearing..." << std::endl;
  for( int i = 0; i < romaa_laser.GetCount(); i++ )
  {
    std::cout << romaa_laser.GetBearing(i) << " ";
  }
  std::cout << std::endl;
  return 0;
}
```

Referencias

D. A. Gaydou, G. F. Pérez Paina, G. M. Steiner, and J. Salomone.
Plataforma móvil de arquitectura abierta.
In *Proceedings of the V Argentine Symposium of Robotics 2008*. Ediuns, 2008, November 2008.

G. F. Pérez Paina and D. A. Gaydou.
Entorno de desarrollo player/stage aplicado a la robótica móvil.
In *Anales del XXII Congreso Argentino de Control Automático*. AADECA 2010, September 2010.

Jennifer Owen.
How to Use Player/Stage, 2dn edition, April 2010.

Toby H. J. Collett, Bruce A. Macdonald, and Brian P. Gerkey.
Player 2.0: Toward a practical robot programming framework.
In *Proc. of the Australasian Conf. on Robotics and Automation (ACRA)*, Sydney, Australia, 2005.

Nick Wong, Jui-Chun Peng Hsu, Toby H.J. Collet, and Bruce A. MacDonald.
Improving the 2.5d stage robotic simulator.
2008.

Richard Vaughan.
Massively multi-robot simulation in stage.
Swarm Intelligence, 2(2-4):189–208, 2008.