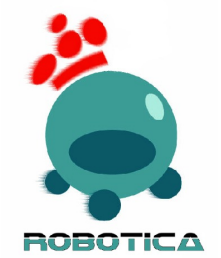
Programación de aplicaciones en robótica, visión artificial y domótica

José María Cañas Plaza

http://gsyc.es/jmplaza

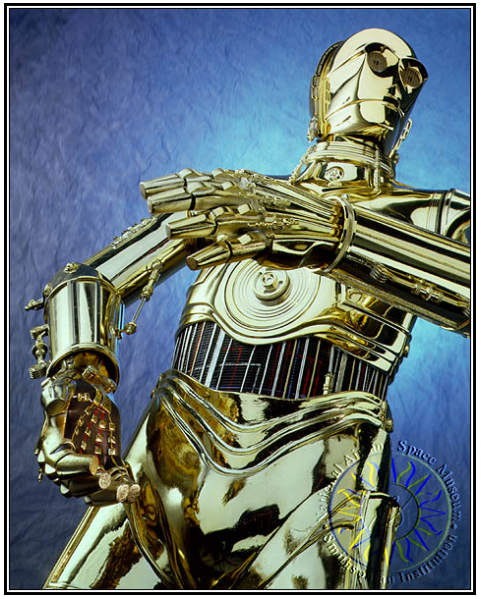


Politécnico Colombiano JIC
Medellín, 21 noviembre 2011

Contenidos

1. Introducción

Robótica ficción vs Robótica real



Aplicaciones reales

- *Dull, Dirty, Dangerous*
- Industria automovilística: brazos para pintar, soldar, mover piezas...
- Gestión de almacenes: KIVA, Cofares
- Espacio: Spirit, Opportunity
- Entretenimiento: Aibo, NXT
- Usos militares, desactivación explosivos: PackBot
- Medicina: DaVinci
- Hogar: Roomba
- Prestige, limpieza centrales nucleares
- Envasado de alimentos

Investigación en robótica

- Generar comportamiento autónomo (inteligencia) en robots móviles
- A más autonomía más aplicaciones
- Multidisciplinar: electrónica, informática, psicología, etología...
- Un robot en cada casa, paralelismo con PC
- Los deseos (y las películas) van por delante de la realidad, pero hay progreso real
- Humanoides
- RoboCup (liga estandard), UrbanChallenge, etc.
- Mapas, localización y navegación
- Interacción con personas
- Prototipos, robustez

¿Qué es un robot?

Sistema informático con:

- Sensores
- Actuadores
- Computador

Hay que programarlo para que consiga sus objetivos y sea sensible a la situación.

La inteligencia reside en su software

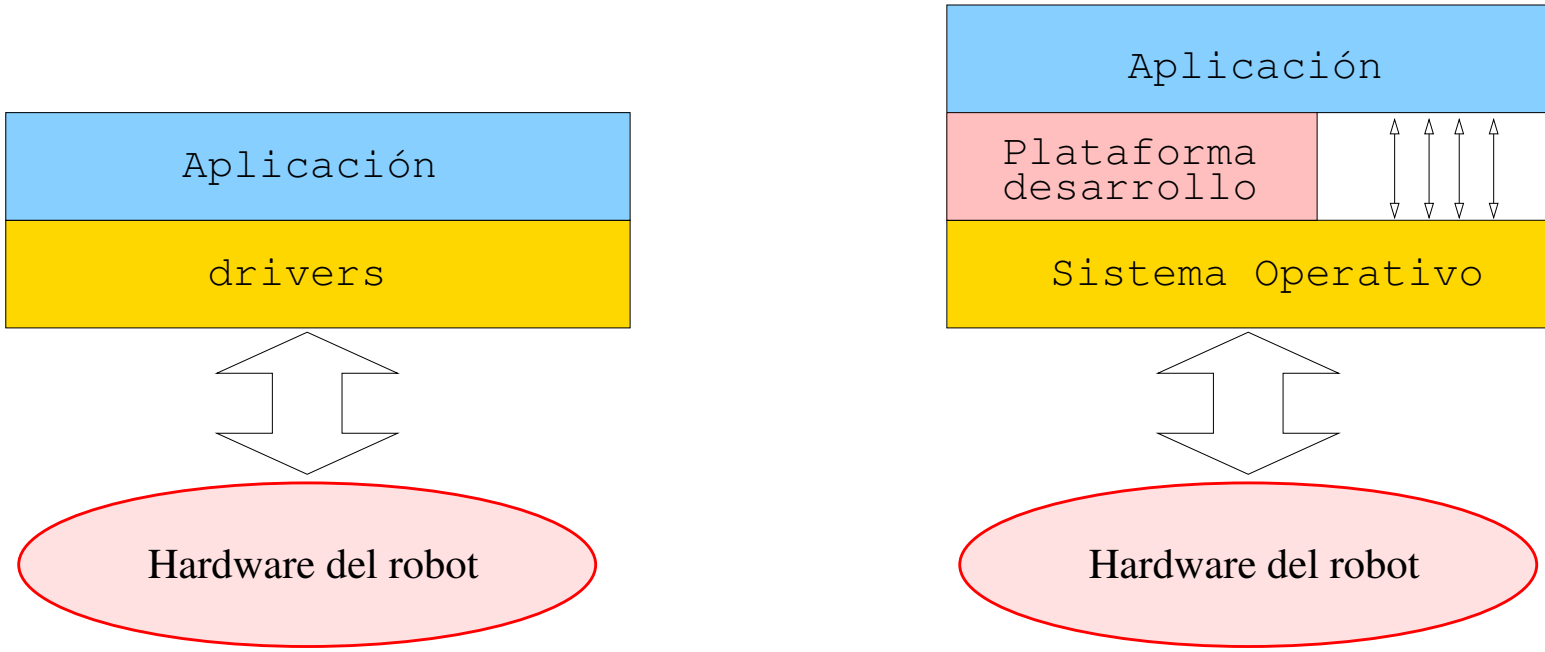
2. Software para robots

- Determina el comportamiento del robot
- No hay una manera universalmente aceptada de programarlos
- Lenguajes: ensamblador, C, C++
- Heterogeneidad
  - Dispositivos hardware
  - Encapsular funcionalidad
- Empezar de cero con cada robot
- Requisitos específicos
- Sistemas operativos y plataformas
- Simuladores

Requisitos específicos

- Vivacidad, agilidad (tiempo real)
- Multitarea (concurrencia, múltiples fuentes de actividad)
- Distribuido, comunicaciones
- Interfaz gráfica, depuración
- Expandible
- Hardware heterogéneo

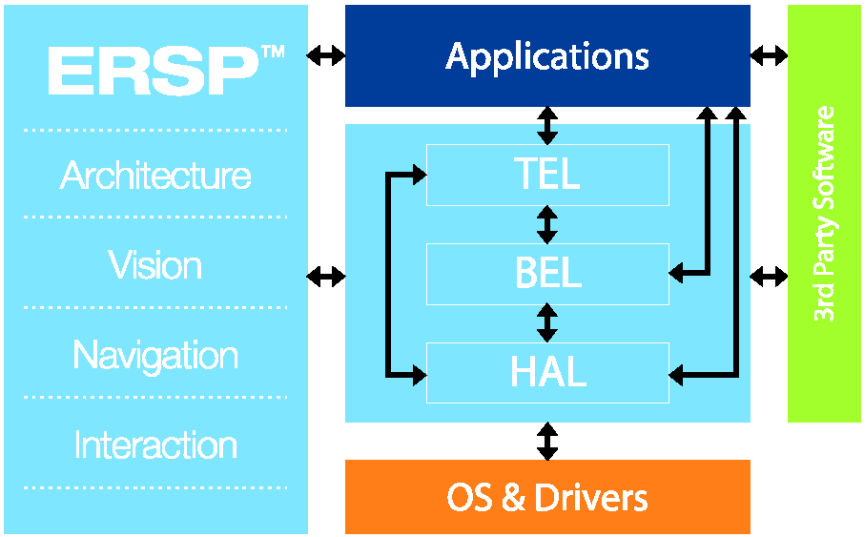
Sistemas operativos y plataformas



- Procesadores empotrados (robots pequeños) o PC (medianos-grandes).
- Sistemas operativos: dedicados o generalistas
- *Middleware* para simplificar la creación de aplicaciones robóticas

¿Qué proporciona una plataforma sw para robots?

- Abstracción del hardware (HAL)
- Arquitectura software
- Funcionalidades de uso común
- Arquitectura cognitiva

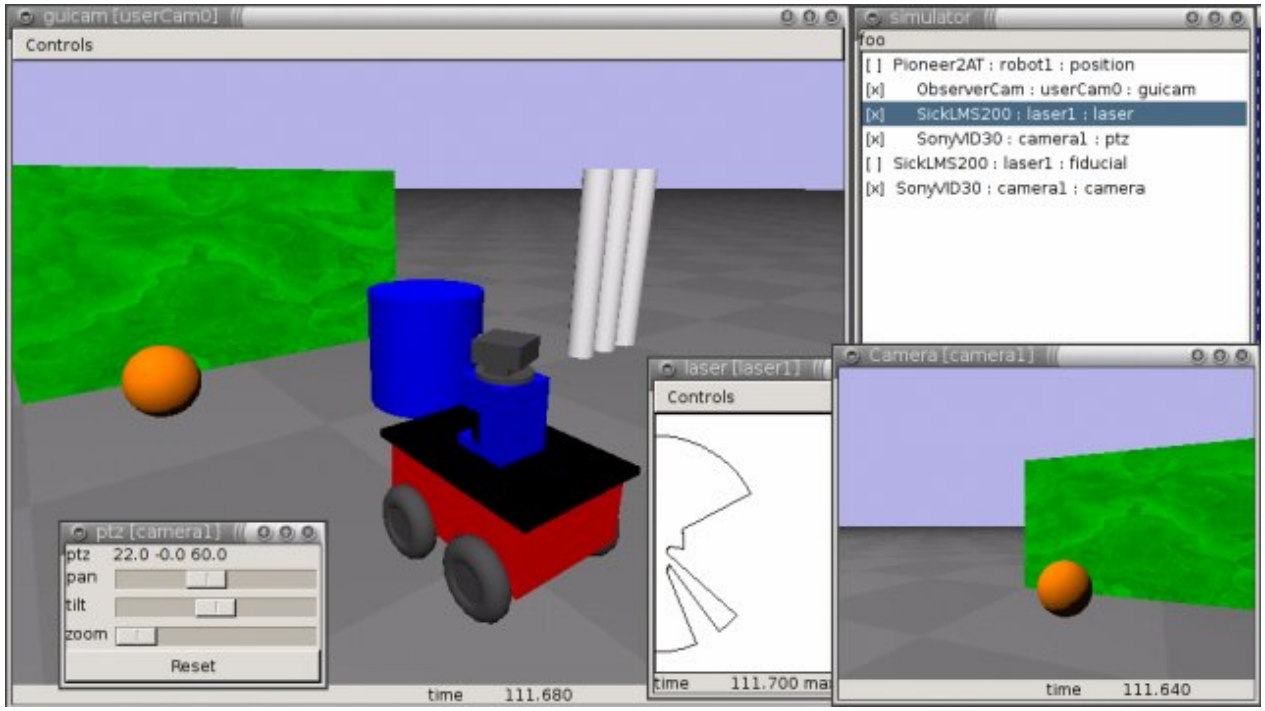


- Comerciales, investigación, software libre
- Ingeniería software: orientación a objetos, distribución
- ROS, Orca, Carmen, OROCOS, ERSP, Player/Stage, Miro, etc.

Simuladores

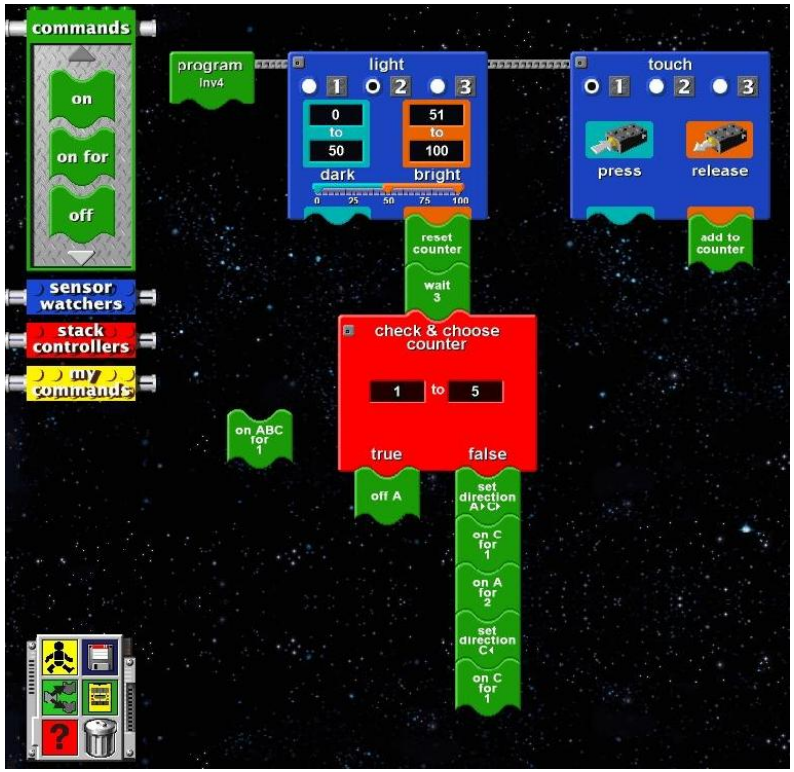
- Madurar algoritmos
- Comodidad trabajar sin robot
- Las caídas no duelen
- Mundo, sensores y actuadores
- OpenGl (OGRE) para imágenes
- Motor físico: ODE (*Open Dynamics Engine*)

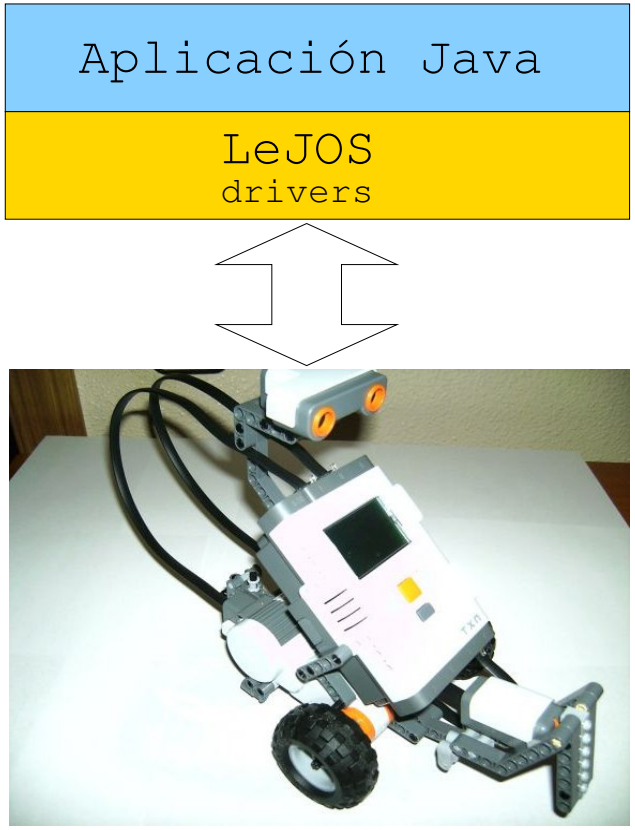
- Gazebo, Stage, Webots, Microsoft Robotics Studio


guicam [userCam0]
Controls
simulator
foo
[ ] Pioneer2AT : robot1 : position
[x] ObserverCam : userCam0 : guicam
[x] SickLMS200 : laser1 : laser
[x] SonyVID30 : camera1 : ptz
[ ] SickLMS200 : laser1 : fiducial
[x] SonyVID30 : camera1 : camera
ptz [camera1]
ptz 22.0 -0.0 60.0
pan
tilt
zoom
Reset
laser [laser1]
Controls
Camera [camera1]
time 111.680
time 111.640


Stage: worlds/fasr.world
File View Run Help

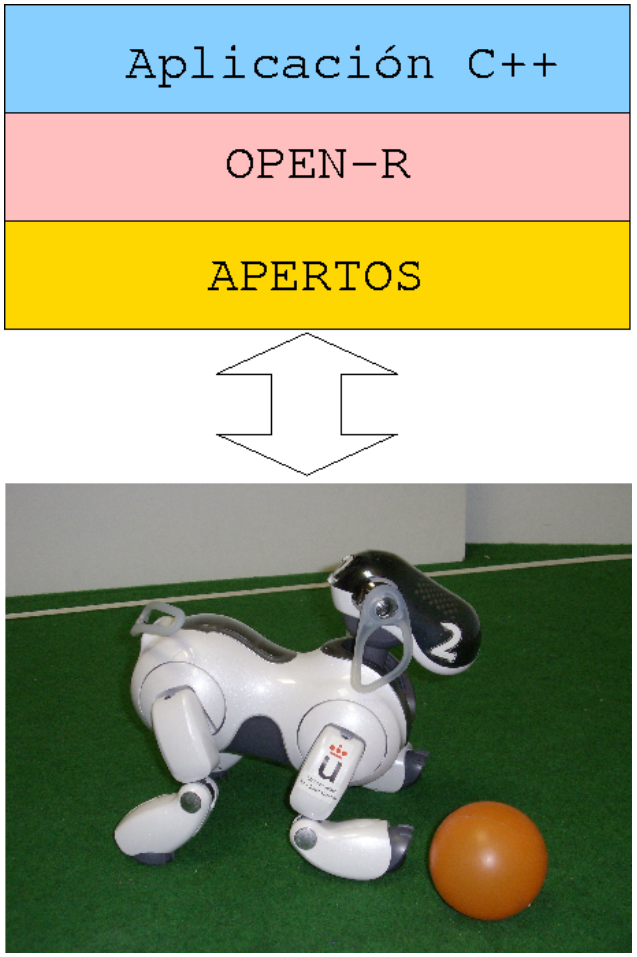
Programación del robot LEGO NXT



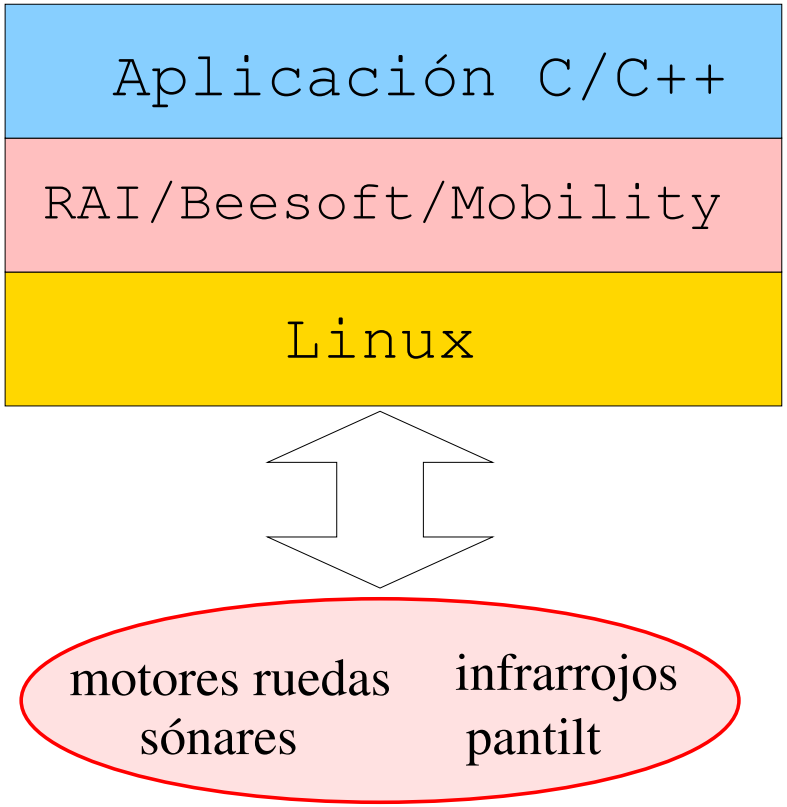


Programación del robot Aibo

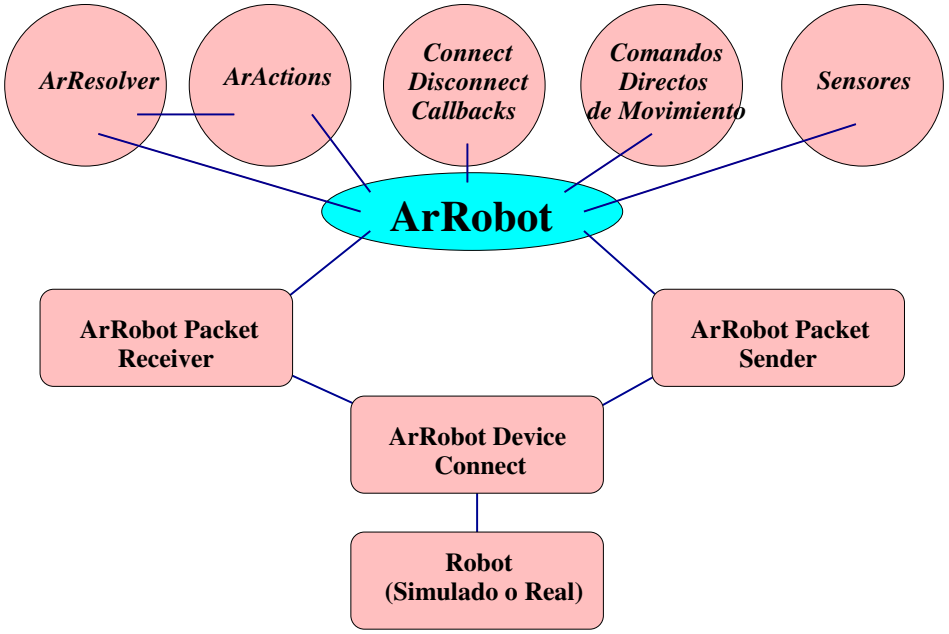
- MIPS 450Mhz
- Objetos monohilo
- Se comunican vía mensajes
- Objetos básicos, OPEN-R
  - OVirtualRobotComm: Effector, Sensor, OFbkImageSensor
  - OVirtualAudioComm: Mic, Speaker
  - ANT Aibo Network Tool

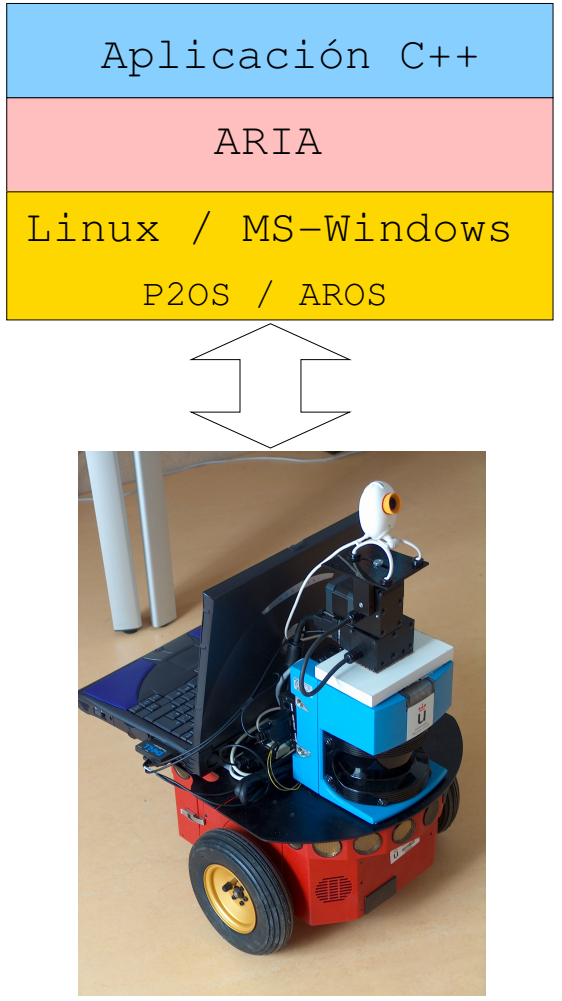


Programación del robot B21



Programación del robot Pioneer

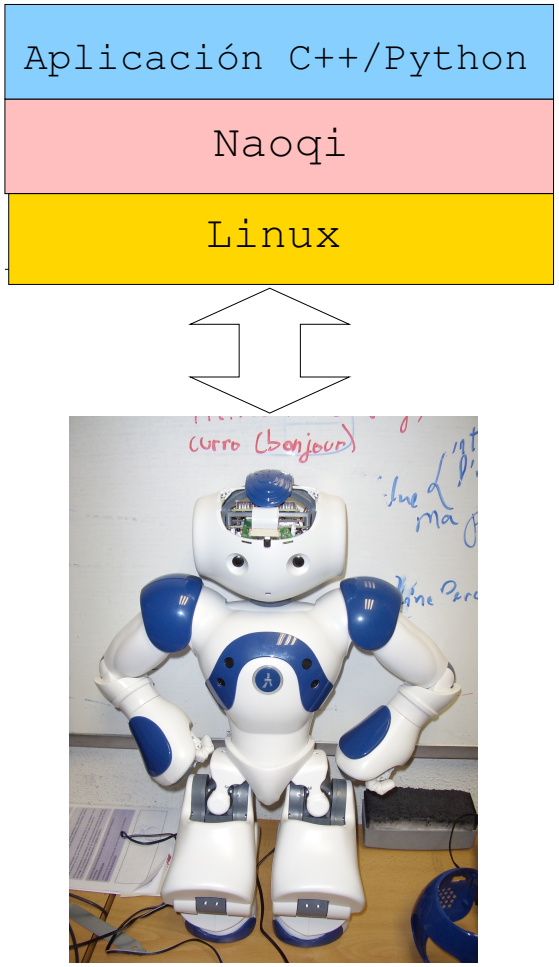




- Acceso a los sensores y actuadores
- Librerías, objetos
- ActivMedia: Saphira, ARIA (C++)

Programación del robot Nao

- AMD Geode 500Mhz
- Objetos distribuidos, invocación de métodos remotos
  - ALMotion
  - ALVision
  - ...
- Naoqi: Módulos, *brokers*



3. Plataforma software Jderobot

- Nace como implementación de arquitectura cognitiva, tesis doctoral
- Evolucionó a una plataforma de programación para aplicaciones robóticas, domóticas y de visión computacional (sensores, actuadores, inteligencia)
- Orientada a componentes distribuidos y multilenguaje
- La mantiene un grupo de desarrolladores, es abierta
- http://jderobot.org, manuales, descarga, ejemplos
- Software libre (licencia GPLv3), paquete debian

Arquitectura cognitiva JDE

- Comportamiento = percepción y control
- Fragmentación en unidades asíncronas concurrentes (esquemas)
  - de percepción elaboran estímulos
  - de actuación toman decisiones
- La colección de esquemas se organiza en *jerarquía* dinámica (JDE)
- Sigue el paradigma basado en comportamientos

Arquitectura cognitiva: Esquemas

Un esquema es un flujo de ejecución independiente, con un objetivo

- Funcionamiento continuo
- Se puede activar y desactivar a voluntad
- Modulable a través de parámetros
- Su funcionalidad se usa despertándolo y modulándolo
- Perceptivos: producen estímulos (piezas de información) y los mantienen actualizados. Lecturas sensoriales, transformaciones más elaboradas
- Actuación: toman decisiones para conseguir o mantener un objetivo, comandos a los actuadores o la activación y modulación de otros

Modelo de programación: Componente

- Cada esquema se materializa en un componente software
- El funcionamiento continuo como ejecución iterativa, iteraciones periódicas a cierto ritmo para percibir o para actuar
- Ofrece funcionalidad y usa la de otros a través de interfaces explícitos
- Aplicación = conjunto de componentes concurrentes que interoperan
- Multimáquina y multilenguaje: C, C++, python, Java...
- Usa el middleware de comunicaciones ICE (zeroC)

Ejecución iterativa

- Iteración de control o de percepción
- component_cycle
- `gettimeofday`
- `usleep`
- Espera condicional
- Patrón de diseño *Algoritmo iterativo*

Interfaces tipados

- Comunicación entre componentes, conexión con otros
- Fijan la estructura de los datos que el componente ofrece o necesita
- Cada componente puede necesitar y ofrecer uno o varios interfaces
- Se definen con `slice`
- Generación de código para múltiples lenguajes (C++, Java, Python...)
- Mecanismos de transmisión de datos eficientes
  - transmisión síncrona o asíncrona
  - publicación/subscripción

Visualización

- Modular y opcional: cada componente tiene (o no) su propio GUI
- Cada componente incluye el código de su interfaz gráfica
- Mostrar e interacción con humano
- Depuración, útil en desarrollo
- GTK, XForms, OpenGL
- Patrón de diseño *Modelo Vista Controlador*
- Se materializa en una hebra adicional

Acceso a los dispositivos hardware

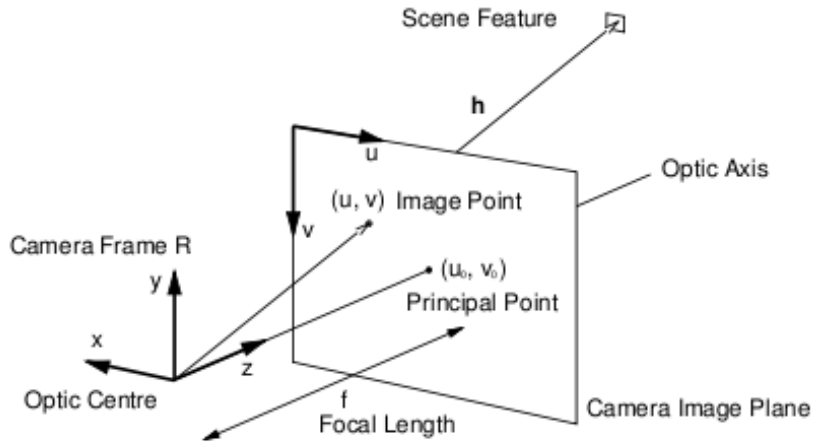
- Se da a través de componentes drivers
- Acceden localmente al sensor/actuador y ofrecen acceso (local o remoto) a otros a través de interfaces ICE
- No suelen tener GUI
- El mismo interfaz de datos lo puede proporcionar varios *drivers*
  - Mismo programa funciona sobre robot real o simulador
  - Independencia de la fuente de imágenes

Ficheros de configuración

- Cada componente puede tener el suyo
- Ficheros de configuración ICE
- Especifican cuál es la fuente de cada interfaz (local o remota)
- Parámetros específicos

Bibliotecas

- Colorspaces: espacios de color para imágenes
- Fuzzylib: control borroso
- Progeo: relacionar las imágenes con 3D, cámaras calibradas
- Visionlib: geometría espacial, analisis de imágenes



Jderobot, proyecto de software libre

- Más de 60000 líneas de código
- Comunidad de usuarios y desarrolladores
- ¿Dónde conseguirla? ¿Cómo preguntar dudas?
- Página web: http://jderobot.org
- Listas de correo: jde-users@gsyc.es y jde-developers@gsyc.es
- Svn, trac, blog, mediawiki
- Paquete debian: `apt-get install jderobot`

¿Por qué software libre en robótica?

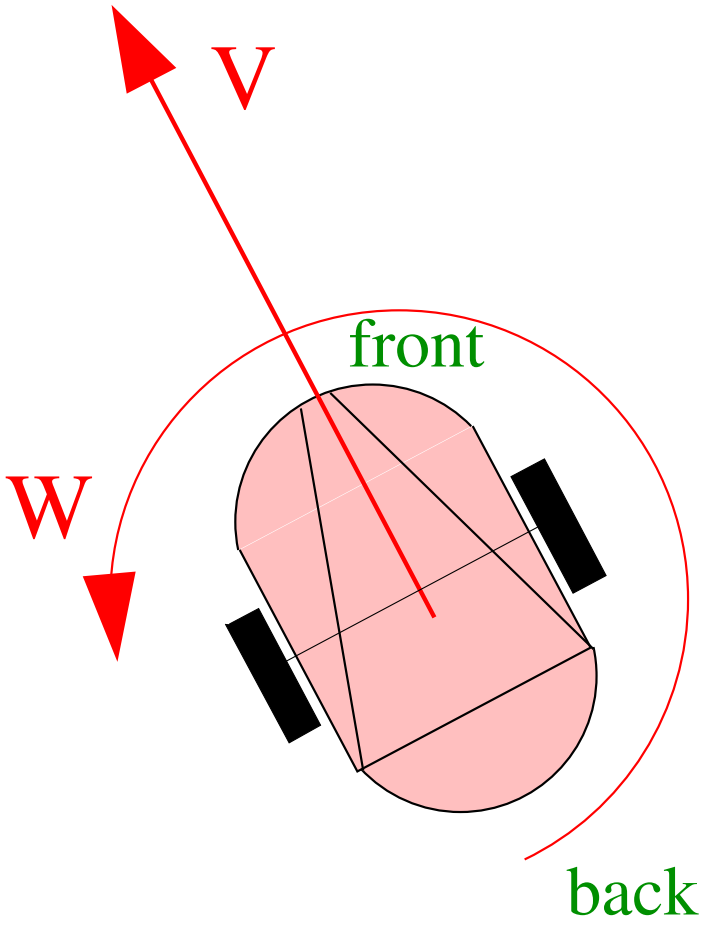
- Independencia de fabricantes
- Varios modelos de robots
- Libertad para modificar código fuente
- Aumentar la calidad del software
- "Devolver el favor": Player, Stage, Gazebo, Opencv, GTK,...

Dispositivos soportados

- Soporte de muchos sensores y actuadores
- Player
- Robots: Pioneer, humanoide Nao
- Laser: Hokuyo, Sick
- Cuellos mecánicos: Directed Perception, Sony Evi
- Simuladores: Stage, Gazebo
- Cámaras: usb, firewire, digitalizadoras, videos
- Wiimote
- Sensor kinect

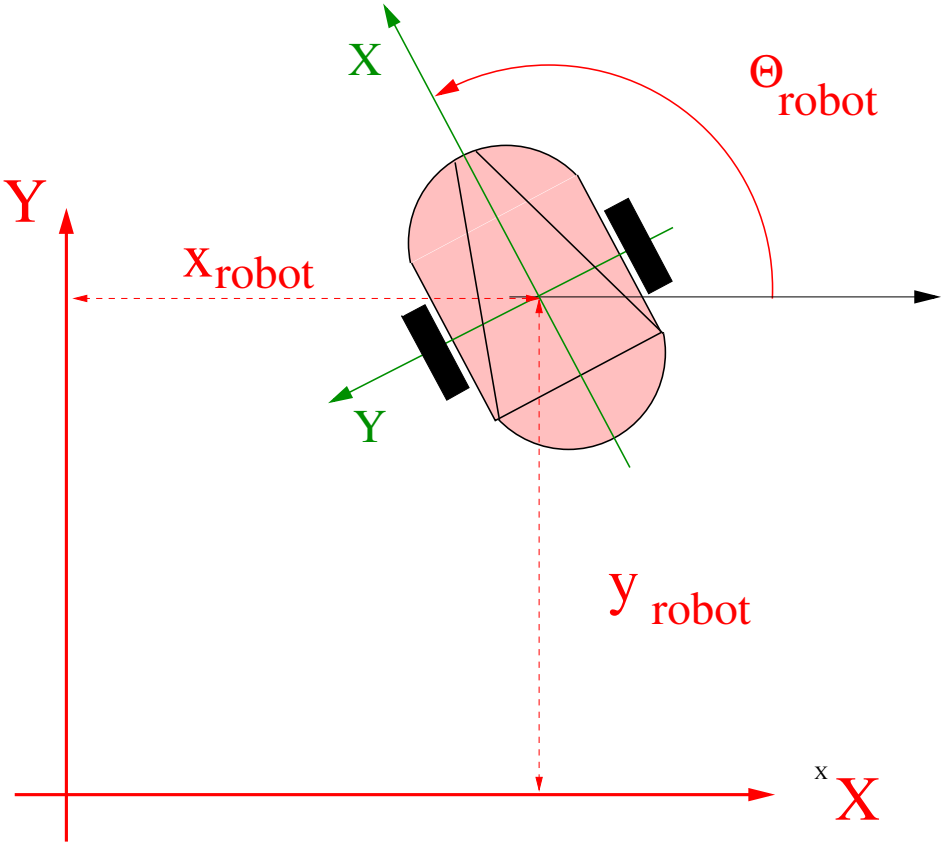
Interfaz para motores

- Datos: v,w
- Playerserver, gazeboserver



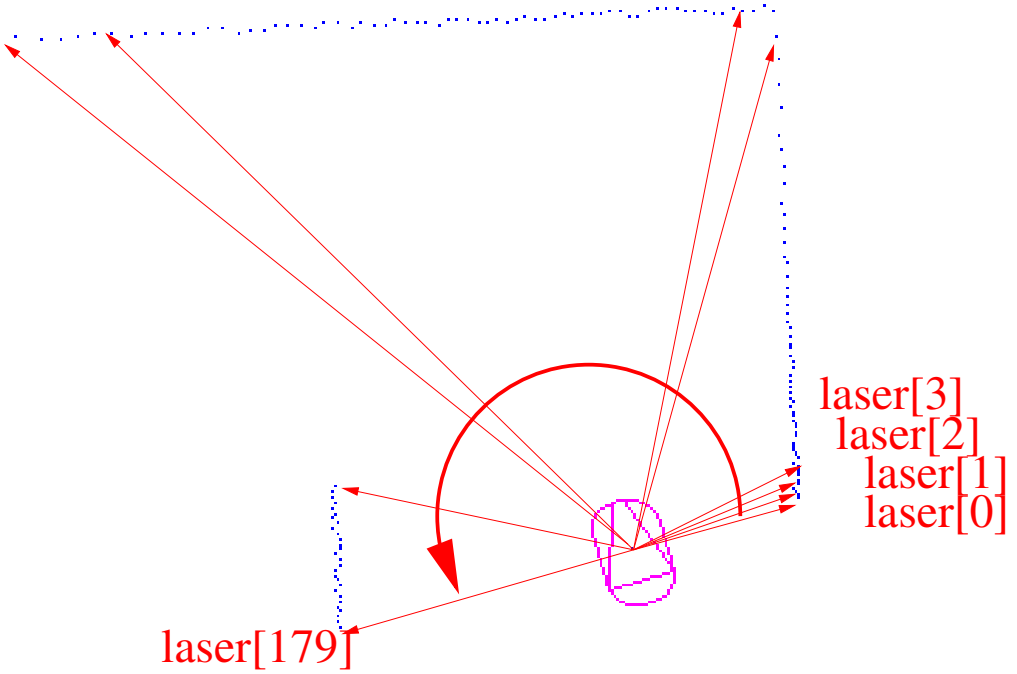
Interfaz de posición

- Datos: x, y, theta (mm,radianes)
- Playerserver, gazeboserver



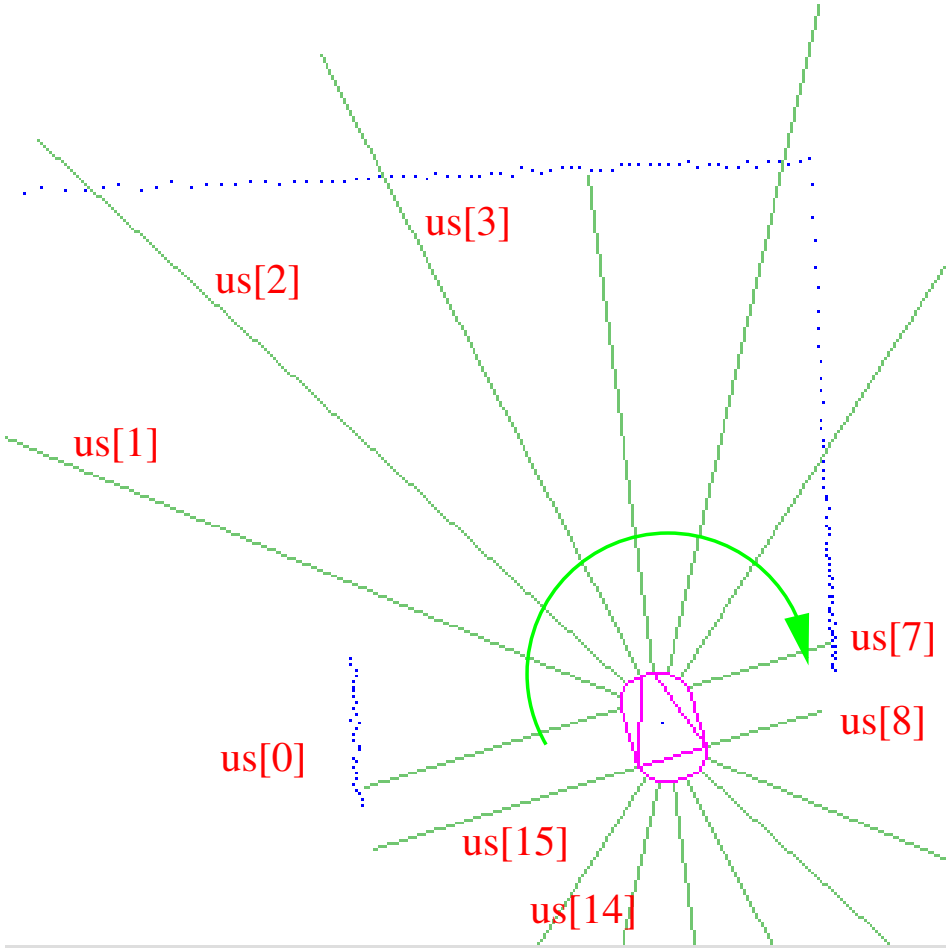
Interfaz para laser

- Datos: laser
- Playerserver, gazeboserver



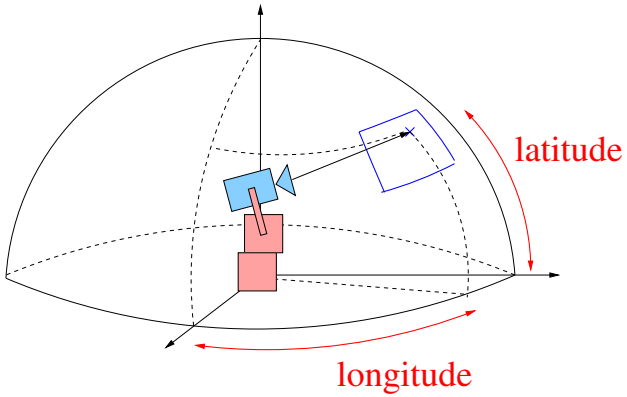
Interfaz para ultrasonidos

- Datos: us
- Playerserver, gazeboserver

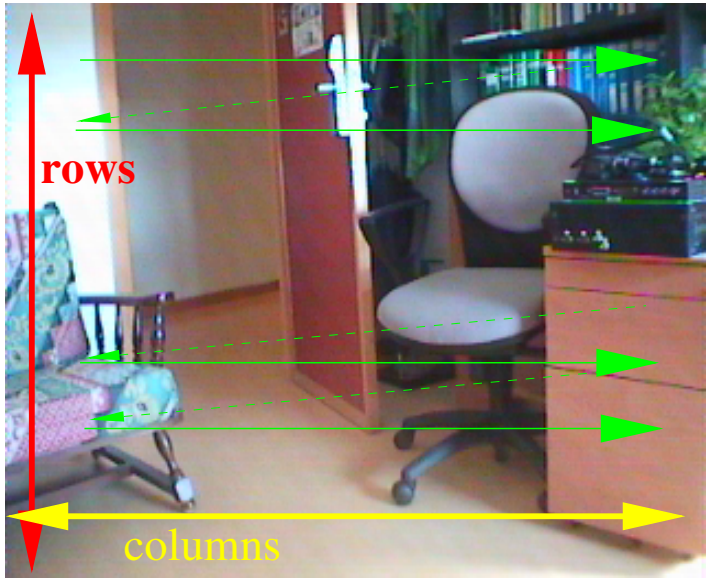


Interfaces para el cuello mecánico

- pan, tilt
- latitude, longitude

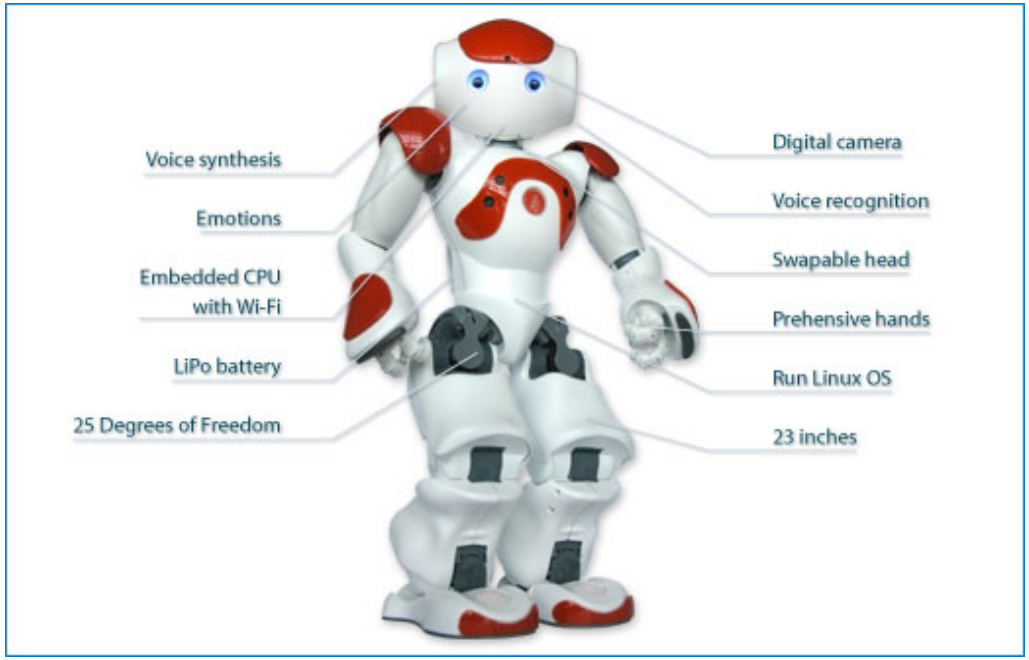


Interfaces para imágenes



- Cámaras webcam USB, firewire, en red, digitalizadoras, ficheros
- Datos: ImageData
- Cameraserver, gazeboserver

Interfaces para humanoide Nao

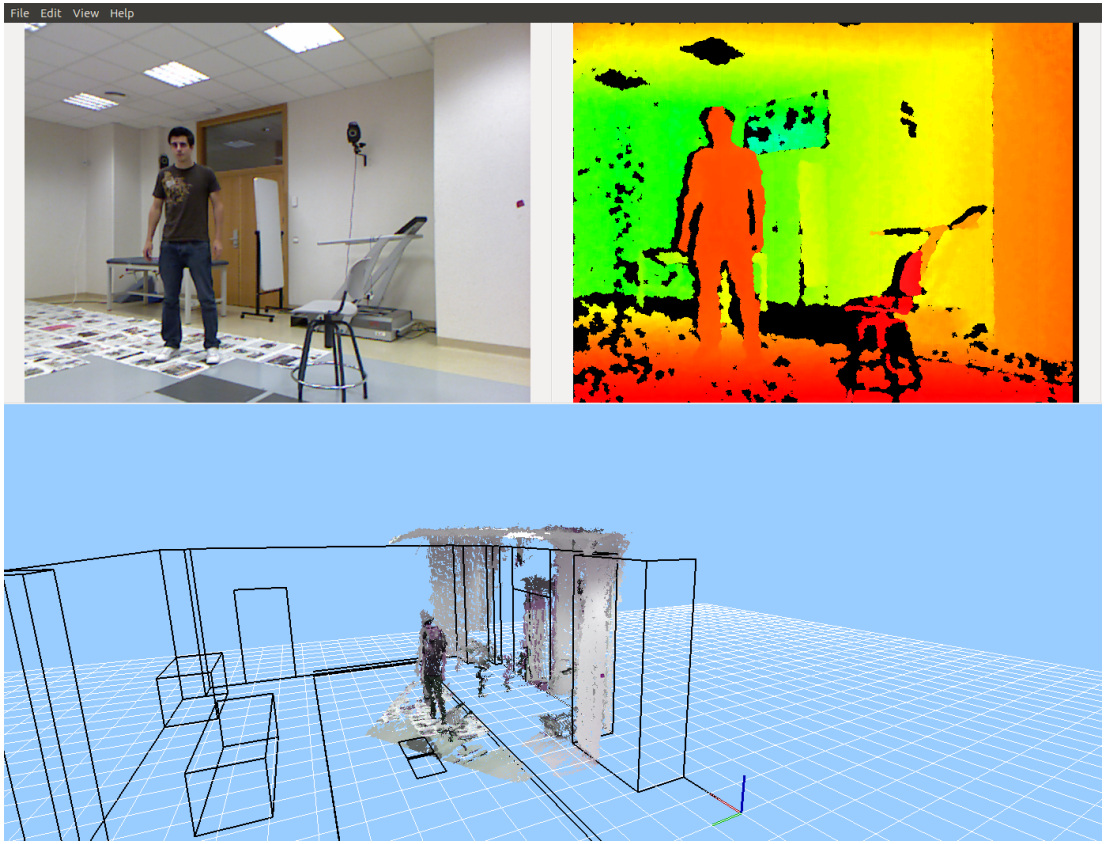


- Articulaciones, cámaras, cuello mecánico, odometría
- NaoDriver

Interfaces para wiimote

- Datos: buttons, accel, ir, nunchuk
- Driver wiimote

Interfaces para kinect



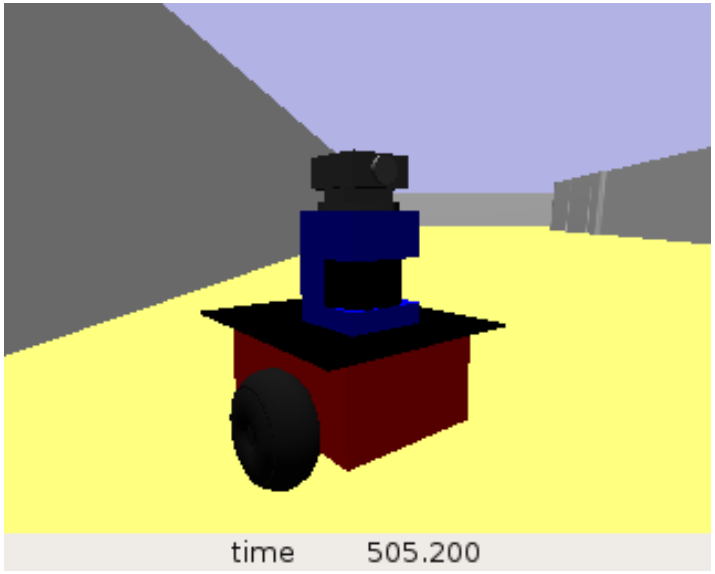
- Datos: imágenes color, imágenes profundidad
- Driver FreeNect

4. Aplicaciones robóticas

- Conjunto de componentes: aplicaciones, drivers, herramientas
- Materializan las capacidades perceptivas y comportamientos
- Docencia en robótica
- Investigación en robótica

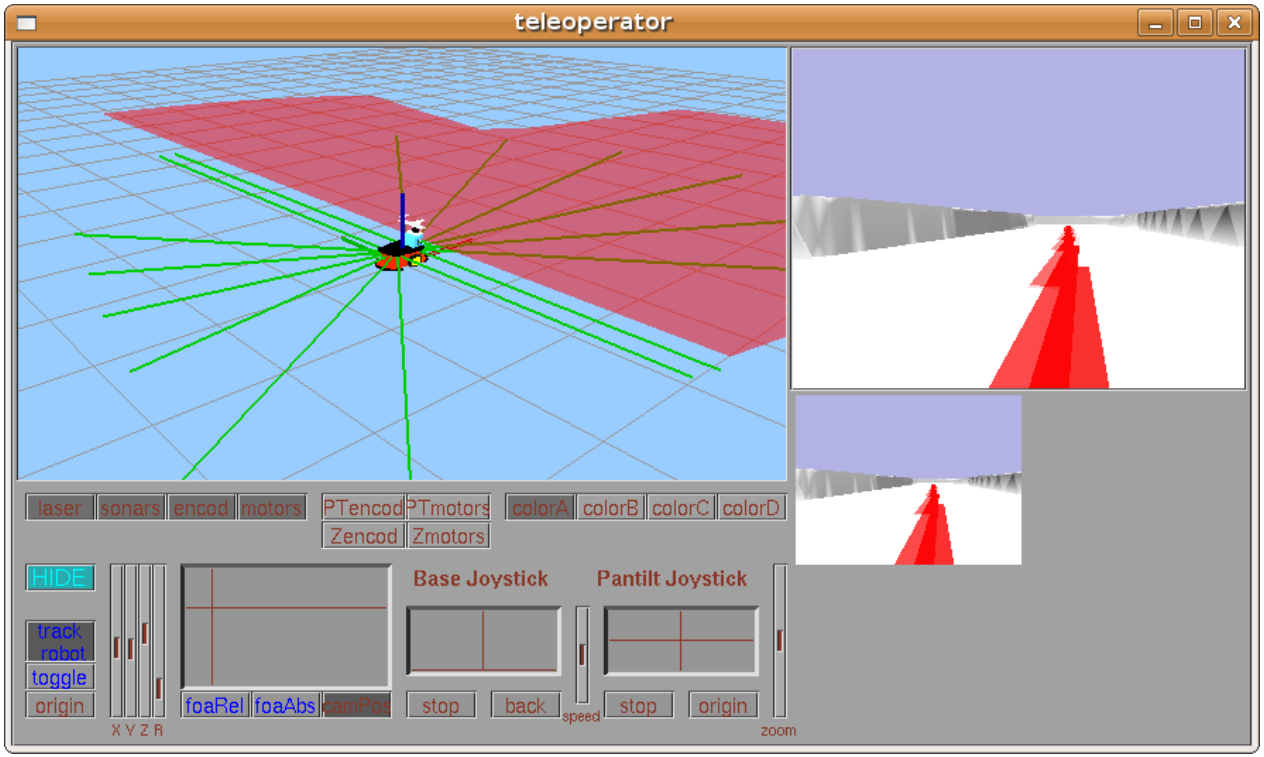
Robot de referencia: Pioneer

- Motores, sónares, láser, odometría, cuello mecánico, 2 cámaras
- Componentes: playerserver, gazebo-server, cameraserver



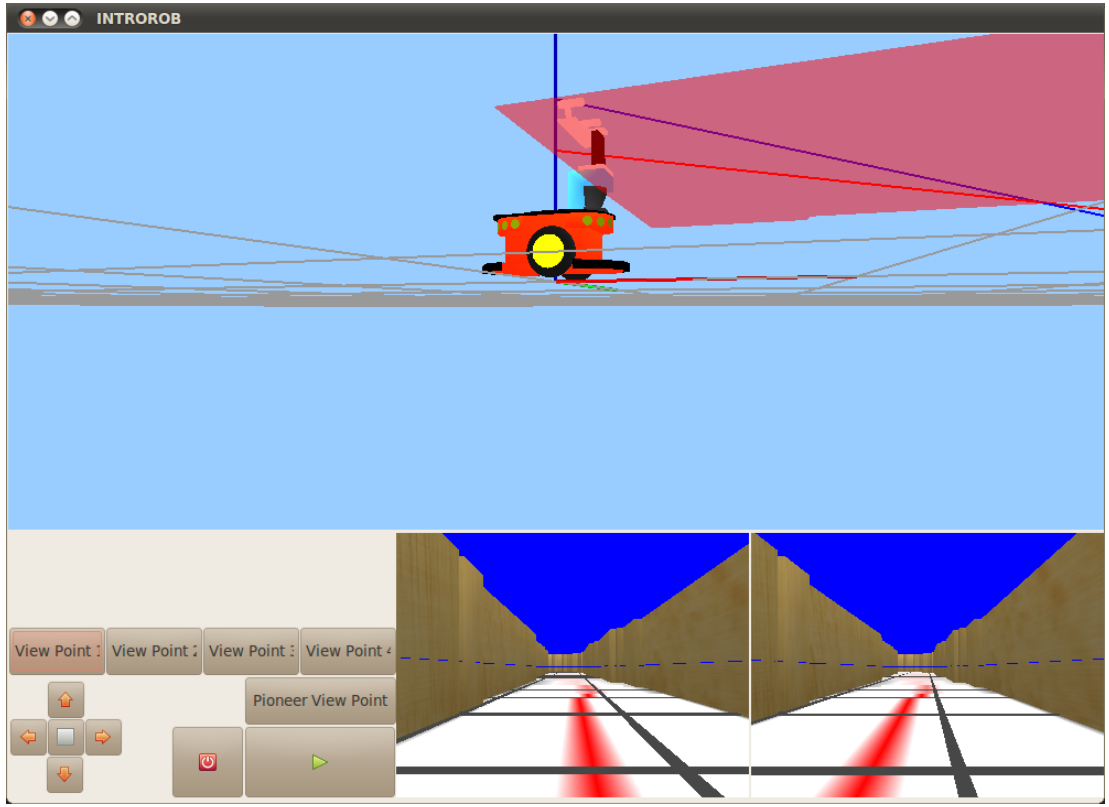
Componente Teleoperator

- Visualiza los sensores, cámaras, etc.
- Permite teleoperar los motores (base y cuello)



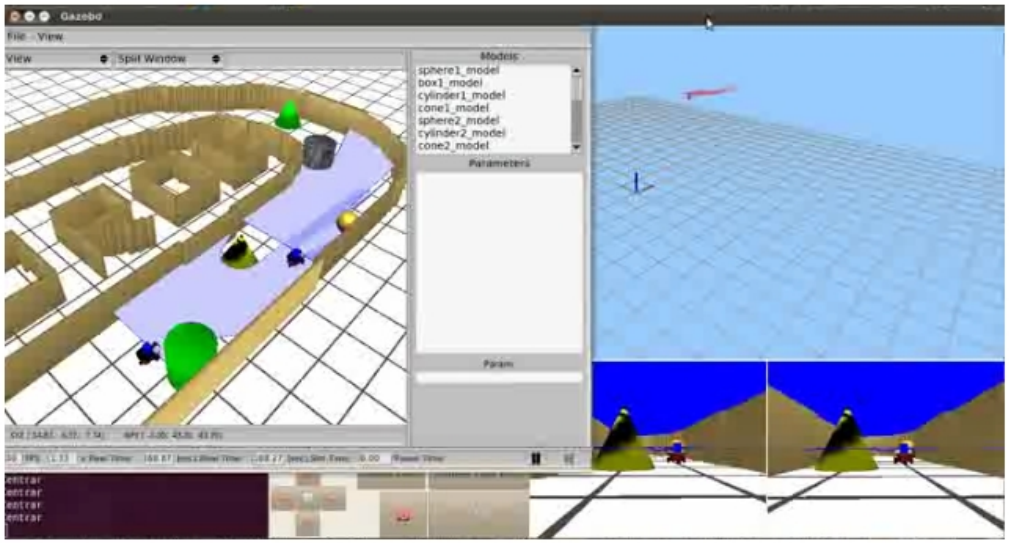
Componente Introrob

- Control
- Visualización 3D
- Cámaras
- Teleoperación
- Prácticas robótica



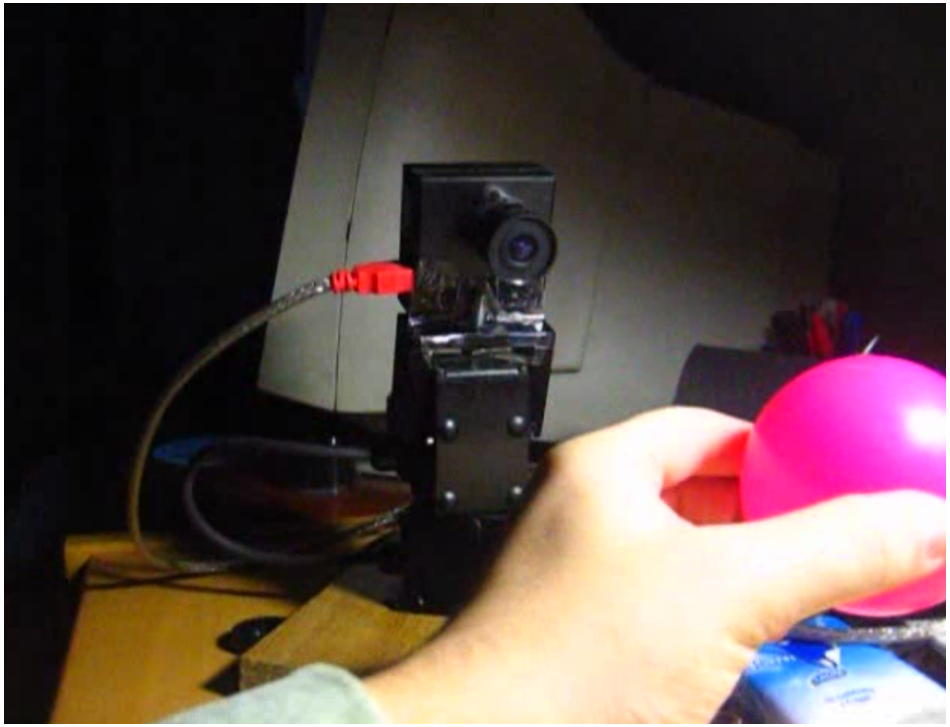
Docencia, prácticas robóticas

- Choca-gira
  (control reactivo, autómata)
- Comportamiento sigue-línea
  (control reactivo PID)
- Navegación VFF
- Carreras
  (navegación híbrida)
- El ratón y el gato



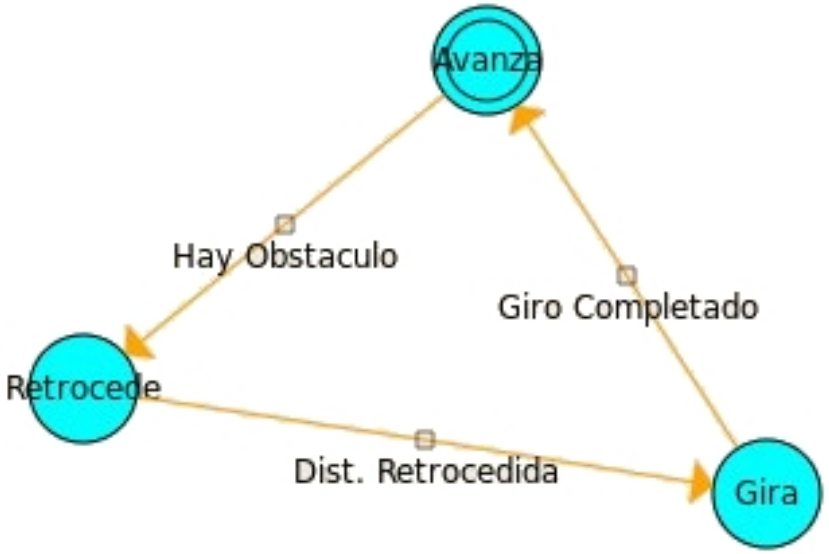
Taxia: FollowBall

- Percepción pelota
- Control reactivo PID
- Interfaces:
  imágenes, motores cuello



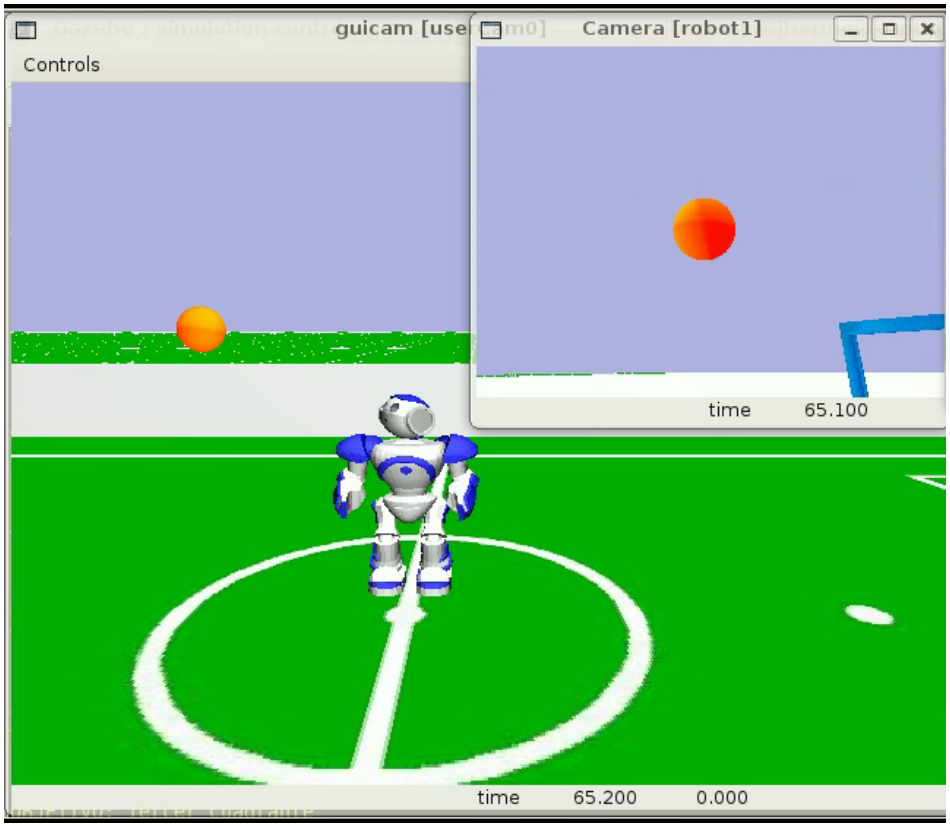
Editor de Autómatas de Estado Finito

- Comportamientos como FSM
- Diseño visual:
  estados y transiciones
- Genera automáticamente GUI



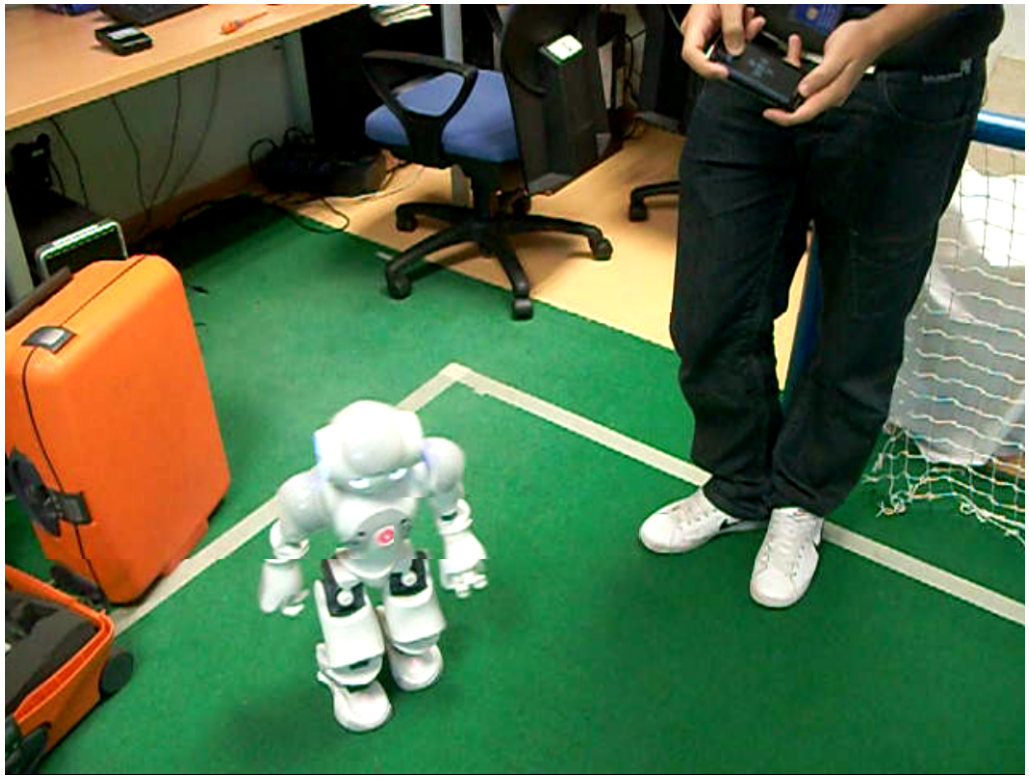
Humanoide Nao en simulador Gazebo

- Construcción del humanoide
- Motores en simulador
- Sensores en simulador
- Pieles
- Interfaces



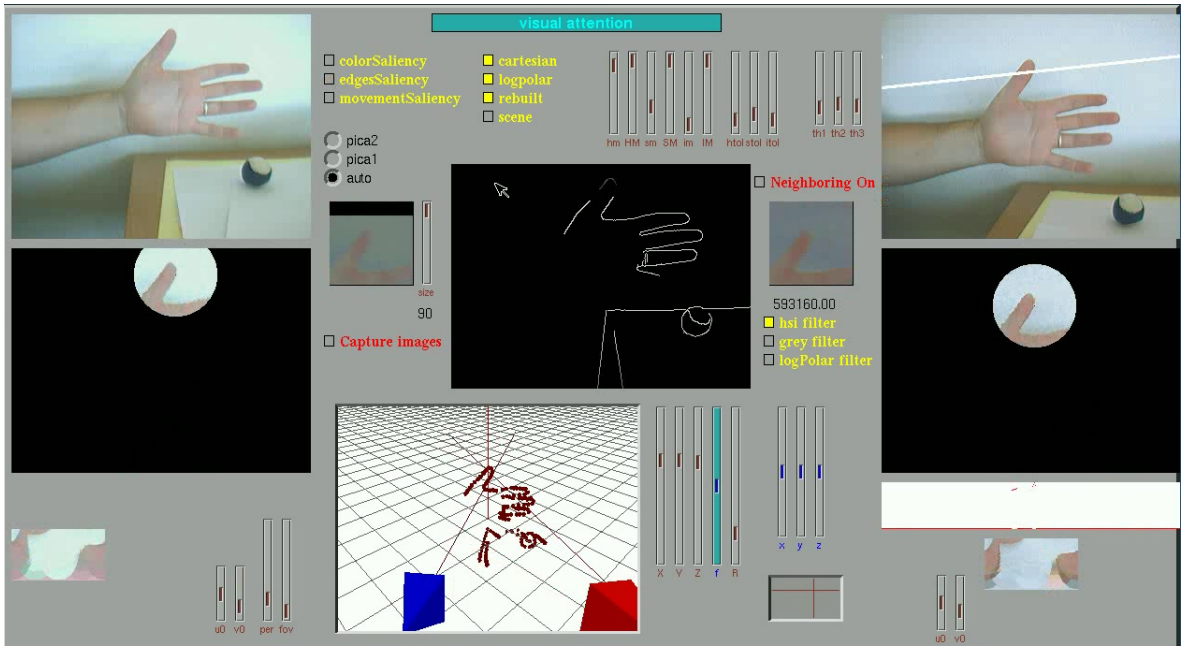
Teleoperación del humanoide Nao

- NaoDriver
- Sensores y motores
- Interfaces ICE
- Teléfono móvil Android

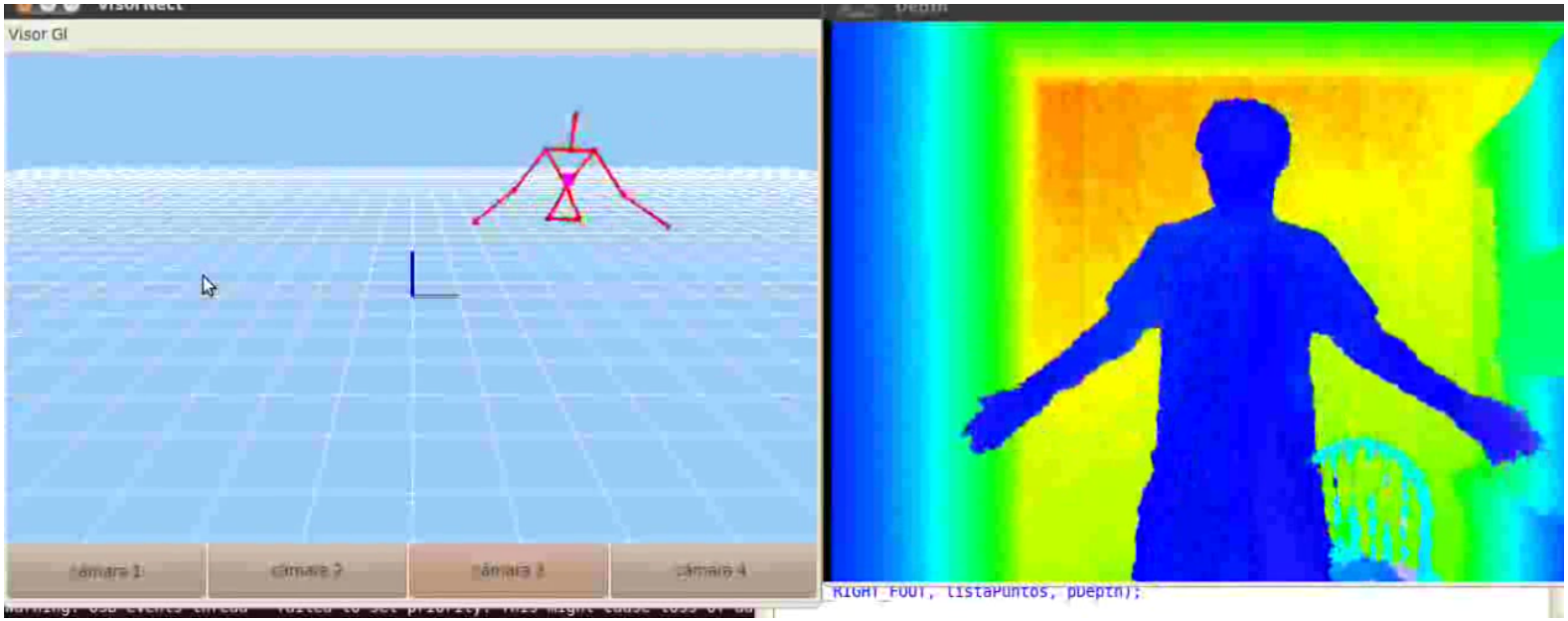


Reconstrucción 3D atentiva

- Robot con estereo
- Emparejamiento
- Línea epipolar
- Triangulación
- Atención visual



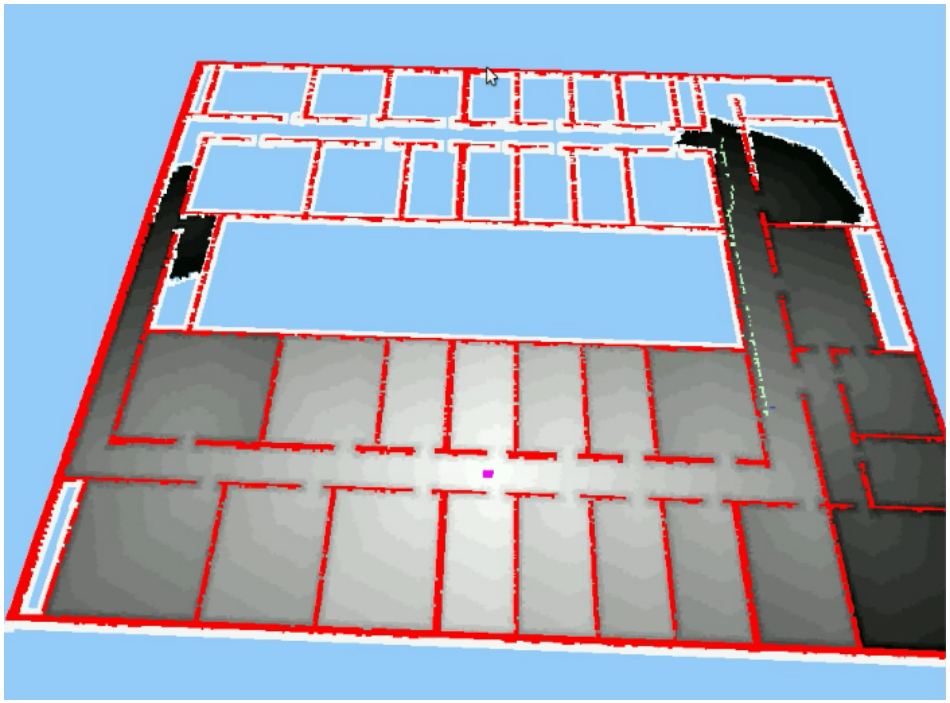
Componente VisorNect



- Visualiza imagen profundidad
- Visualiza puntos 3D
- Detecta cuerpo humano

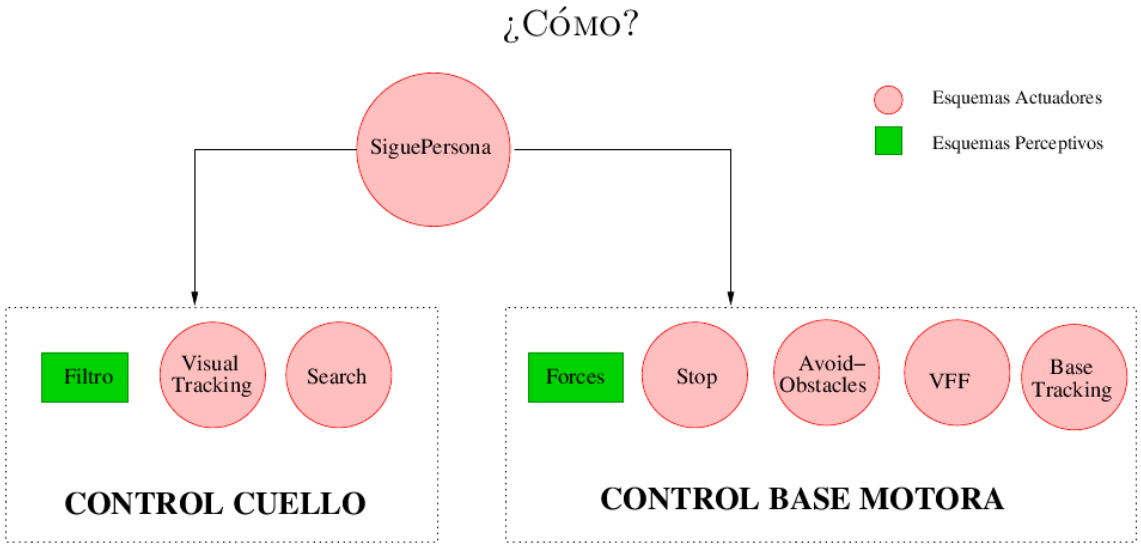
Navegación global con GPP

- Planificación caminos
- Campo de navegación
- Campo de obstáculos
- Navegación hibrída



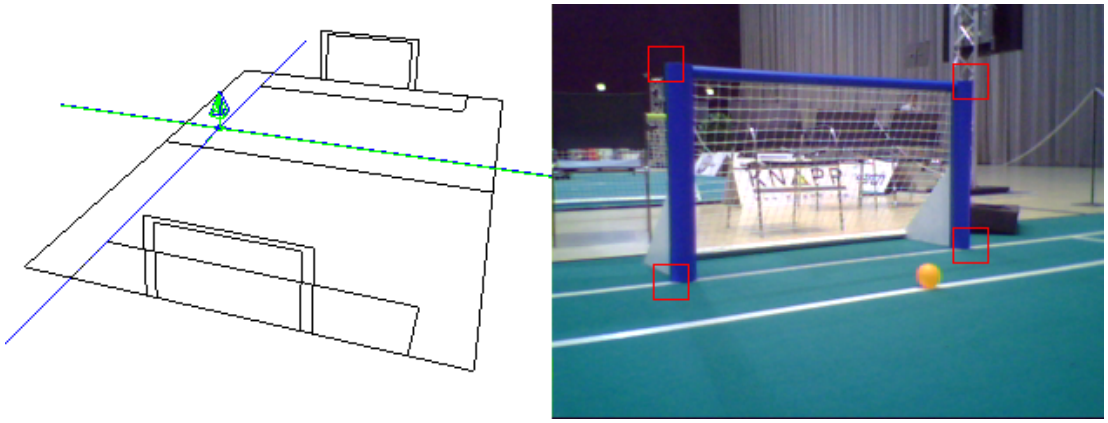
Comportamiento sigue-persona

- Persona en imagen
- No calcula distancia
- Sortear obstáculos
- Condiciones activación
- Sigue congénere



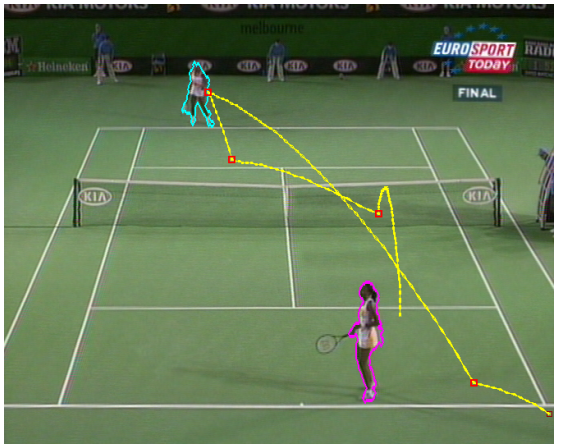
Autolocalización evolutiva

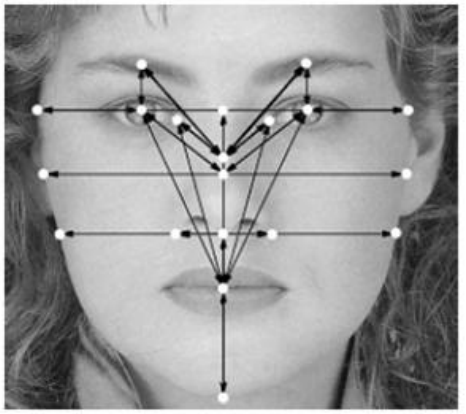
- El comportamiento adecuado depende de posición
- Robusto y rápido
- Continua
- Multimodal para simetrías
- Imagen teórica y observada



5. Aplicaciones de visión computacional

- Cámara, imágenes
- Procesamiento:
  bordes, transformadas, color, segmentación objetos, seguimiento...
- Reconocimiento de formas, caras, control accesos
- Videovigilancia
- Pares estéreo, reconstrucción 3D



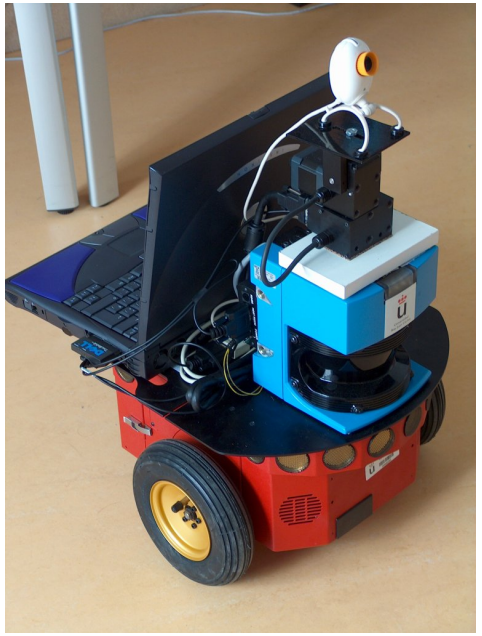


¿Qué es la visión computacional?

- Visión en robótica
- Componentes en jderobot de visión
- TrafficMonitor
- ElderCare
- MonoSLAM

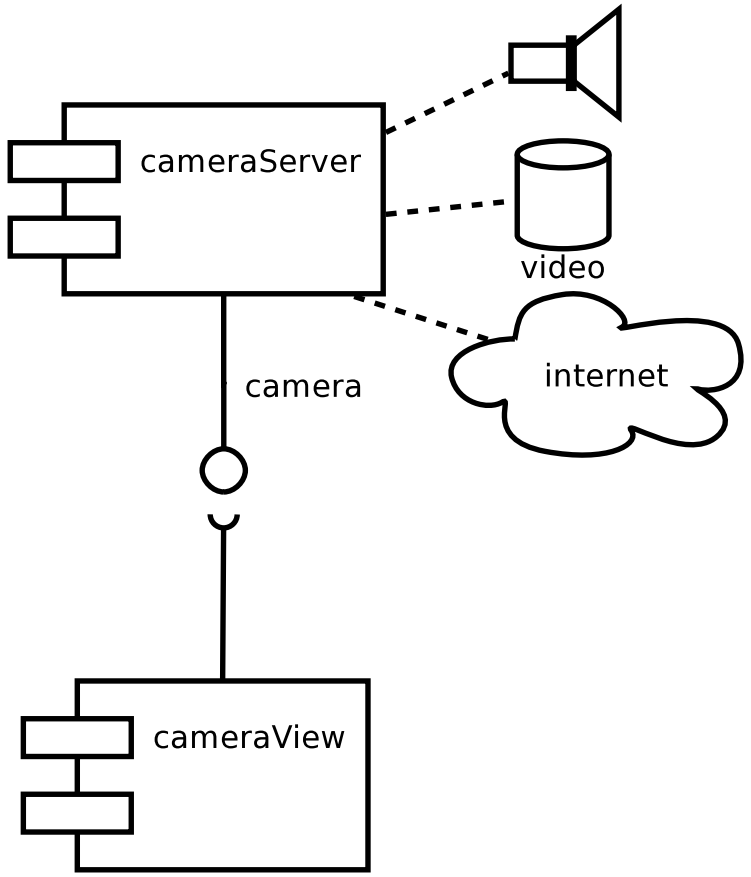
Visión computacional en robótica

- Es un sensor más: proporciona información del entorno
- Potencialmente muy rico
- MUY barato (cmos, webcam)
- Extraer información útil es complejo
- Flujo desbordante de datos
- Los robots más modernos la usan



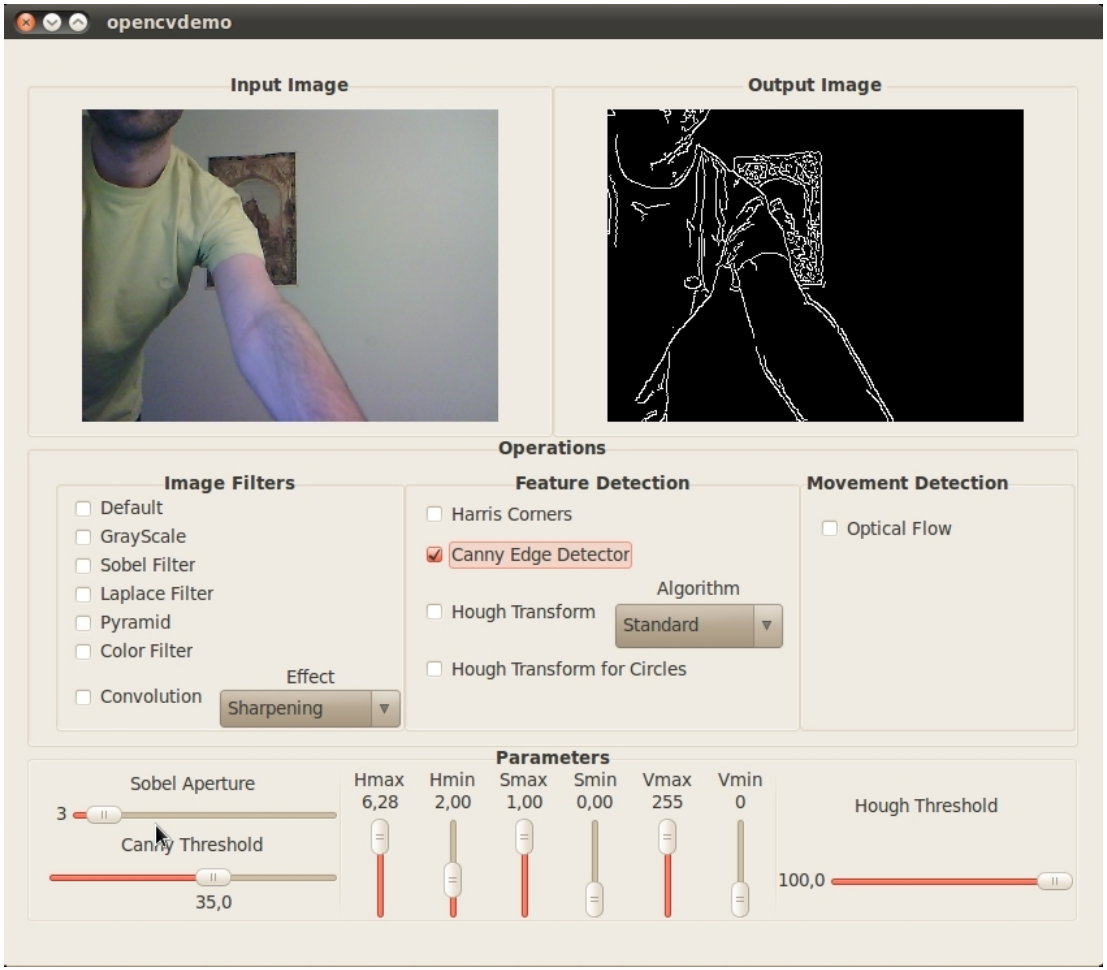
Componente CameraServer

- Captura imágenes y las sirve
- Interfaz cámara
- Admite ficheros
- Video4linux, firewire, cámarasIP



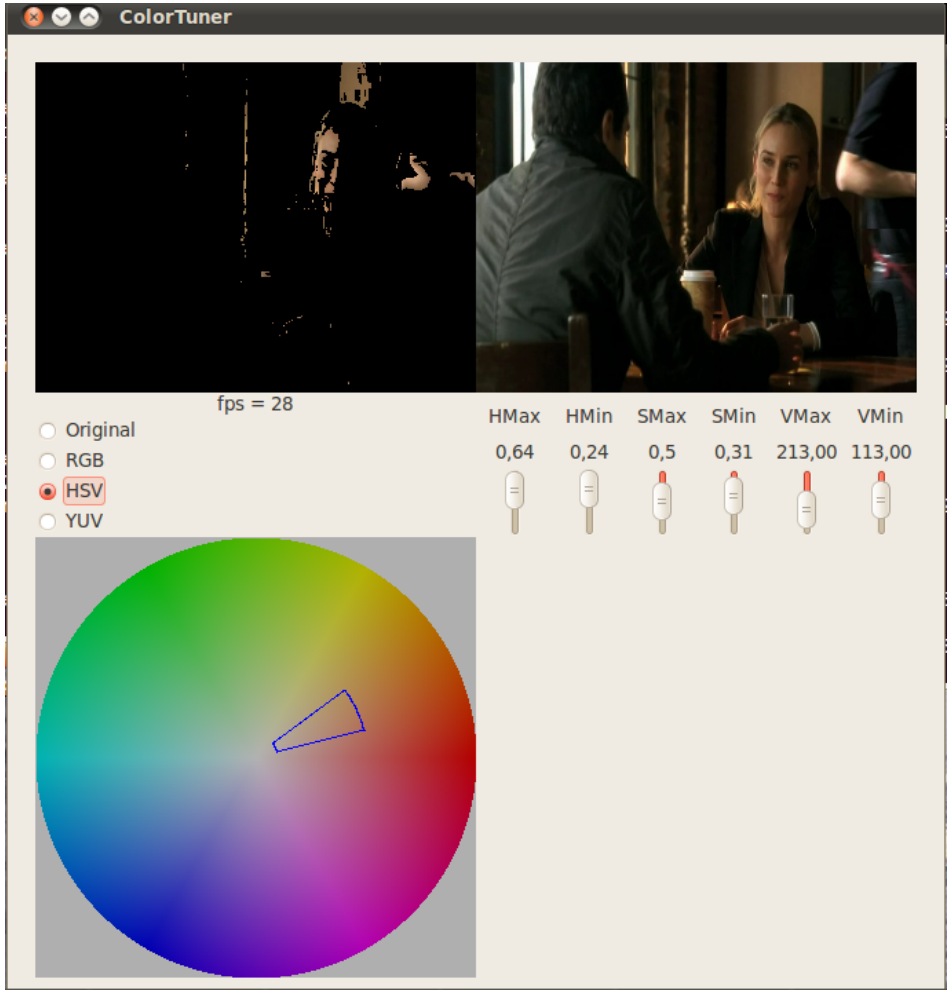
Componente Opencvdemo

- Procesamiento 2D
- Bordes
- Flujo óptico
- Convoluciones
- Segmentación



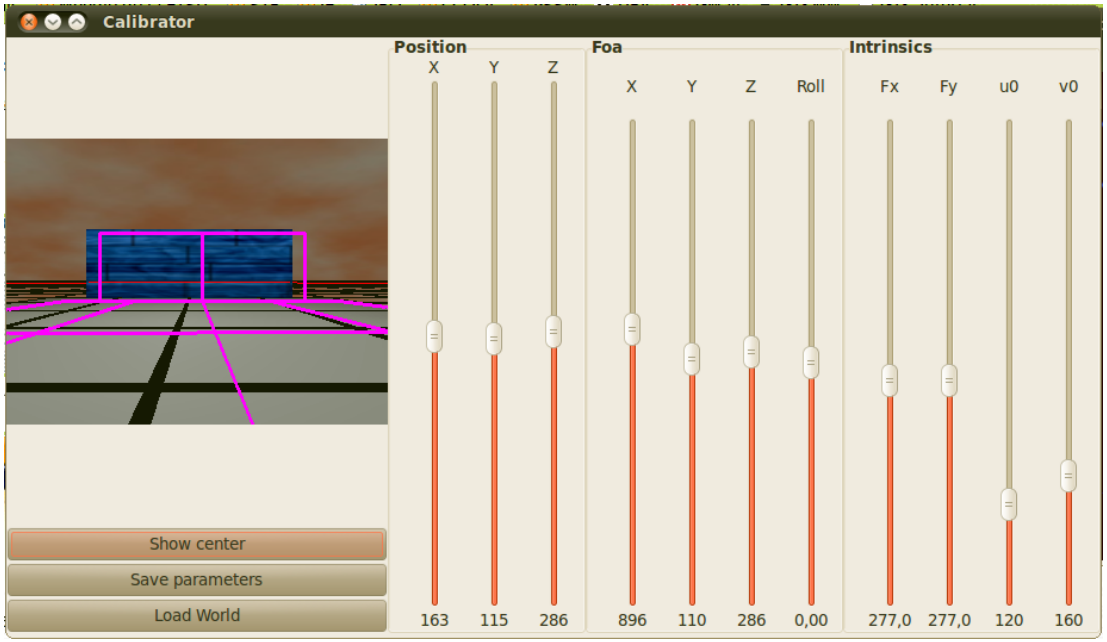
Componente ColorTuner

- Filtros de color
- Espacios RGB, HSV



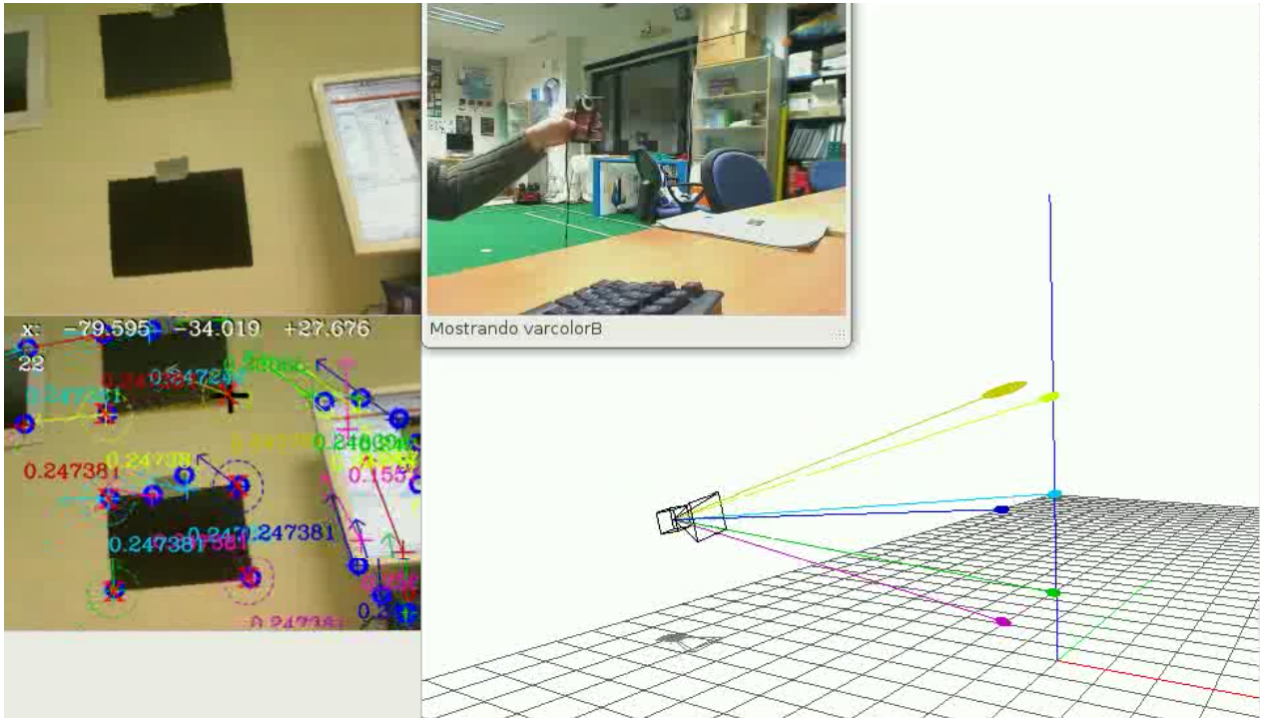
Componente Calibrator

- Cámaras calibradas para usar 3D
- Parámetros intrínsecos y extrínsecos
- Modelo *pinhole*
- Biblioteca Progeo
- Geometría proyectiva



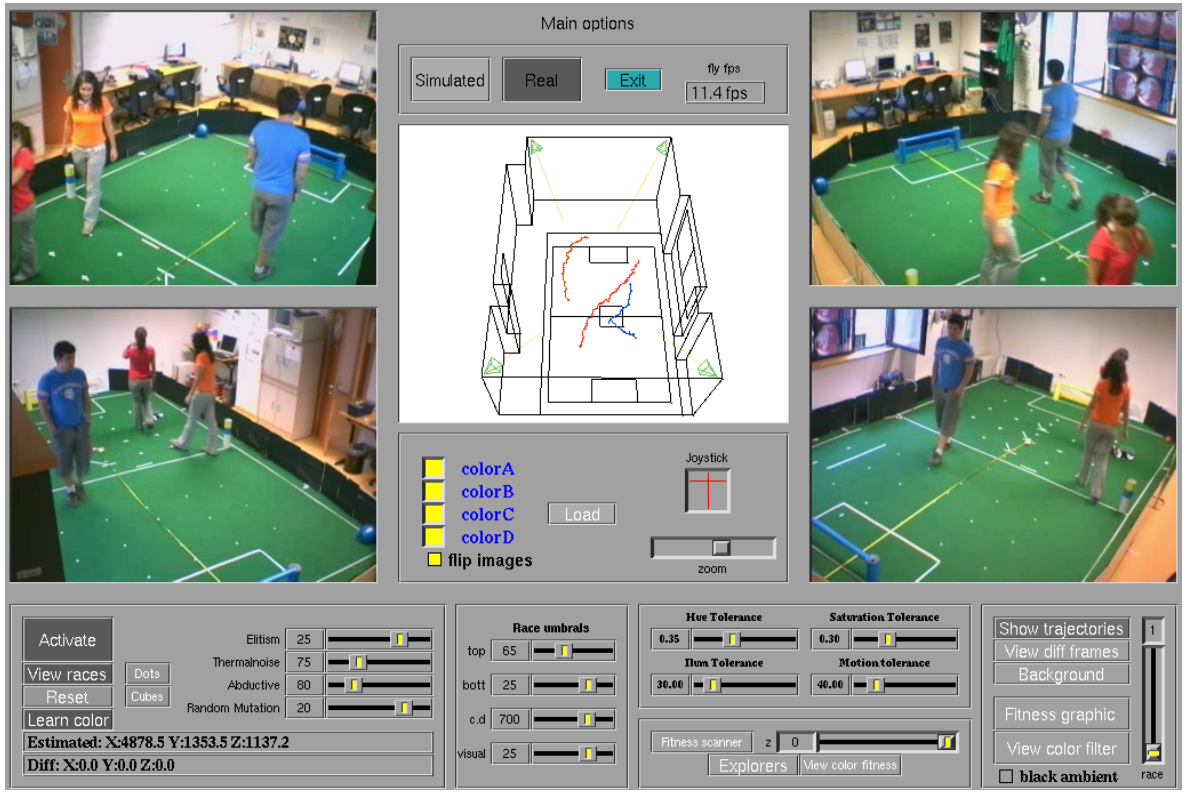
MonoSLAM

- Construye mapa y autolocaliza
- Filtro Kalman rápido
- Estado: puntos 3D y posición cámara
- Andrew Davison



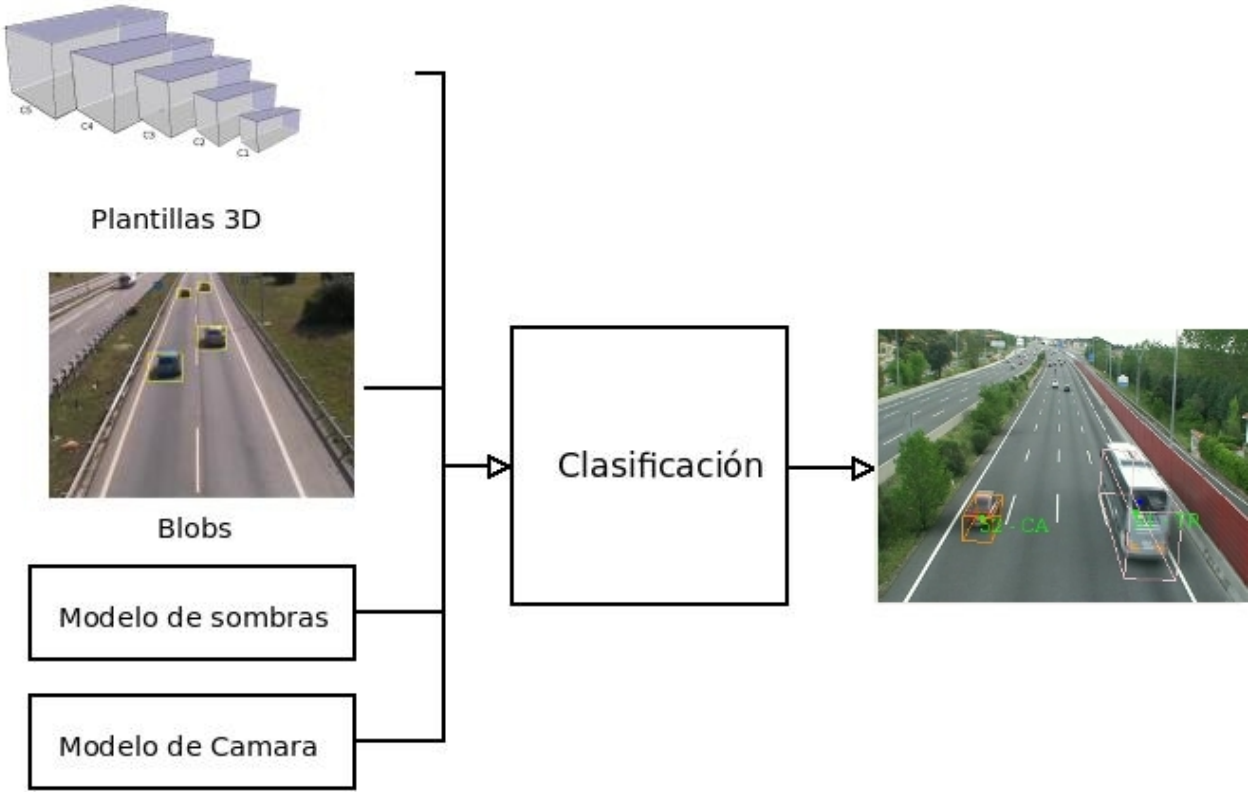
Aplicación ElderCare

- Detección de caidas
- Seguimiento 3D
- Color y movimiento
- Algoritmo evolutivo multi-modal
- Puntos y prismas



Aplicación TrafficMonitor

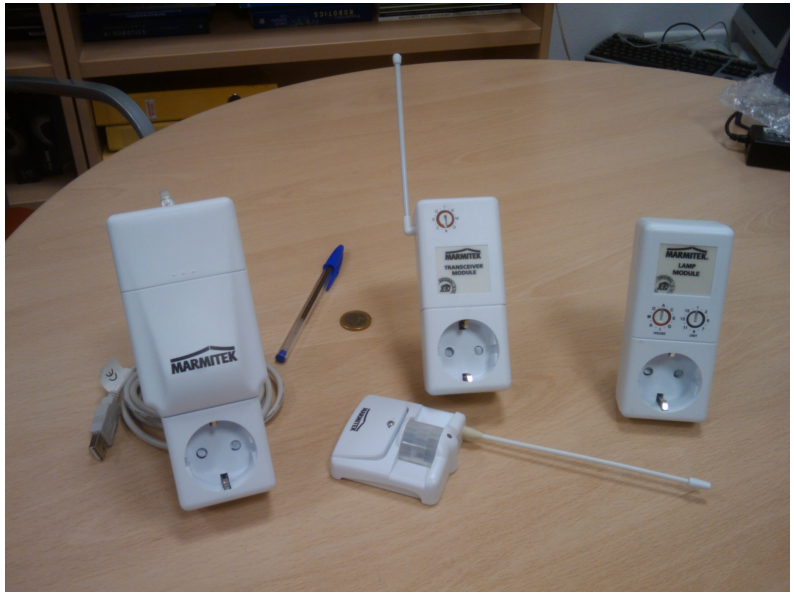
- Cámaras en túneles, carreteras...
- Cuenta vehículos
- Mide velocidad
- Clasifica
- Algoritmo evolutivo



6. Aplicaciones domóticas

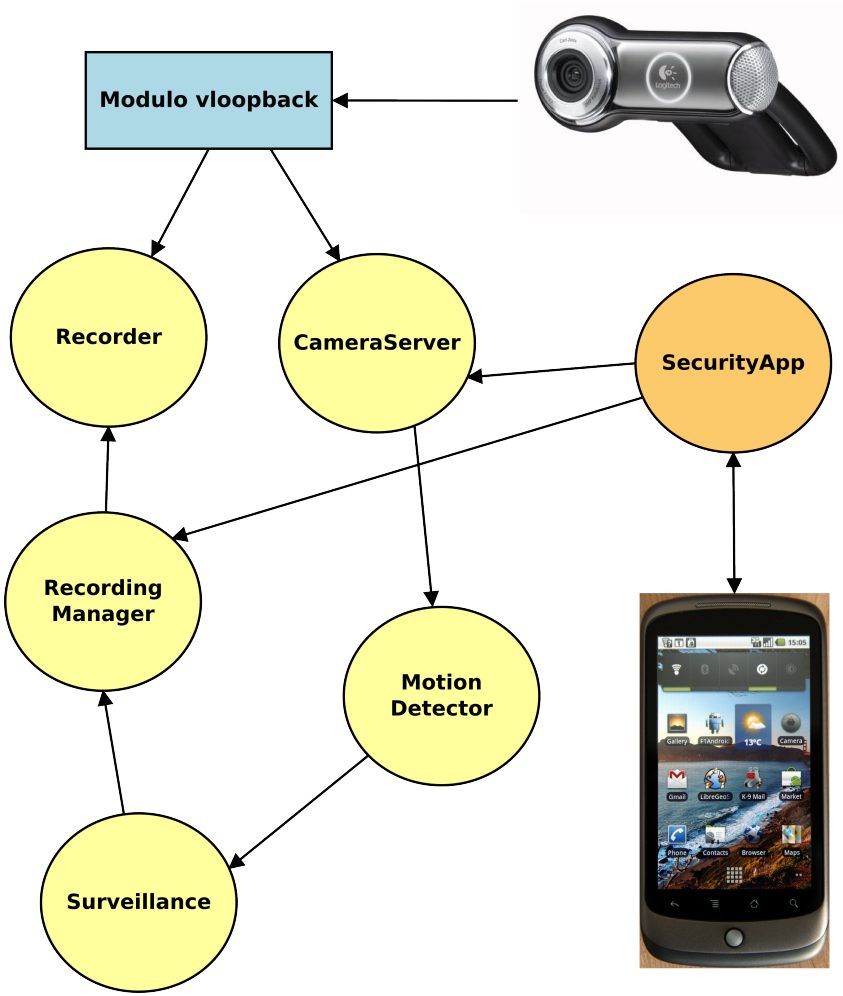
Domótica: conjunto de sistemas capaces de automatizar una vivienda, aportando servicios de *gestión energética*, *seguridad*, *bienestar* y *comunicación*, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. [Wikipedia]

- Protocolos X10, KNX
- Comunicación vía cables alimentación (PLC)
- Sensores domóticos: volumétricos, puertas, termómetros, humo, inundación, gas...
- Actuadores domóticos: alimentación discreta o gradual, motores, luces....
- Controlador
- Conectables a ordenador



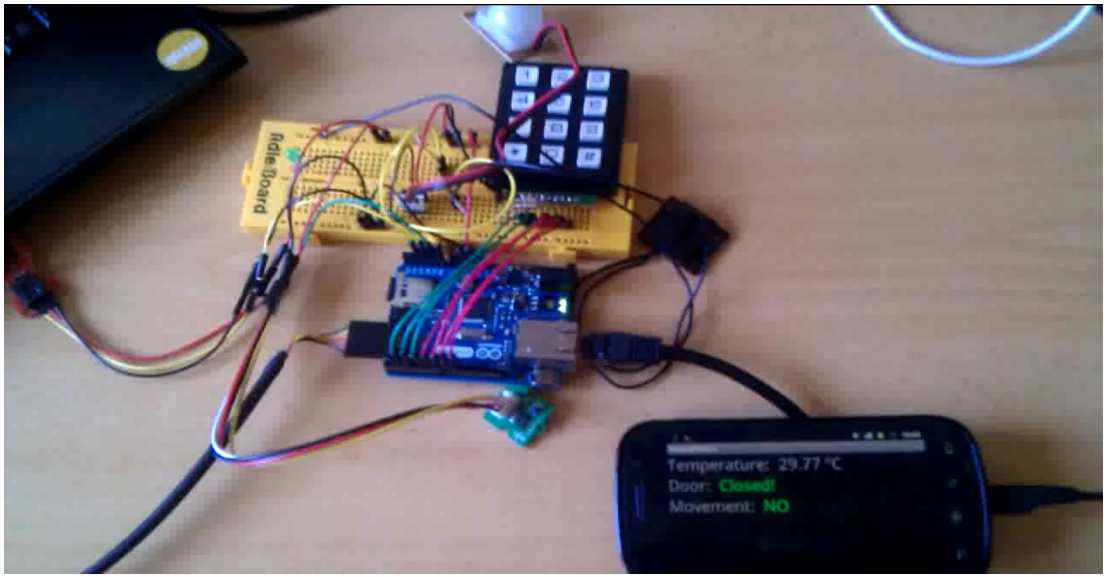
Surveillance-1

- Videovigilancia
- Cámaras
- Conexión con teléfono móvil
- Varios componentes



Surveillance-2

- Distribuido
- Arduino
- Inalámbrico, zigbee
- Bajo coste
- Teléfono móvil
- Kinect



7. Conclusiones

- Robótica, visión computacional, domótica
- Sensores, actuadores, inteligencia
- Inteligencia reside en el software
- Programar estos sistemas no es fácil, plataformas ayudan
- http://jderobot.org
  - Componentes
  - Interfaces
  - Distribución intermáquina (ICE)
  - Interoperabilidad lenguajes