PROGRAMACIÓN DE COMPORTAMIENTOS BÁSICOS PARA EL ROBOT HUMANOIDE NAO

"Tirar penaltis"

Alumno: Héctor Gómez Varela
Director: Carlos Vázquez Regueiro
Ingeniería técnica en Informática de Sistemas
Proyecto clásico de Ingeniería
Facultad de Informática

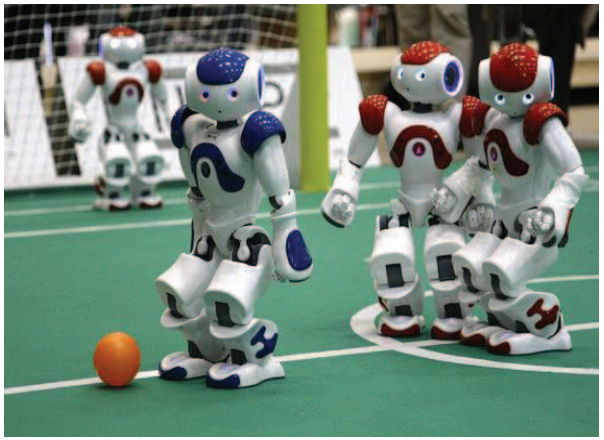
· A Coruña, 15 de septiembre de 2014 ·


UNIVERSIDADE DA CORUÑA

Diseñar comportamientos de autómatas en **entornos dinámicos**

Estudiar el diseño de comportamientos básicos para la **Robocup**

- Equipos de robots
- Se adaptan dinámicamente al entorno para jugar al fútbol



Diseño de comportamiento dinámico

Comportamiento que permita interacción con los humanos

Integrar diversos aspectos: visión, desplazamiento y respuesta...

PC local:

- ROS
- Python 2.7
- Webots 7.4.3

Nao:

- Hardware
- Software: Naoqi

Intel Atom 1’6 GHz/ 2 GB RAM

Dos cámaras

Sensores ultrasonidos

Sensores táctiles

Motores/ actuadores:

- 2 en la cabeza
- 8 en los brazos
- 2 en la manos
- 12 en las piernas

Choregraphe

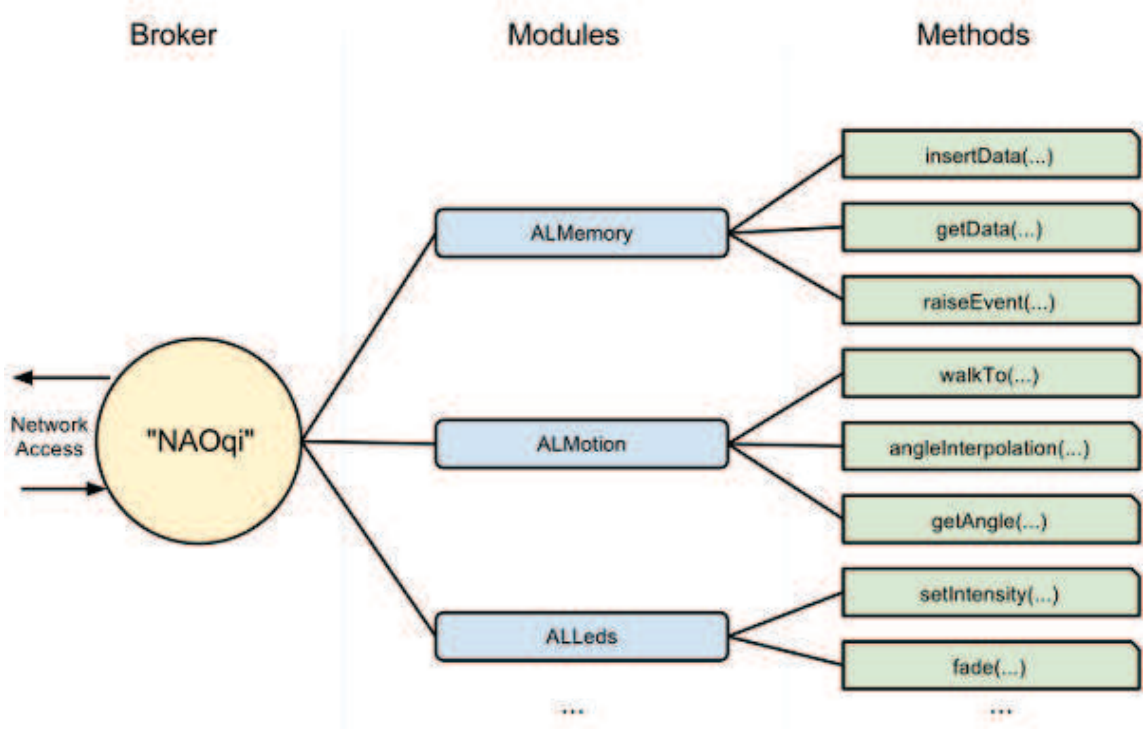
Naoqi: kernel y librerías

Acceso a módulos de control:

· ALMotion, ALMemory, ALVideoDevice

Usamos la versión 1.14.5

¿Cómo funciona Naoqi?



Sistema operativo robótico

Proporciona servicios semejantes a los de un sistema operativo ordinario

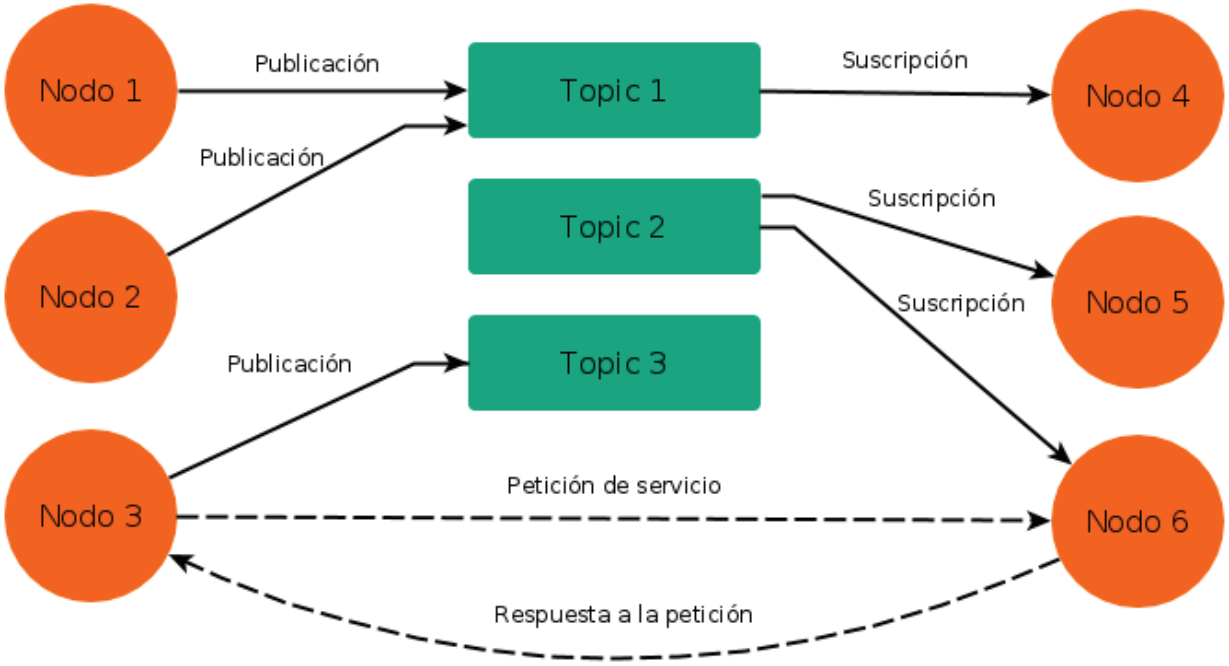
- Transporte entre procesos
- Sistema de gestión de ficheros
- Sistemas de compliación

Focalizado en la gestión de sistemas dinámicos a tiempo real con gran flujo de información

¿QUÉ ES ROS?

¿CÓMO FUNCIONA ROS?

- Nodos: ejecutables
- Topics: canales de información
- Mensajes
- Servicios



ELECCIÓN DE SOFTWARE

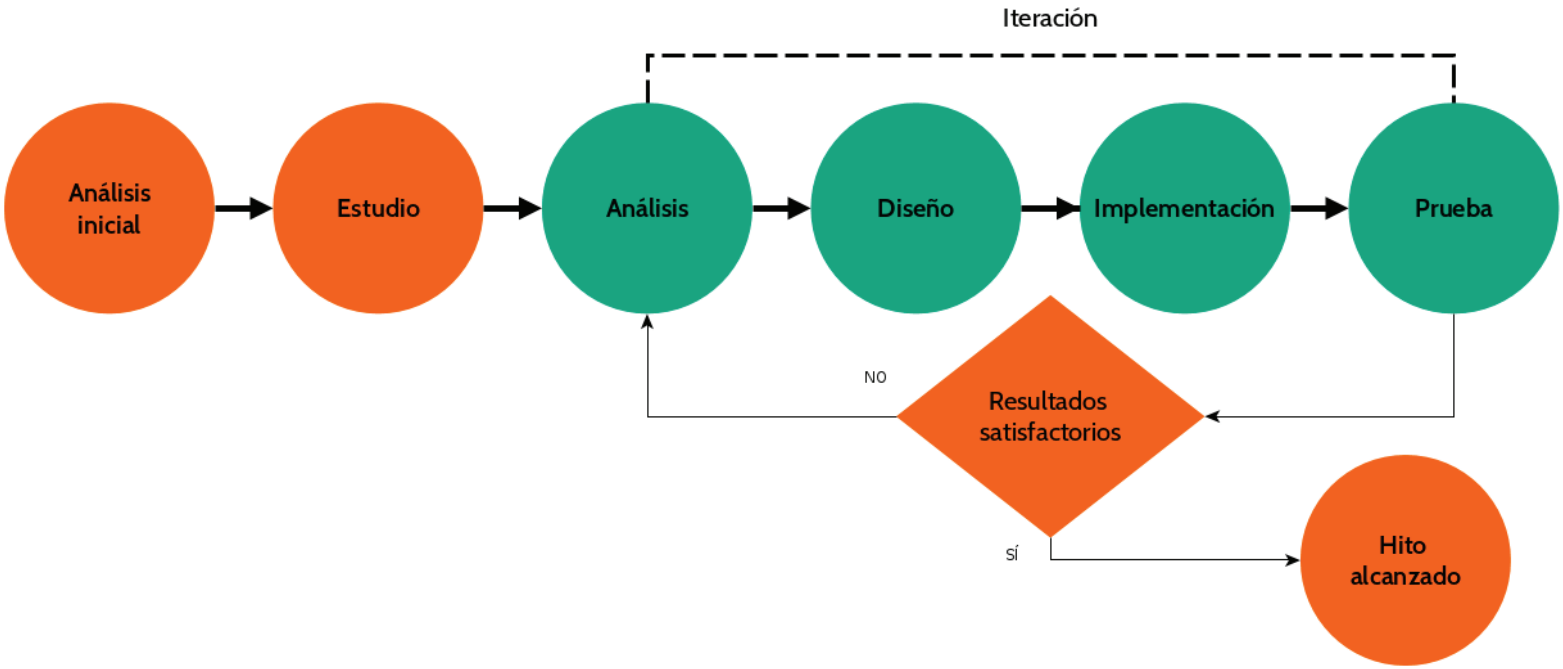
Simulación:

- Gazebo: sin modelo 3D URDF (unified robot description format)
- V-Rep: sin soporte para Naoqi
- Webots: software elegido

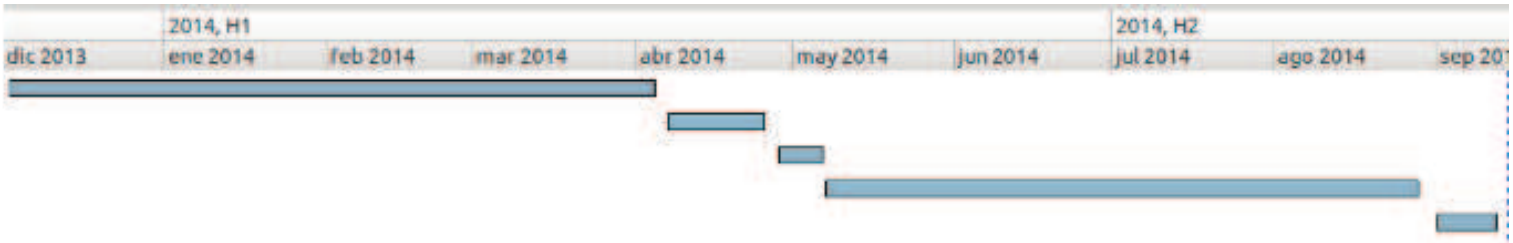
Detección:

- Detección mediante Naoqi: estable e integrada
- Detección mediante OpenCV: valorable para futuras implementaciones

Metodología ágil iterativo-incremental



PLANIFICACIÓN



Name	Start	Finish	Work	Duration
Estudio de la tecnología	dic 2	abr 4	90d	90d
Definición de tareas	abr 7	abr 25	15d	15d
Análisis inicial	abr 28	may 6	7d	7d
Iteraciones funcionales y pruebas en simulador	may 7	ago 28	82d	82d
Pruebas reales	sep 1	sep 12	10d	10d

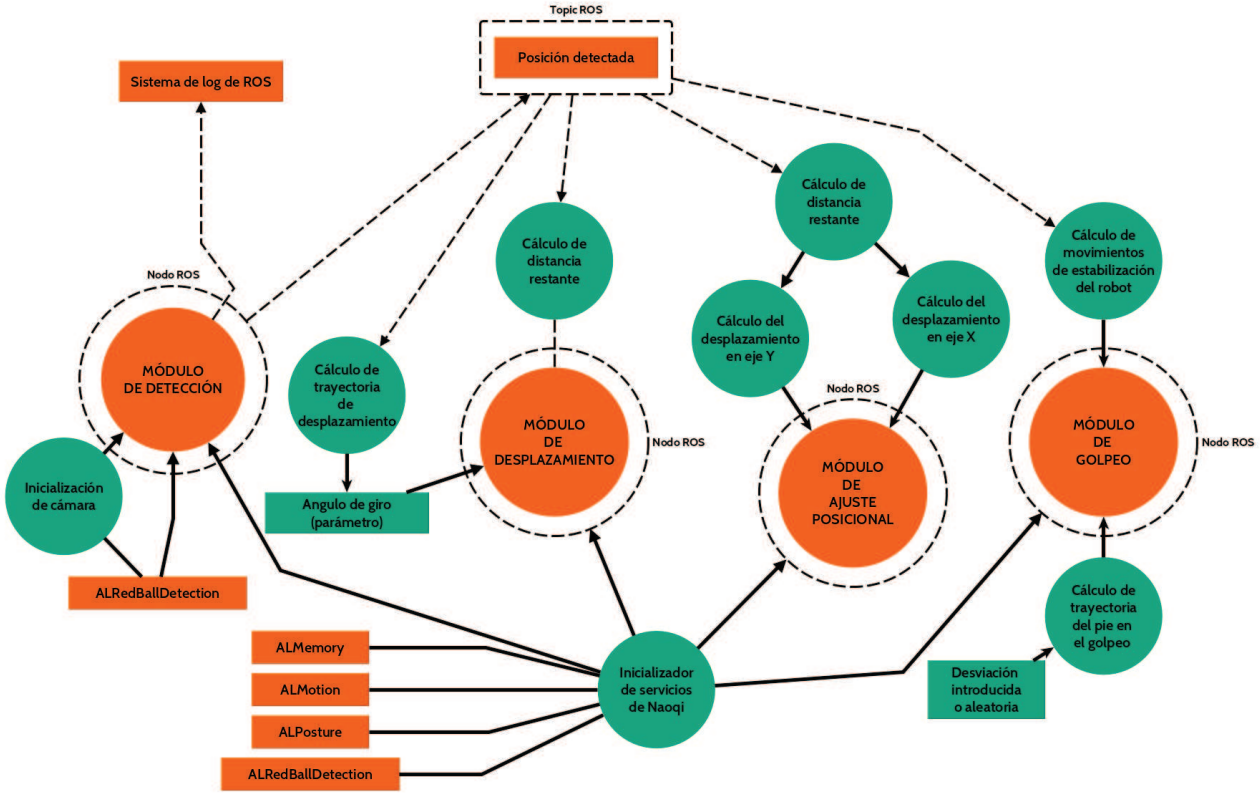
Basado en el software/ hardware del que se dispone

Lista incremental de objetivos

- Golpeo estable
- Detección
- Colocación
- Golpeo variable en base a percepción de objeto externo (v.g., portería)

Plenamente integrado en ROS

Cada una de las funcionalidades es un nodo ROS escrito en Python



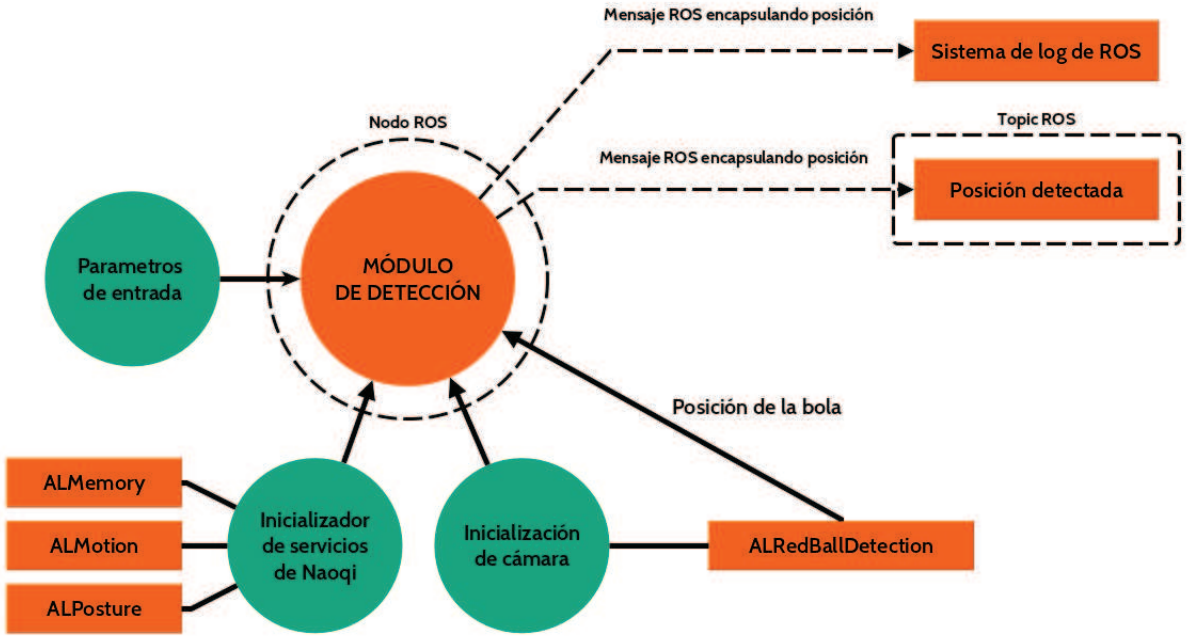
MÓDULO DE DETECCIÓN

Se implementa mediante Naoqi

Genera unas coordenadas respecto al centro de gravedad del robot

Transmite objeto Point a través de ROS

Permite almacenar a tiempo real la trayectoria de la pelota

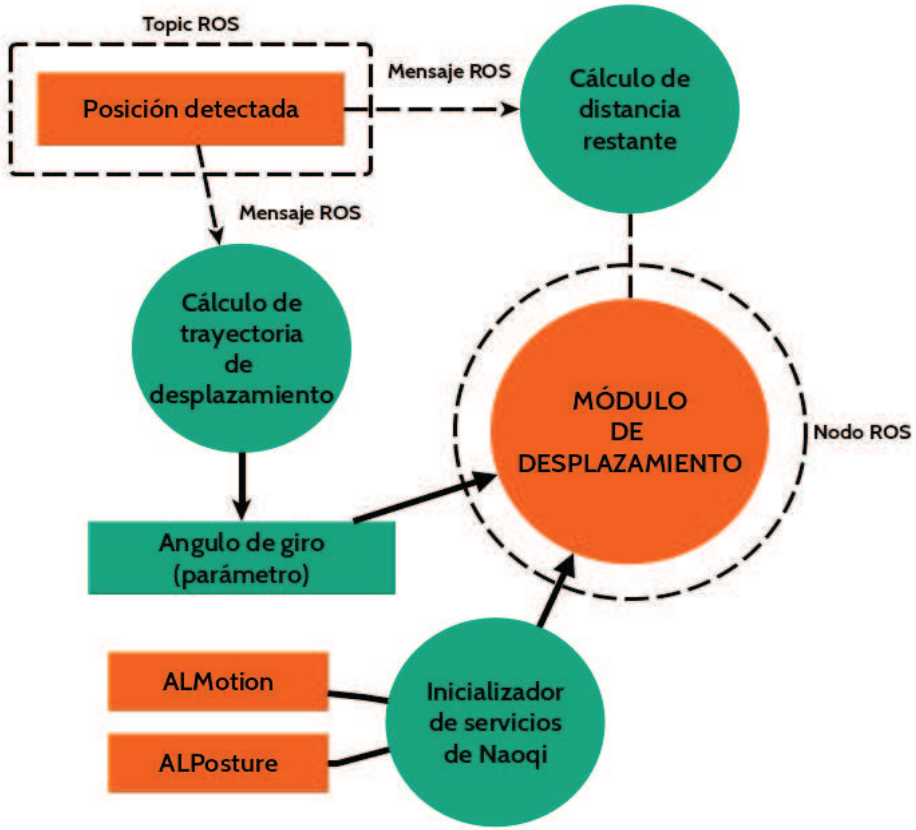


MÓDULO DE DESPLAZAMIENTO

Desplaza el robot de forma acorde a las coordenadas que recibe a través de ROS

Detecta la no-presencia de la pelota en el entorno

Modifica parámetro de módulo de detección para cambio de cámara activa

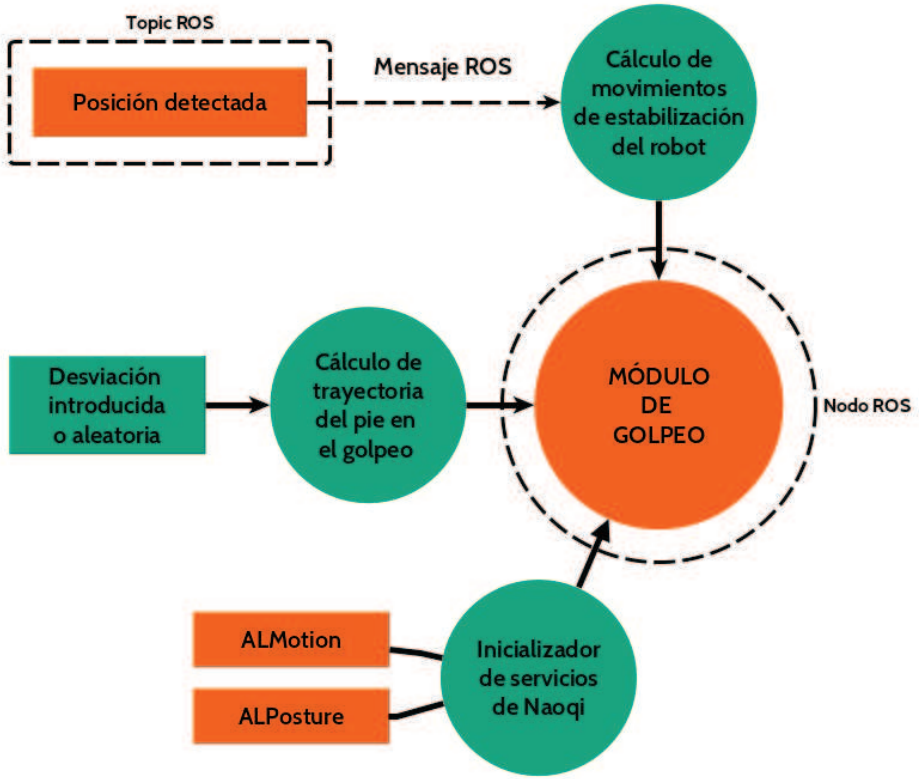


MÓDULO DE GOLPEO

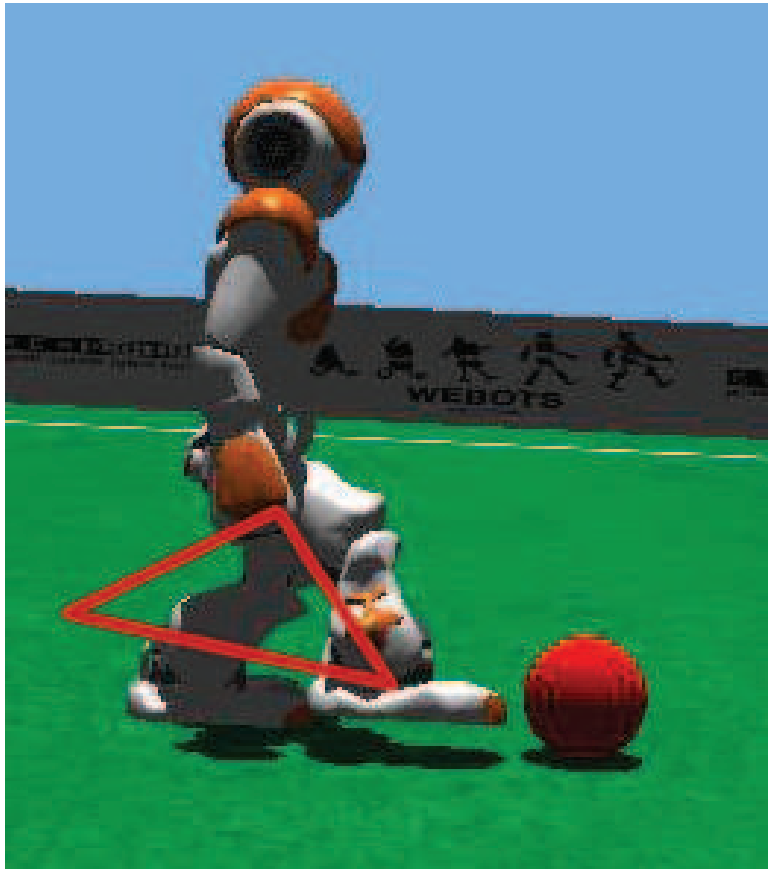
Recibe orden de golpeo del módulo de desplazamiento

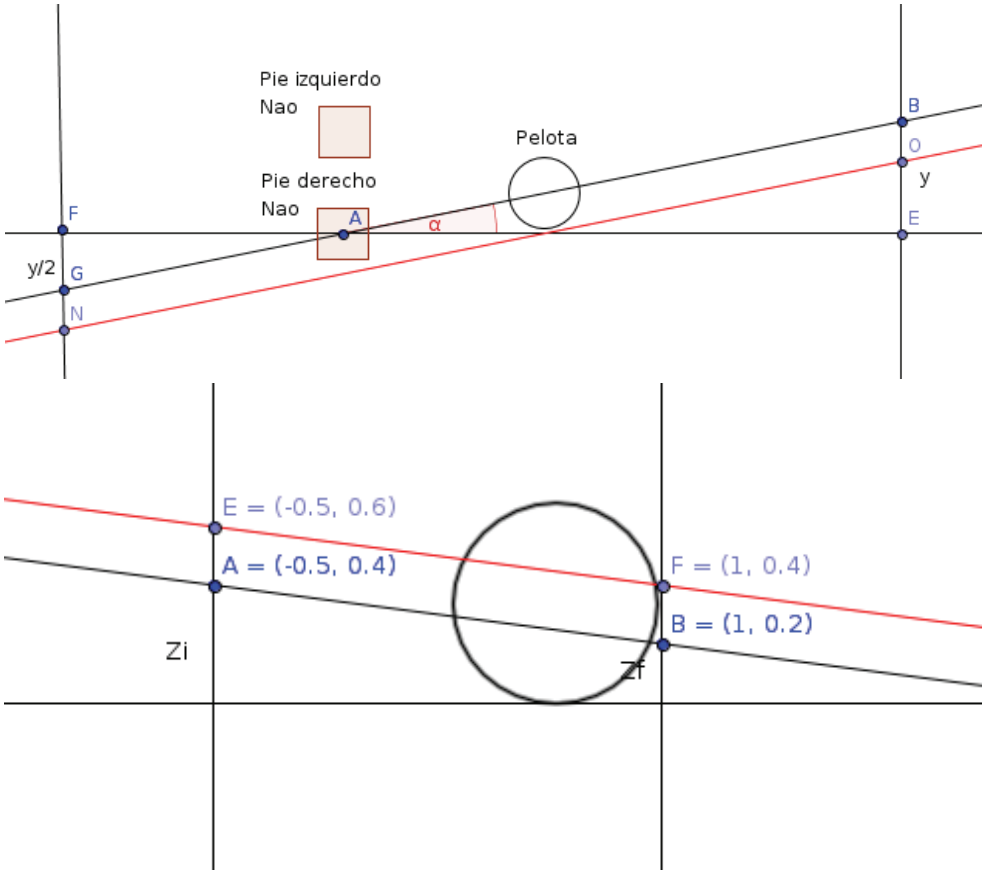
Genera un movimiento de estabilización del robot previo

Genera una trayectoria para el pie, introducida por usuario o aleatoria en cierto rango



MÓDULO DE GOLPEO





SIMULACIÓN

Proporcionalidad directa ángulo- trayectoria

Gran repetibilidad

ROBOT REAL

Dificultad para repetir las mismas condiciones del entorno

Los resultados difieren en las repeticiones

SIMULACIÓN

Algoritmo converge siempre que la bola es detectada

Posibilidad de trabajar con rangos de error razonablemente pequeños

ROBOT REAL

Deficiencias en la percepción

Dificultades para que el robot siga trayectorias precisas

Fallos en la detección motivados por la iluminación

Conclusiones:

Sistema estable

Sistema distribuído

Sistema extendible

Comportamiento obtenido conforma una unidad lógica

Trabajo futuro:

Detección de portería, mediante OpenCV o usando las herramientas de Naoqi

Incorporación de algoritmo de búsqueda

Incorporación de algoritmo de aprendizaje por refuerzo

Detección de posibles obstáculos usando sensores de ultrasonidos

Incorporación de una cámara externa (Kinnect o similiar)

Detección de otras formas y colores mediante OpenCV

Desarrollo de una interfaz gráfica

Gracias por su atención