DESARROLLO DE UNA ARQUITECTURA CLIENTE-SERVIDOR ASÍNCRONA BASADA EN EVENTOS POR NOTIFICACIÓN DE MENSAJES PARA LA COORDINACIÓN DE ROBOTS MÓVILES

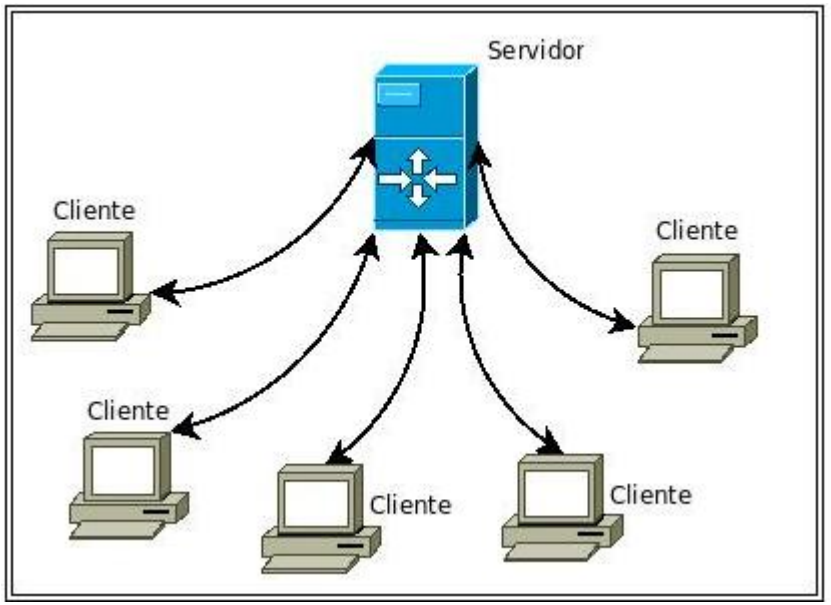
Francisco Luis Gómez Pérez
Máster en Automática e Informática Industrial

ÍNDICE

- Conceptos teóricos
  - Arquitectura cliente-servidor
  - Mecanismo de comunicación: Sockets
  - Tipos de Comunicación: Síncrona / Asíncrona. Ejemplos de uso.
  - Ejemplos prácticos:
    - Comunicación coordinada asíncrona de dos robots móviles
    - Robot recogedor
    - Limpieza inteligente
- Desarrollo práctico
  - Elementos de la arquitectura:
    - Robots móviles
      - Create de IRobot
      - Koala de K-Team
    - Sistema embebido IGEPv2
    - OMNIA i900 de SAMSUNG
    - BAM
    - PC cliente y Cámara cenital
  - Control del robot Create
    - Control por Bluetooth.
    - Control con el sistema IGEPv2
  - Generación de trayectorias.
  - Procesamiento de imágenes.
  - Control de trayectorias.
  - Esquema de comunicaciones.
- Resultados
- Conclusiones
- Trabajos futuros.
- Artículo y Referencias.

Arquitectura cliente-servidor

- Los clientes realizan peticiones a los servidores para realizar alguna tarea o solicitar información.
- Atendiendo a quien procesa el trabajo, los clientes y servidores pueden ser activos o pasivos:
  - Activos. Realizan el trabajo indicado.
  - Pasivos. Sólo procesan información.

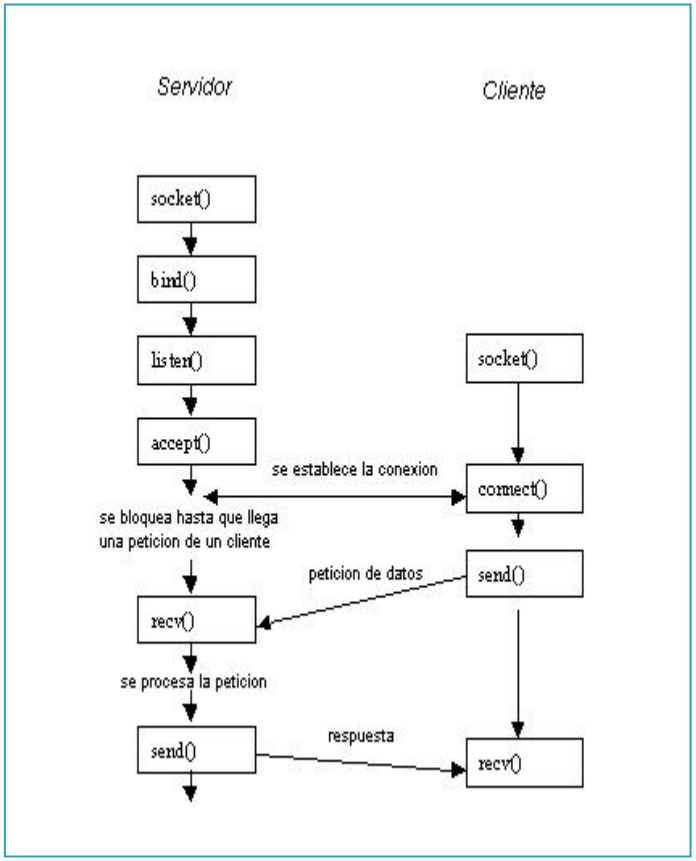


Sockets (I)

- Proporcionan una comunicación punto a punto entre dos procesos. Existen diferentes tipos:
  - **Sockets STREAM**. Hace uso del protocolo TCP que provee un flujo de datos bidireccional, orientado a conexión, secuenciado, sin duplicación de paquetes y libre de errores.
  - **Socket DATAGRAM**. Hace uso del protocolo UDP, el cual provee un flujo de datos bidireccional, no orientado a conexión, en el cual los paquetes pueden llegar fuera de secuencia, puede haber pérdidas de paquetes o pueden llegar con errores.

Sockets (II)

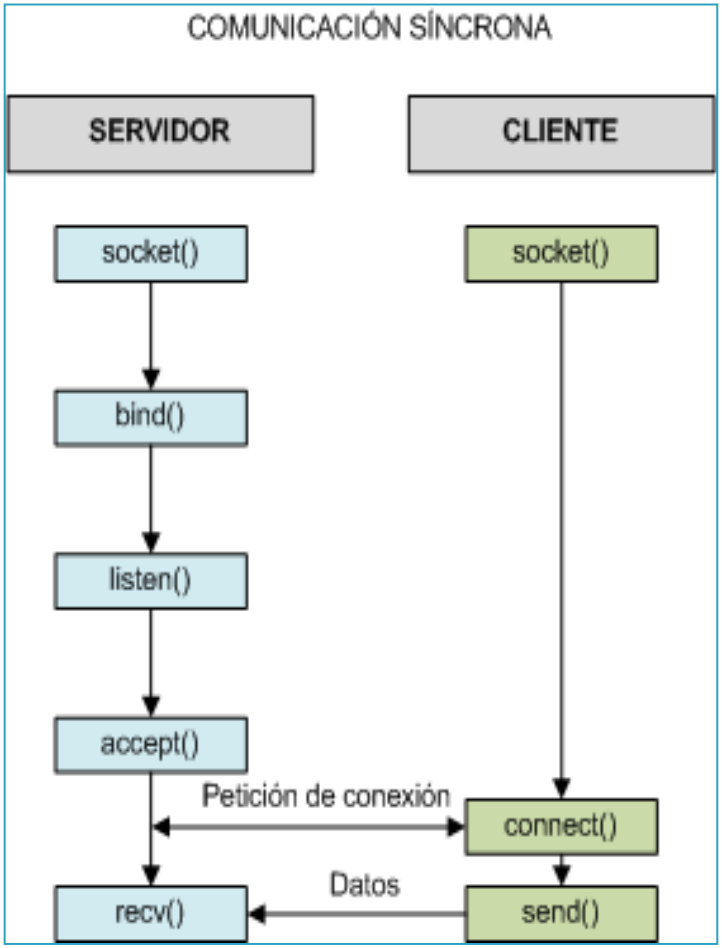
- Esquema básico de funcionamiento
  - **socket()**. Crea el socket.
  - **bind()**. Nombre del socket (IP y Puerto).
  - **listen()**. Habilita para recibir conexiones.
  - **accept()**. Llamada bloqueante. Espera a que un cliente se conecte al servidor.
  - **connect()**. Inicia la conexión con el servidor.
  - **send()**. Enviar datos.
  - **recv()**. Recibir datos.

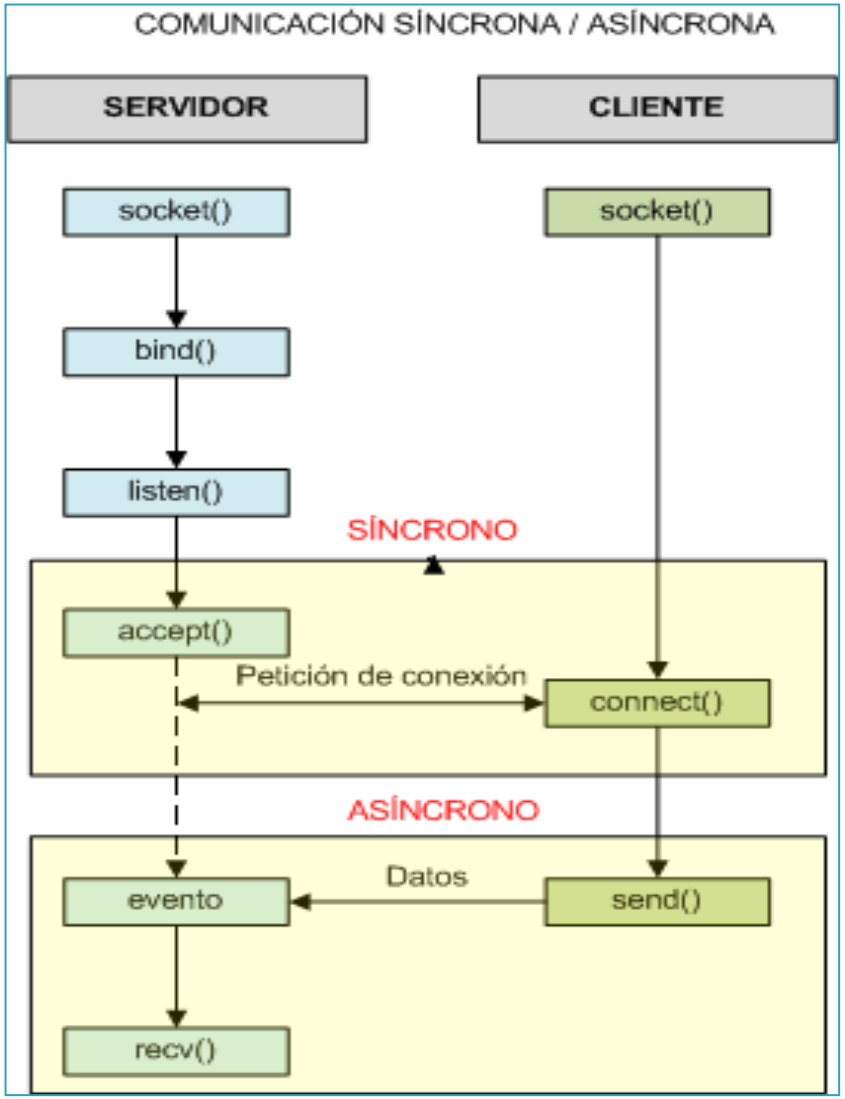


Tipos de Comunicación (I)

- **Comunicación síncrona**. Programación secuencial donde las llamadas Accept() (petición de conexión de un cliente) y Recv() (recepción de datos) son bloqueantes. Similar a una conversación telefónica.
- **Comunicación asíncrona**. Las llamadas NO son bloqueantes y la programación es mediante eventos o señales. Similar a interrupciones.
  - Necesario disponer de algún tipo de mecanismo para tratar la comunicación asíncrona: EVENTOS por notificación de mensajes.

Tipos de Comunicación (II)





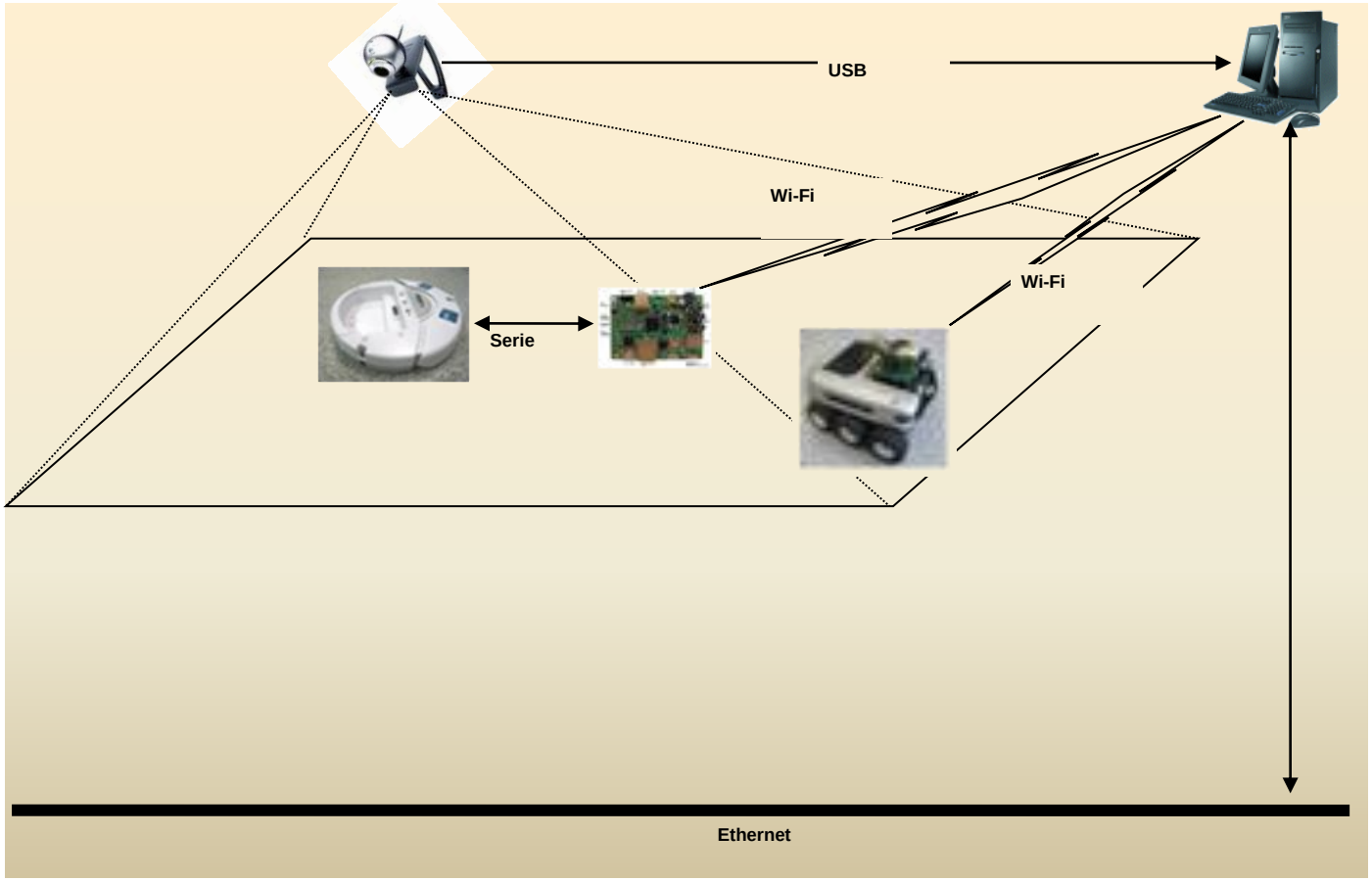
Tipos de Comunicación (III)

- Mecanismos para indicar que el socket es asíncrono:
  - WINDOWS (Winsock2.h)
    - int **WSAAsyncSelect**(SOCKET *s*,HWND *hWnd*,unsigned int *wMsg*,long *lEvent*).
      - s → Socket.
      - hWnd → Manejador que identifica la ventana donde se recibirá el mensaje cuando ocurra el evento.
      - wMsg → Mensaje que se recibe cuando ocurre el evento.
      - lEvent → Eventos que se quieren manejar con este tipo de comunicación:
        - FD_READ
        - FD_WRITE
        - FD_CONNECT
        - FD_CLOSE
        - FD_ACCEPT
  - UNIX
    - **fcntl**(sd, F_SETFL, O_ASYNC | O_NONBLOCK);
    - **fcntl**(sd, F_SETOWN, getpid());

Tipos de Comunicación (IV)

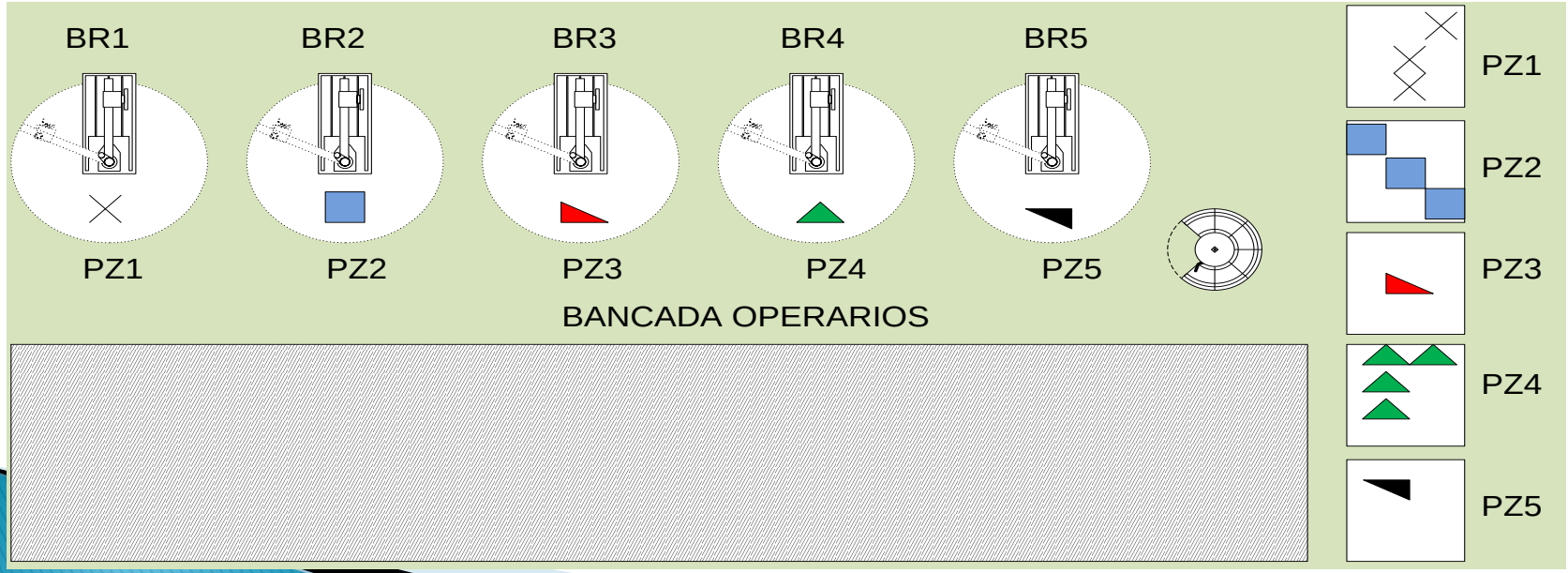
- Ejemplo de comunicación asíncrona
  - WSAAsyncSelect(s,principal,msg1, FD_READ);
  - ON_MESSAGE(msg1, Recepcion_CREATE)
  - Recepcion_CREATE(wParam,lParam) {
  - …
  - Recv(s,datos,tam(datos))
  - …
  - }
- Ejemplo de comunicación síncrona
  - WSAAsyncSelect(s,principal, 0, 0);
  - WSAIoctl(s, FIONBIO, …,…);

Comunicación coordinada asíncrona de dos robots móviles



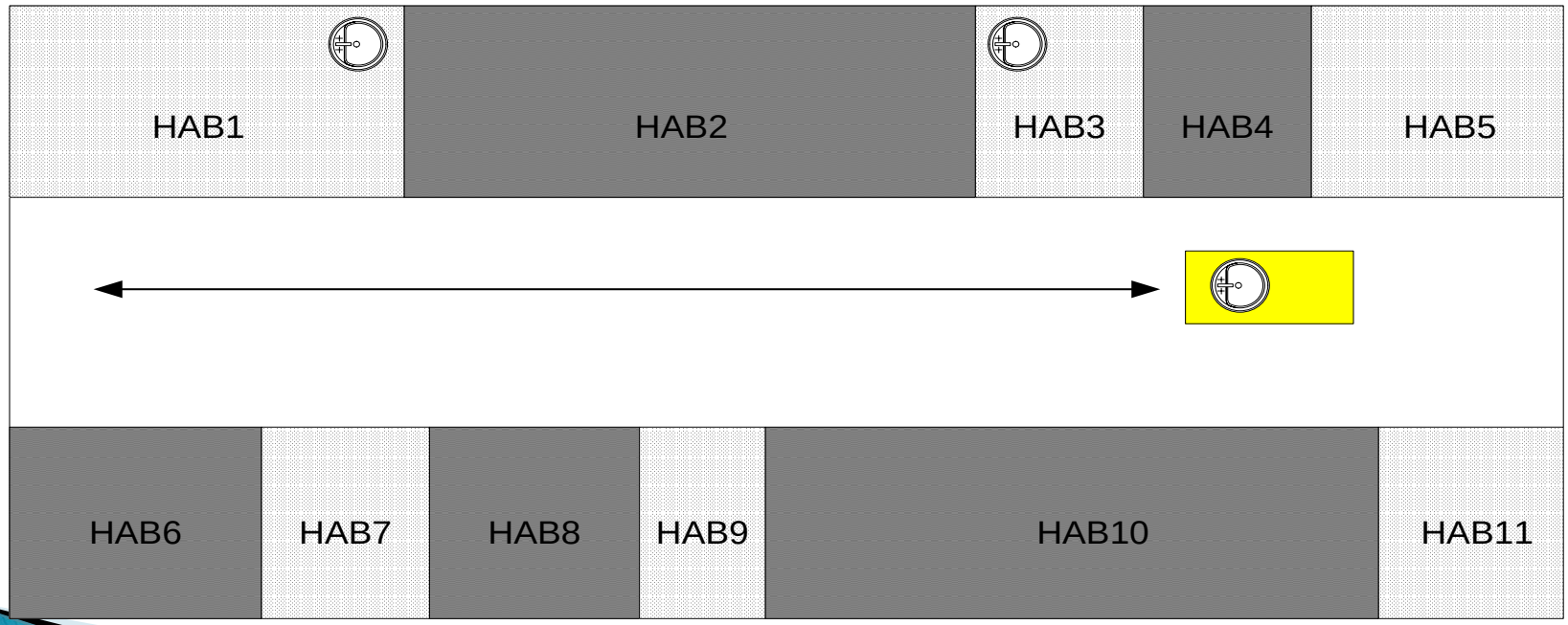
Robot Recogedor

- Entorno industrial con 5 brazos robóticos, cada uno termina una pieza diferente.
- Objetivo: Llevar las piezas a los 5 palets correspondientes. No hay espacio para instalar una cinta transportadora.



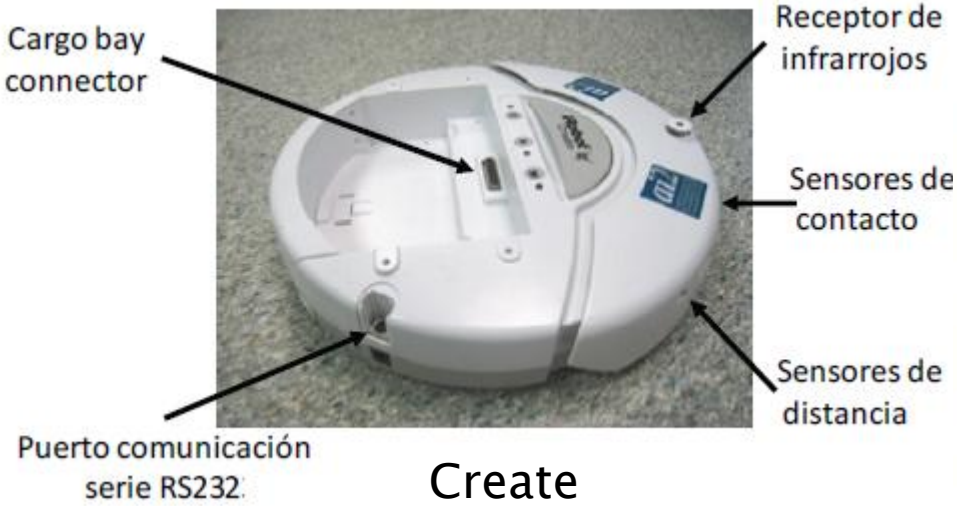
Limpieza Inteligente

- Objetivo: Limpiar (aspirar el suelo) un edificio completo con varias plantas. Tarea coordinada con varios robots y un robot transportador.

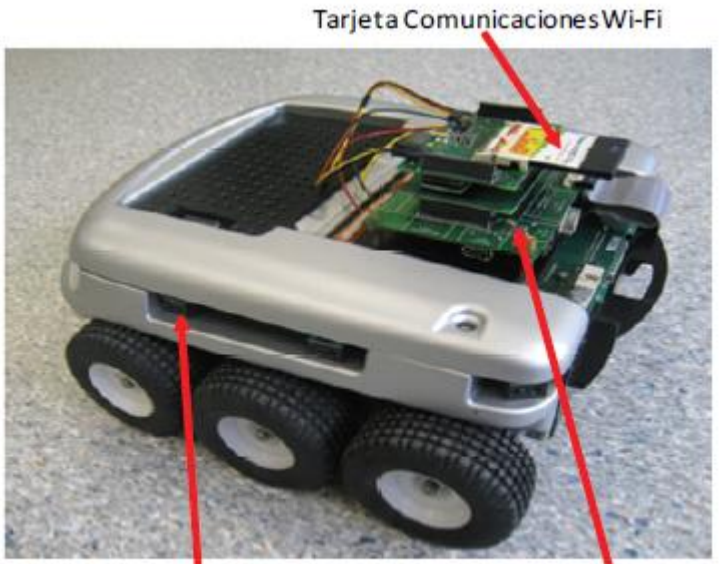


Robots móviles

- Objetivo: Desarrollar una aplicación que necesite la coordinación de varios robots móviles. Ejemplo: Aplicación de limpieza mediante el trabajo coordinado del Create de IRobot y del Koala de K-Team.



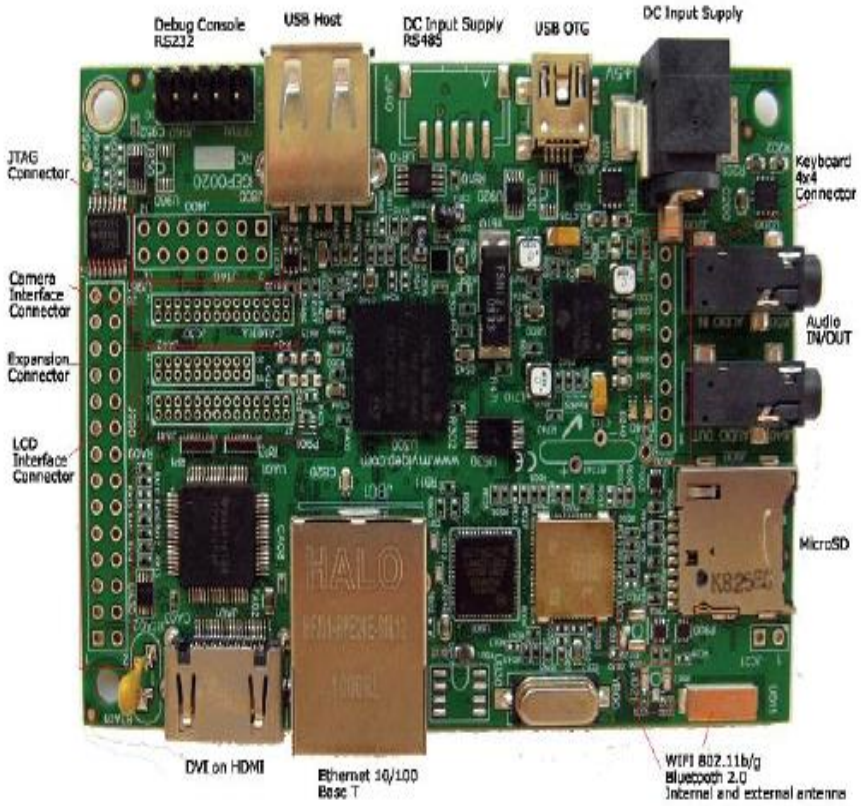
Create



Koala

Sistema embebido IGEPv2

- Arquitectura TI OMAP3530.
- CPU ARM CORTEX A8 core a 720 MHz
- DSP TMS320C64x+
- RAM 512MB
- Flash 512MB NAND
- Ethernet 10/100 Mb BaseT
- Wi-Fi IEEE 802.11b/g
- Bluetooth 2.0
- Puerto USB 2.0
- Puerto DVI-D / HDMI
- Entrada de audio y salida estéreo
- Puerto MicroSD / MicroSDHC
- Etc…



OMNIA i900 de SAMSUNG

- Sistema operativo: Windows Mobile 6.1.
- Pantalla táctil LCD TFT de 3.2" WQVGA.
- 16 Gb. de memoria interna.
- Conexiones: Bluetooth, WiFi y USB.
- Acelerómetros para obtener la orientación del móvil.
- etc…

Bluetooth Adapter Module (BAM)

- Antena bluetooth que se conecta en el “cargo bay connector” del robot Create.
- Configuración del puerto serie (virtual):
  - Baudios: 57600 BPS.
  - Bits de datos: 8.
  - Paridad: No.
  - Control de flujo: No.

PC cliente

- Recoge y procesa las imágenes capturadas por la cámara cenital. Solicita la realización del trabajo a los dos servidores (Create y Koala)

  - Cámara Web 4x3x4 cm.
  - Sensor de imagen CCD1/4”
  - Velocidad de 30 fps.
  - Iris del objetivo F/2.0
  - Margen mínimo de enfoque de 15 cm.

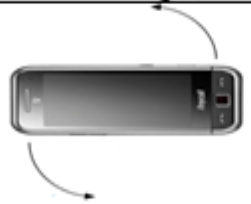
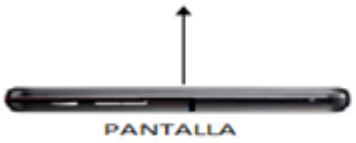
Control del robot CREATE

- Dos métodos:
  - **Control con una comunicación inalámbrica** (Bluetooth. Robot sin autonomía):

  - **Control con un sistema embebido**(robot autónomo):

Control por Bluetooth (I)

- Dirigir al CREATE con los acelerómetros.

Posición del móvil	Valor Y	Valor Z
	0 < Y < 1.1	--
	-1.1 < Y < 0	--
	--	0 < Z < 1.1
	--	-1.1 < Z < 0

Control por Bluetooth (II)

- Acción de control:
  - Vel_Izq = (ACELERACION * Z) + (RADIO * Y)
  - Vel_Der = (ACELERACION * Z) – (RADIO * Y)

Y	Z	Vel_Izq (mm/s)	Vel_Der (mm/s)	Comportamiento
-1	0	-200	200	El robot gira sobre sí mismo hacia la izquierda.
1	0	200	-200	El robot gira sobre sí mismo hacia la derecha.
0	-1	-300	-300	El robot retrocede hacia atrás a 300 mm/s de forma lineal.
0	1	300	300	El robot avanza a 300 mm/s de forma lineal.
0.5	0.5	250	50	El robot avanza girando hacia la derecha
0.5	-0.5	-50	-250	El robot retrocede girando hacia la izquierda
-0.5	-0.5	-250	-50	El robot retrocede girando hacia la derecha
-0.5	0.5	50	250	El robot avanza girando hacia la izquierda
0	0	0	0	El robot no se mueve

Control por Bluetooth (III)

- Video ilustrativo…

Control con el sistema IGEPv2

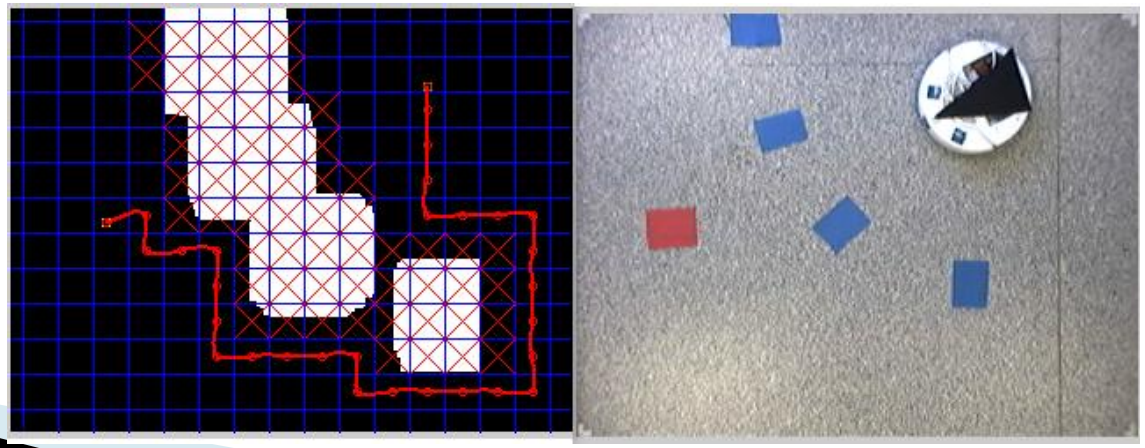
- Coordinación de robots: Tarea de limpieza.
  - 1. El PC solicita a los servidores la realización de la tarea conjunta. Captura la imagen para obtener las posiciones iniciales de los robots y la posición final, que es la pelota.
  - 2. El servidor Koala se sitúa detrás de la pelota. Cuando está preparado se lo comunica al PC cliente. Esta comunicación es asíncrona.
  - 3. El servidor Create se sitúa delante de la pelota. Cuando está preparado se lo comunica al PC cliente. Esta comunicación es asíncrona.
  - 4. Cuando los dos robots ya están en sus posiciones, el PC cliente le indica al Create que empuje la pelota.
  - 5. Finalmente, los dos robots vuelven a su posición de partida, llevando consigo (el Koala) la pelota.

Generación de trayectorias (I)

- Es necesario emplear algún tipo de algoritmo que encuentre la trayectoria libre de colisiones desde el robot al destino:
  - **Descomposición en celdas.** Se trata de descomponer el espacio en regiones simples llamadas celdas y encontrar una ruta sin obstáculos. Las celdas pueden ser:
    - **EMPTY.** Si su interior no intersecta con ningún obstáculo.
    - **FULL.** Si está completamente incluida en el interior de un obstáculo.
    - **MIXED.** Ninguna de las dos situaciones anteriores. Contiene un espacio EMPTY y otro espacio FULL.

Generación de trayectorias (II)

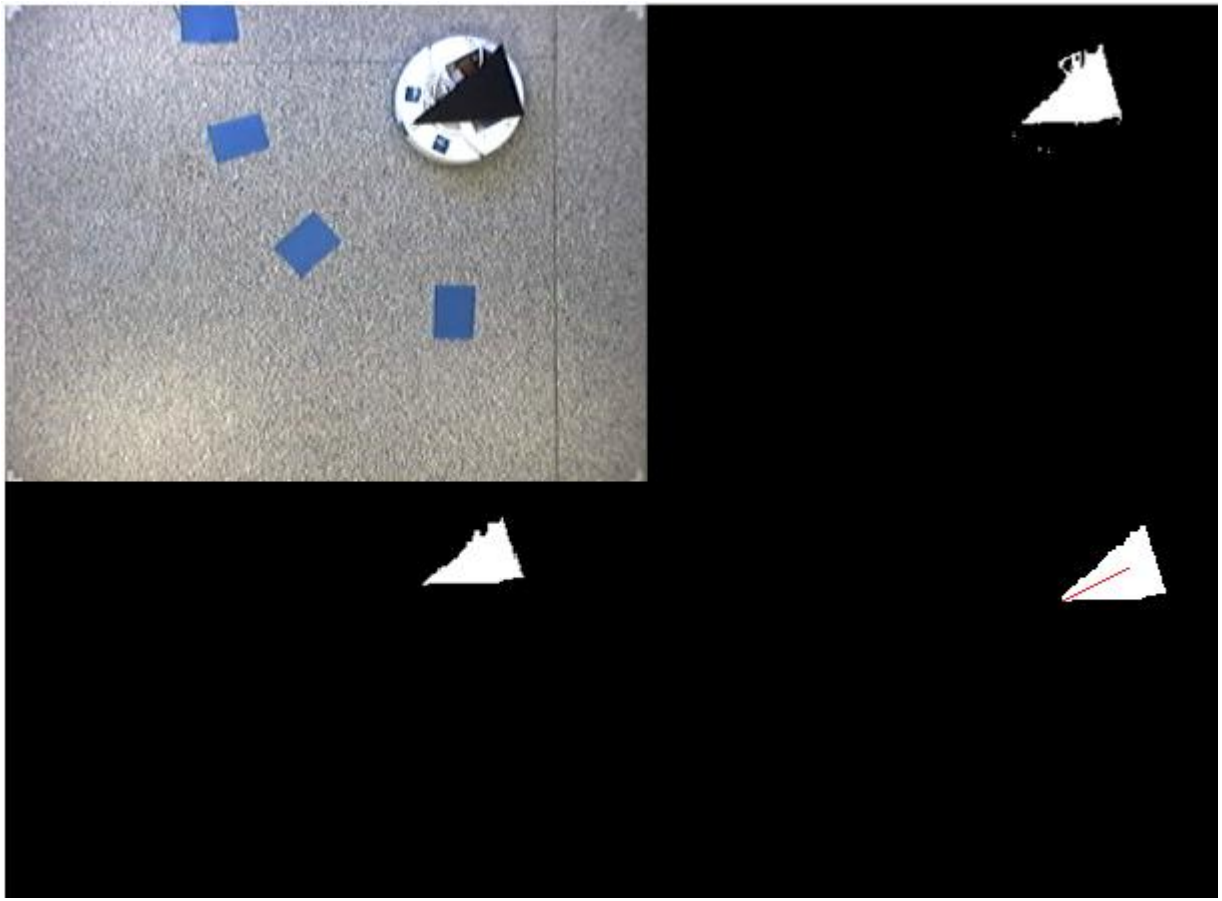
- Tamaño de la celda (**RASTER**) crítico:
  - Tamaños pequeños pueden producir la colisión con el obstáculo.
  - Tamaños grandes pueden no encontrar camino aún existiendo este.
  - Añadir el tamaño del robot a los obstáculos y considerar éste como un punto → RASTER pequeño.



Procesamiento de imágenes (I)

- Se dispone de la cámara cenital para capturar las imágenes.
  - Segmentación de la imagen para obtener las posiciones iniciales de los robots, así como su orientación, y la posición final.
  - La segmentación se realiza en base a los componentes RGB de la imagen.
  - Necesario realizar tratamientos de la imagen: erosión y dilatación

Procesamiento de imágenes (II)



Control de trayectorias (I)

- **Control por punto descentralizado**. Establecer el control a partir de la posición y velocidad de un punto que está separado una distancia e del eje de tracción del robot.

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x_m + e\cos\theta \\ y_m + e\sin\theta \end{bmatrix}$$

- Cálculo de posiciones

$$X_m = X_m + (\text{Distancia} * \cos(\theta)$$

$$y_m = y_m + (\text{Distancia} * \sin(\theta)$$

$$\theta = \theta + \frac{v_d - v_i}{2b} * T$$

Control de trayectorias (II)

- Cálculo de velocidades

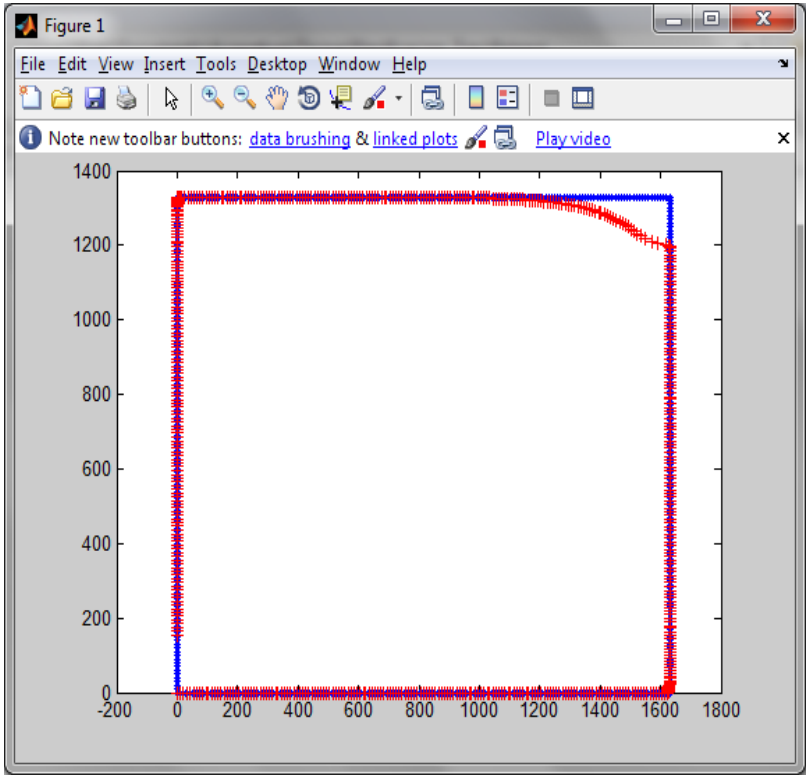
$$\begin{bmatrix} v_i \\ v_d \end{bmatrix} = \frac{1}{e} \begin{bmatrix} e\cos(\theta) + b\sin(\theta) & e\sin(\theta) - b\cos(\theta) \\ e\cos(\theta) - b\sin(\theta) & e\sin(\theta) + b\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix}$$

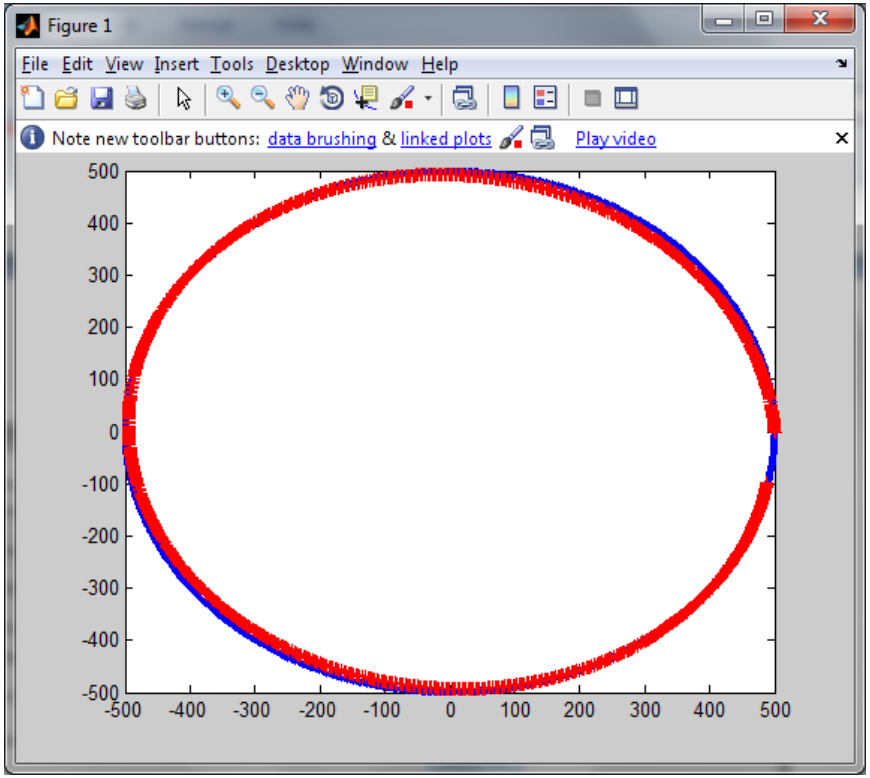
- Ley de acción cinemática

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} k_{vx}\dot{x}_{ref} \\ k_{vy}\dot{y}_{ref} \end{bmatrix} + \begin{bmatrix} k_{px} & 0 \\ 0 & k_{py} \end{bmatrix} \begin{bmatrix} x_{ref} - (x_m + e\cos(\theta)) \\ y_{ref} - (y_m + e\sin(\theta)) \end{bmatrix}$$

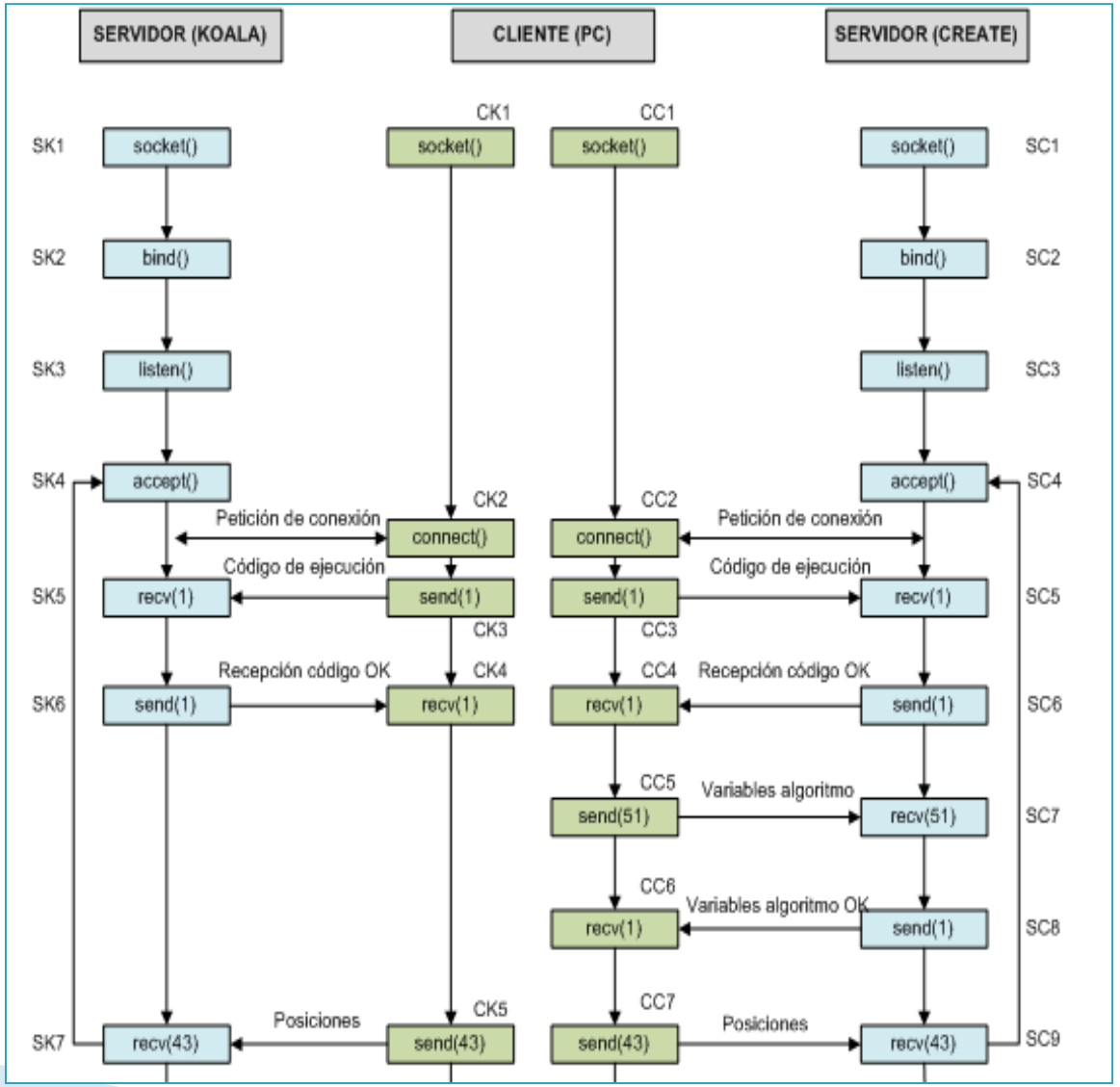
Control de trayectorias (III)

- Trayectoria de referencia VS trayectoria real

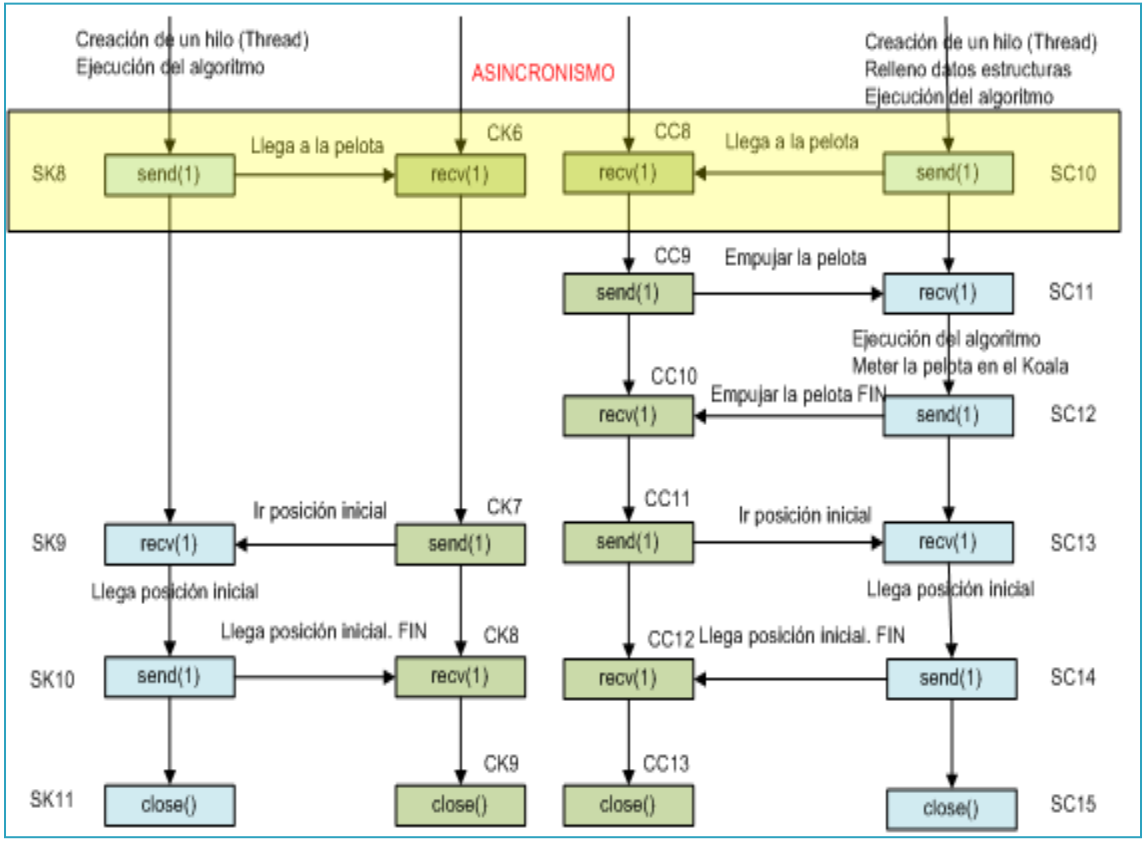




Esquema de Comunicación (I)



Esquema de Comunicación (II)



Resultados

- Resultado de la tarea coordinada (video ilustrativo donde se ve a los robots trabajar. Al mismo tiempo se pone el video de lo que sucede en el PC…)

CONCLUSIONES

- De una forma sencilla se ha podido comprobar la comunicación asíncrona con diferentes dispositivos en una arquitectura cliente-servidor.
- Estos experimentos se pueden extrapolar a cualquier entorno y sirven de base para realizar tareas más complejas.
- El procesamiento de las imágenes con MATLAB y OPENCV proporciona buenos resultados.
- Algoritmos de seguimientos de trayectorias y control de caminos relativamente sencillos, aunque sea difícil encontrar los valores de las ganancias que obtienen un funcionamiento óptimo de éstos.
- Complejidad del trabajo reside en las tareas adicionales, como son el procesamiento de las imágenes, generación de trayectorias libre de obstáculos, creación de una librería, algoritmos de control de trayectorias, etc.

TRABAJOS FUTUROS

- Ampliar la arquitectura diseñada, añadiendo nuevos dispositivos, ya sean robots móviles o cualquier otro tipo de robot.
- Disponer PCs clientes en un lugar físico diferentes al actual. VPN
- Ampliar autonomía del robot Create:
  - Incorporar una cámara Web y utilizar el DSP de IGEPv2.
  - Incorporar brazo robótico.

ARTICULO y REFERENCIAS

- **El trabajo actual ha sido aceptado en las Jornadas de Automática 2010 de Jaén.**

- iRobot: Educational & Research Robots. Disponible en http://store.irobot.com/shop/ index.jsp?categoryId=3311368.
- ISEE – IGEP Platform Homepage. Disponible en: http://www.igep-platform.com/.
- The open source, cross platform Code::Blocks. Disponible en: http://www.codeblocks.org/.
- Centro de desarrollo de Microsoft para consultas de “Microsoft Visual Studio 2008”. Disponible en http://msdn.microsoft.com/es-es/default.aspx.
- Mathworks. Disponible en http://www.mathworks.com/.
- OpenCV. Disponible en http://opencv.willowgarage.com/documentation/c/index.html.