Programación de un robot industrial Scara mediante un dispositivo móvil

2 de septiembre de 2012

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Alumno: Ángel Fernández - Cañada Vilata

Director: Martín Mellado Arteche

Índice

Índice de figuras

1. Agradecimientos

Agradezco la oportunidad, la ayuda y el apoyo que me han dado el Dr. Martín Mellado Arteche de la UPV y Víctor Manuel Peiró Torres Director de la zona Levante de la empresa KUKA Robots Ibérica, S.A, así como al resto de miembros de su equipo, sin olvidar a mi familia y a todos los demás que han estado a mi lado durante este largo año. Muchas gracias a todos.

2. Introducción

En la actualidad vivimos una revolución tecnológica ocasionada por la aparición de nuevos dispositivos móviles de gran potencia. Es inevitable que nos veamos tentados de combinar este mundo con el otro campo y futuro tecnológico conocido como la robótica.

Con la versatilidad y potencia de los nuevos dispositivos móviles podemos agilizar y poner al alcance de cualquier persona el manejo remoto de un robot industrial. Gracias a las pantallas táctiles las órdenes pueden ser introducidas por el usuario de forma fácil y intuitiva.

2.1. KUKA ROBOTS IBÉRICA S.A.

KUKA Robots Ibérica, S.A. es miembro del Grupo KUKA, una corporación internacional centrada en gran medida en la industria de bienes de consumo y en el mercado del automóvil. Su abanico de servicios va desde el desarrollo de la Ingeniería para el producto final y la producción de sistemas para su realización, hasta la formación de los trabajadores y soporte para el mantenimiento y modernización de las líneas de producción.

2.2. Android

Android es un sistema operativo para móviles. Fue prácticamente desconocido hasta que en 2005 Google lo compró. Hasta noviembre de 2007 sólo hubo rumores, pero en esa fecha nació la Open Handset Alliance, que agrupaba a muchos fabricantes de teléfonos móviles, chip-sets y Google. Es en ese año cuando se proporcionó la primera versión de Android, junto con el SDK para que los programadores empezaran a crear sus aplicaciones para este sistema. Sus inicios fueron un poco lentos, debido a que se lanzó antes el sistema operativo que el primer móvil compatible. No obstante, rápidamente se ha colocado como el sistema operativo de móviles más vendido del mundo.

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma. El sistema permite programar aplicaciones en una variación de Java llamada Dalvik. El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, el bluetooth, la cámara, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java.

Características importantes para el proyecto:

- La plataforma es adaptable a pantallas más grandes, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0 y Android.
- Android soporta las siguientes tecnologías de conectividad: GSM/EDGE, IDEN CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+ y WiMAX.
- Soporte para hardware adicional. Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, gamepad, termómetro, aceleración por GPU 2D y 3D.
- Android tiene soporte nativo para pantallas capacitivas con soporte multitáctil.
- Multitarea real de aplicaciones está disponible, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj, a diferencia de otros sistemas de la competencia en la que la multitarea es congelada.
- Una de las mejores características de este sistema operativo es que es completamente libre. Es decir, ni para programar en este sistema ni para incluirlo en un teléfono hay que pagar nada.

2.3. Realidad aumentada

La realidad aumentada es el término que se usa para definir una visión directa o indirecta de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real. Consiste en un conjunto de dispositivos que añaden información virtual a la información física ya existente, es decir, añadir una parte sintética virtual a lo real. Los componentes que forman el área de la realidad aumentada son los siguientes:

- Monitor del computador o display: instrumento donde se ve reflejado la suma de lo real y lo virtual que conforman la realidad aumentada.
- Cámara u otros sensores: dispositivo que toma la información del mundo real y la transmite al software de realidad aumentada.
- Software: programa que toma los datos reales y los transforma en realidad aumentada.
- Marcadores: los marcadores básicamente son hojas de papel con símbolos que el software interpreta y de acuerdo a un marcador especifico realiza una respuesta específica (mostrar una imagen 3D, hacer cambios de movimiento al objeto creado...)

En nuestro proyecto como ya comentaremos más adelante utilizamos marcadores para mostrar información sobre una caja 3D. En nuestro caso, a modo de ejemplo hemos impreso los logos de KUKA y el instituto Ai2 tal y como se puede ver en las figuras(7) y (8)

2.4. Bluetooth

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) con un enlace por radiofrecuencia en la banda de los 2,4 GHz. Permite comunicaciones, incluso a través de obstáculos, a distancias de hasta unos 10 metros. Los principales objetivos que se pretenden conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Bluetooth se utiliza principalmente en un gran número de productos tales como teléfonos, impresoras, módems y auriculares. Su uso es adecuado cuando puede haber dos o más dispositivos en un área reducida sin grandes necesidades de ancho de banda. Su uso más común está integrado en teléfonos y PDA, bien por medio de unos auriculares Bluetooth o en transferencia de ficheros.

Bluetooth tiene la ventaja de simplificar el descubrimiento y configuración de los dispositivos, ya que éstos pueden indicar a otros los servicios que ofrecen, lo que redunda en la accesibilidad de los mismos sin un control explícito de direcciones de red, permisos y otros aspectos típicos de redes tradicionales.

En nuestro proyecto decidimos utilizar bluetooth ya que queríamos limitar la conexión a una proporción 1:1 entre el robot y un dispositivo móvil. Otro factor importante es el alcance del bluetooth (más o menos 10 metros). Por temas de seguridad consideramos que el usuario debe de estar presente cuando el robot se mueva. También hay que destacar que las órdenes enviadas están en formato XML, lo que supone poco peso, por lo que la velocidad de trasmisión del bluetooth es suficiente. Y por último, como se ha comentado antes, el protocolo ofrece grandes facilidades para gestionar las conexiones.

3. Requisitos

A continuación se describen los requisitos de la aplicación propuestos por la empresa KUKA Robots Ibérica, S.A.. Estos los podemos dividir en:

- Requisitos funcionales:
  - Conexión a distancia de un dispositivo Android con el robot scara de KUKA.
  - Mover el robot por articulaciones.
  - Mover el robot indicando coordenadas cartesianas.
  - Mostrar logo de la empresa mediante realidad aumentada.
  - Mostrar información del progreso del estado del robot.
  - Permitir el uso de la aplicación con varios modelos de robot scara de KUKA.
  - Permitir interrumpir el uso de la aplicación caso de emergencia.
  - Reproductor capaz de grabar y reproducir órdenes.
- Requisitos de interfaz:
  - Introducir las órdenes mediante botones en pantalla.
  - Introducir órdenes utilizando con otras posibilidades que permita Android.

3.1. Requisitos funcionales:

3.1.1. Conexión a distancia de un dispositivo Android con el robot scara.

El usuario debe de ser capaz de conectarse con un robot scara de KUKA mediante un dispositivo Android y bluetooth. Como datos de entrada el usuario especificará un dispositivo bluetooth al que quiera conectarse. Cuando se encuentre el deseado, el usuario podrá confirmarlo. Una vez hecho esto se habrá conseguido la conexión con el robot. No se debe permitir conectar varios dispositivos con el servidor a la vez

3.1.2. Mover el robot por articulaciones.

El robot scara tiene cuatro articulaciones. El usuario debe poder especificar una orden mediante la cual se mueva cualquiera de las articulaciones por separado. El proceso será el siguiente. El usuario indica que articulación desea mover e indica cuanto desea que se mueva y a que velocidad. Como salida la aplicación debe generar una orden que mueva el robot.

3.1.3. Mover el robot indicando coordenadas cartesianas.

El robot scara debe poder moverse en las coordenadas del mundo real. El origen de coordenadas se encuentra en la base del robot. El usuario debe poder especificar una orden mediante la cual se mueva el robot en las coordenadas x,y,z y a una posición angular alpha. Se podrá indicar también una velocidad para el movimiento. El proceso será el siguiente, primero el usuario indicará que eje desea mover y cuanto desea que se mueva. Como salida la aplicación generará una orden que mueva el robot

3.1.4. Reproductor capaz de grabar y reproducir órdenes.

El usuario podrá enseñar al robot una serie de órdenes introducidas mediante los modos antes nombrados. Una vez introducidas todas las órdenes deseadas el robot las podrá reproducir cíclicamente cuando el usuario lo requiera. Las órdenes posibles del reproductor serán reproducir, grabar o parar.

3.1.5. Mostrar logo de la empresa mediante realidad aumentada.

Debe existir un modo en la aplicación en el que se utilice una cámara. Cuando dicha cámara enfoque un marcador se mostrará el logo de la empresa.

3.1.6. Mostrar información del progreso del estado del robot.

En la aplicación se debe poder mostrar al usuario cual es la posición actual del robot. Se debe mostrar la posición de las articulaciones, la posición en coordenadas cartesianas, el estado de la herramienta y la velocidad.

3.1.7. Permitir el uso de la aplicación con varios modelos de robot scara.

La aplicación debe poder ser utilizada con varios modelos de robot scara disponibles. Antes de empezar a enviar órdenes el usuario debe poder elegir con que robot va a trabajar. La diferencia entre un robot y otro afectará en la aplicación al rango máximo de movimiento del robot en cualquiera de sus modos.

3.1.8. Permitir interrumpir el uso de la aplicación caso de emergencia.

En caso de emergencia el usuario podrá parar la ejecución de la aplicación. En el modo parada de emergencia el usuario podrá seguir

consultando la posición del robot. Se deberá permitir retomar la ejecución. Esta funcionalidad deberá aparecer en todas las pantallas de movimiento y en la principal.

3.2. Requisitos de interfaz:

3.2.1. Introducir las órdenes mediante botones en pantalla.

El usuario debe poder introducir las órdenes de forma análoga a como lo haría en una aplicación de escritorio. Las interfaces deben de ser similares y no causar sorpresa.

3.2.2. Introducir órdenes utilizando con otras posibilidades que permita Android.

El usuario debe poder introducir las órdenes aprovechando las posibilidades que ofrece la plataforma. Una posibilidad puede ser introducir órdenes en la pantalla mediante gestos, agitar el dispositivo, etc... No importa que la interfaz de esta funcionalidad sea diferente si es necesario para su propósito.

4. Propuesta

4.1. Consideraciones iniciales

Para que la aplicación funcione debidamente hay que tener en cuenta las siguientes consideraciones:

- Para poder ejecutar las aplicaciones debemos tener acceso al panel de control del robot scara ya sea en modo operario o experto. No obstante, se recomienda el acceso en modo experto para poder inspeccionar mejor el procedimiento de conexión.
- El pc servidor y el robot deben de estar conectados a un red común.
- Se debe tener instalado en el pc servidor PC y en el pc del robot el programa KUKA Router configurado según la red a la que estén conectado.
- El pc servidor debe tener un dispositivo bluetooth visible y el programa bluesoleil instalado.
- El dispositivo android debe tener bluetooth. Las versiones de android probadas van desde la 2.2 hasta la 3.2. aunque posiblemente funcione en versiones posteriores.

4.2. Presentación de la propuesta

Ante los requisitos anteriormente planteados y las consideraciones oportunas realizadas describo en este punto mi propuesta. La podemos dividir en tres apartados:

- Robot
- Servidor PC
- Cliente Android

La figura (1) sorbe para ilustrar la siguiente descripción.

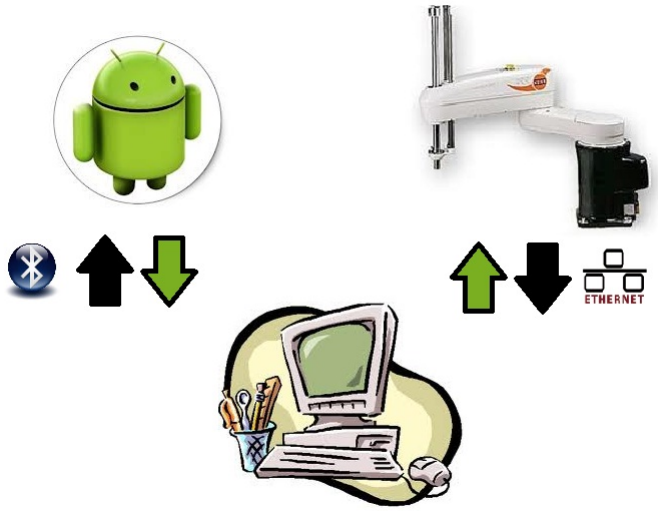

Figura 1: Esquema de comunicación

4.2.1. Robot

1. Función

   El robot es el encargado de realizar las acciones indicadas por el usuario. Recibe por la red las órdenes del Servidor PC y le envía información sobre el estado actual del robot. Lo podemos dividir en dos partes: lo que es el robot propiamente dicho con sistema operativo VxWorks y el PC del robot con sistema operativo Windows XP.

2. VxWorks

   Es el sistema operativo del robot y el encargado de gestionar sus recursos. Se comunica con el PC del robot mediante el programa KUKA Router. En la figura (6) se puede ver un esquema de la conexión.

4.2.2. Servidor PC

1. Función

   Recibe las ordenes del dispositivo móvil mediante una conexión bluetooth. Envía las ordenes procesadas al robot. Es el verdadero artífice la funcionalidad de reproducción y grabación. También se encarga de gestionar prioridades.

2. Diagrama de clases

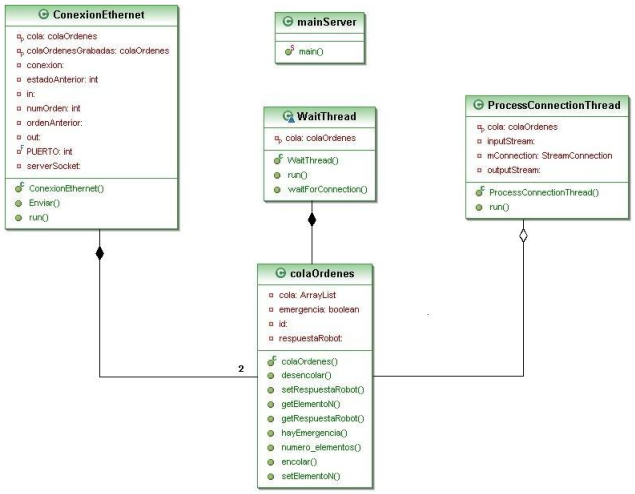

Figura 2: Diagrama de clases Servidor PC

WaitThread: es la clase encargada de iniciar la conexión el robot a través de la red y esperar las conexiones vía bluetooth. Crea la cola de órdenes de entrada la cual será común a la clase ConexionEthernet y ProcessConnectionThread.

ProcessConnectionThread: es la clase encargada de manejar la conexión bluetooth, enviar las respuestas y recibir las órdenes.

ConexionEthernet: es la clase encargada de enviar las órdenes, formatearlas, en algún caso añadir órdenes de control y realizar las acciones necesarias para la grabación y reproducción de órdenes.

colaOrdenes: es una clase encargada de guardar las órdenes.

4.2.3. Cliente Android

1. Función

   Es el que más carga de trabajo tiene ya que es el que interactúa directamente con el usuario. Sus funciones son:

   - la introducción de datos mediante teclado y pantalla con todos sus modos (por articulaciones y ejes cartesianos).
   - la gestión de los diferentes tipos de robot Scara e idiomas de la aplicación.

- la comunicación bluetooth.
- mostrar el estado del robot.
- la aplicación de realidad aumentada.
- gestionar la interfaz de la reproducción y la grabación.

2. Diagrama de clases

   El diagrama muestra las clases más importantes del proyecto Android. Se han obviado métodos y las clases que representan a las interfaces de la aplicación con el objetivo de mejorar su legibilidad.

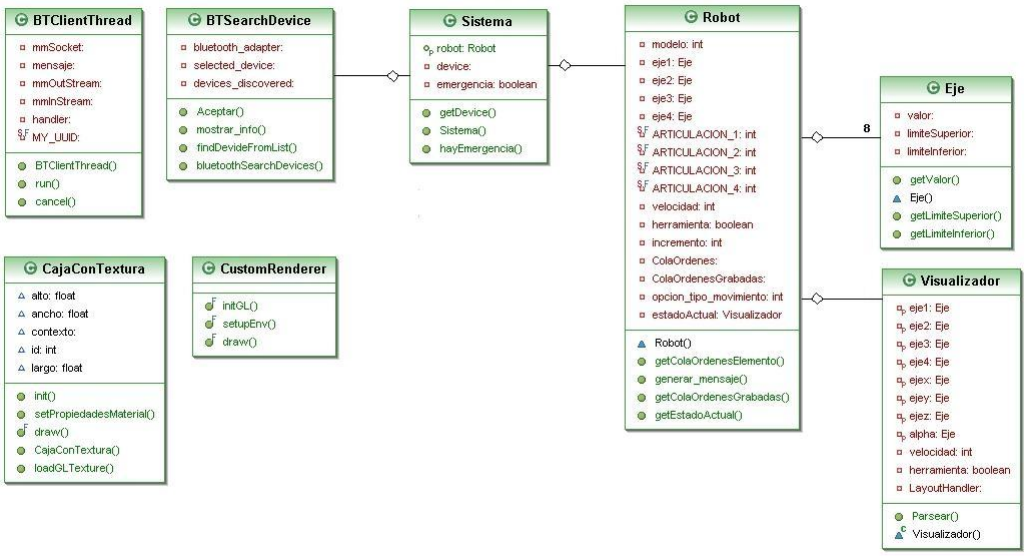

Figura 3: Diagrama de clases de la aplicación Android

Sistema: es la clase principal contiene el robot , el estado del sistema y los parámetros de la conexión bluetooth.

BTSearchDevice: es la clase encargada de detectar dispositivos bluetooth.

BTClientThread: esta clase es la encargada de realizar la conexión bluetooth al dispositivo indicado por el sistema, enviar las órdenes y recibir las respuestas.

Eje: representa los ejes por los que se mueve el robot.

Robot: Contiene todas las características del robot (modelo,estado de la herramienta,velocidad,....), guarda el estado actual, es el encargado de gestionar las órdenes hasta su envío y una lista de órdenes grabadas.

Visualizador: esta clase representa el estado actual y es uno de los atributos de la clase robot antes vista.

CajaConTextura: es una clase del tipo ARObject de realidad aumentada. Es la representación gráfica de una caja con una textura como tapadera. Esta clase es la que nos permite dibujar letreros de realidad aumentada. Véanse la figura(8) y la figura(7).

CustomRenderer: es la clase donde se definen las características de la visualización en OpenGL.

4.3. Hardware utilizado

A continuación presento el hardware utilizado en el proyecto. Explicaré para qué se utilizan cada uno de los elementos así como sus características.

4.3.1. Robot Scara de KUKA

El robot con el que se trabaja es un robot SCARA KR 5 R350 proporcionado por la empresa KUKA. Sin embargo, la aplicación está programada para que se pueda trabajar con el resto de modelos de robot SCARA de la marca. El robot se utiliza para llevar a cabo todas las órdenes del usuario.

En general, un robot SCARA (Selective Compliant Articulated Robot Arm) es un brazo articulado con tres ejes angulares (dos rotacionales y uno traslacional) y uno prismático. En total dispone de cuatro ejes que permite movimientos en los ejes X-Y-Z. Tiene un alcance de 350 mm con una capacidad de carga de 5 kg y una alta exactitud de posicionamiento de menos de 0,02 mm. El robot mismo necesita solamente una superficie de emplazamiento de pequeñas dimensiones (aprox. 150 x 150 mm) y pesa aproximadamente 20 kg.

Son ideales siempre que se requiera gran precisión y velocidad. Consiguen efectuar con gran precisión las más complicadas tareas en espacios reducidos siendo al mismo tiempo los más rápidos de su clase. Por ello, representan la opción más acertada para tratar componentes pequeños de forma precisa y rápida. Como se observa en la figura(4) el robot se compone de:

- Robot SCARA de KUKA: es el brazo robot que realizará las órdenes recibidas por la unidad de control o por el panel de control.

Figura 4: Robot SCARA de KUKA, unidad de control y panel de control

- Unidad de control (PC del robot): funciona como un servidor, se trata de una cpu que recibe las órdenes de usuario, las transforma en órdenes del robot y las envía al robot.
- Panel de control: va conectado a la unidad de control. Permite introducir la línea de comandos para las tareas del robot así como cargar y guardar programas. En el proyecto no se empleará este elemento.

4.3.2. PC con conectividad bluetooth

En nuestro caso es un ordenador con un dispositivo bluetooth y una tarjeta de red. Como he comentado antes en el punto 4.2.2 , el servidor PC se encarga principalmente de recibir las órdenes del dispositivo móvil, "desgranar" la información y traducirla en un formato adecuado para su posterior envío a la unidad de control (pc del robot). También es capaz de gestionar prioridades y realizar algunas funciones más complejas como son la reproducción y grabación de órdenes.

4.3.3. Tablet ACER Iconia A500

La tableta ACER ICONIA TAB A500 es el dispositivo móvil táctil que utilizaremos preferentemente. Entre sus características podemos destacar: su velocidad de procesador de 1GHz, memoria RAM DDR2 de 1GB, pantalla de 10 pulgadas con una resolución de 1280x800, cámara de alta definición de 1080p, Bluetooth y una versión de Android 3.0.

También se ha probado la aplicación sobre un móvil LG Optimus M con android 2.2 y sobre un HTC EVO 3D con android 2.3 dotados también con bluetooth y cámara.

Figura 5: ACER ICONIA TAB A500

4.4. Software utilizado

A continuación se enumera el software utilizado en el desarrollo de este proyecto.

4.4.1. KRC editor

Es un editor de código desarrollado por KUKA. Se utiliza para implementar programas .krl (kuka robot language) los cuales pueden ser ejecutados en los robots de la marca. Carece de corrector lo que hace que los programas deban ser depurados en el propio robot.

En el proyecto se utiliza este editor para programar el bucle de ejecución de órdenes así como el envío y recepción de información en la parte del robot.

4.4.2. KUKA Router

Es un programa desarrollado por KUKA encargado en nuestro caso de la comunicación entre un ordenador externo y el PC del robot. Para que la comunicación sea satisfactoria se debe ejecutar en los dos PC's e indicar en el PC del robot un puerto y la ip de escucha. El diagrama de la figura (6)muestra la conexión entre el robot y una red con sensores, pc's e incluso otras redes.

4.4.3. Eclipse

Eclipse es una plataforma de desarrollo abierta. La Fundación Eclipse es una organización sin fines de lucro que aloja los proyectos de Eclipse y ayuda a la creación de una comunidad de código abierto y así como productos y servicios complementarios.

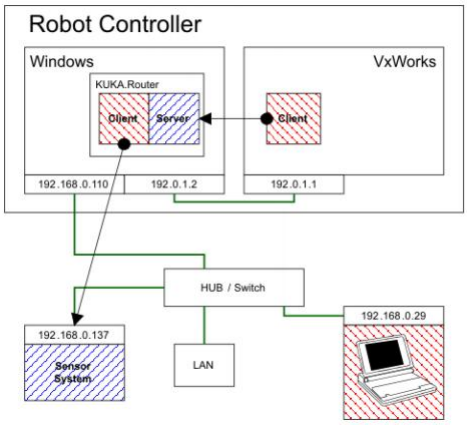

Figura 6: Conexión KUKA Router y una red de sensores

El Proyecto Eclipse fue creado originalmente por IBM en noviembre de 2001 y apoyado por un consorcio de proveedores de software. La Fundación Eclipse se creó en enero de 2004 como una organización independiente sin ánimo de lucro para actuar como administrador de la comunidad Eclipse.

Eclipse es el entorno de desarrollo en el que se llevará a cabo el proyecto, dado que esta plataforma nos da un buen soporte para el desarrollo de aplicaciones Android y Java.

4.4.4. JRE, JDK y JVM de Java

El JDK (Java Developer Kit) contiene herramientas necesarias para desarrollar los programas de Java. Entre las herramientas del JDK se incluye el compilador, el lanzador de aplicaciones Java, applets, etc ...

El JRE (Java Runtime Environment) es necesario para ejecutar programas desarrollados en Java. Contiene las bibliotecas de clases, otros archivos auxiliares y la JVM (Java Virtual Machine). La JVM permite independizar el código del programa de la plataforma de ejecución.

4.4.5. NDK Android

Android permite la utilización de código como OpenGL, C y C + + en sus aplicaciones. El NDK (Native Development Kit) es un conjunto de herramientas que nos permiten utilizar el código nativo de otros lenguajes.

En nuestro caso el NDK es fundamental pues necesitamos programar en OpenGL para la representación de formas en las interfaces con realidad aumentada.

4.4.6. Librería Bluecove 2.1

BlueCove es una librería de Java para la gestión de conexiones Bluetooth. Funciona en la mayor parte de los sistemas operativos pero nosotros en nuestro proyecto lo utilizaremos con Windows XP.

4.4.7. Aplicación Bluesoleil 2.3

BlueSoleil es una herramienta con la que puedes conectar un ordenador con diferentes tipos de dispositivos: cámaras de fotos, PDA's, teléfonos móviles,etc... BlueSoleil es capaz de identificar todos los dispositivos en la zona de cobertura. Es una herramienta con todas las opciones y funciones necesarias para controlar y gestionar las conexiones Bluetooth y es muy popular entre los fabricantes de hardware.

4.4.8. AndAR (Android Augmented Reality)

AndAR es un proyecto Open Source basado en ARToolKit que permite realizar aplicaciones de realidad aumentada en la plataforma Android. Nació como proyecto final de carrera de Tobias Domhan. Hoy en día esta hospedado por Google en `http://code.google.com/p/andar/`.

Nosotros utilizaremos AndAR para la representación de información en unas cajas 3D. En la figura (7) **se muestra una caja con el logotipo de la empresa KUKA Robots Ibérica, S.A. En la figura**(9) **se muestra el marcador utilizado.**

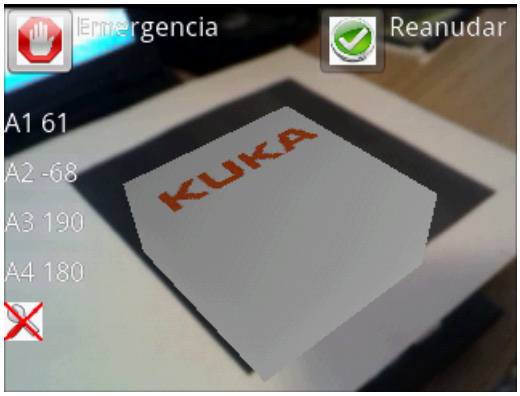

Figura 7: Logo de la empresa Kuka.

Figura 8: Logo del Instituto de Automática e Informática Industrial

Figura 9: Marcador de realidad aumentada.

5. Factores de interés en el desarrollo del proyecto

5.1. Conexión bluetooth

La conexión bluetooth es la existente entre el servidor pc y el dispositivo móvil. Para este cometido, utilizamos la librería bluecove 2.1 para la aplicación java y el programa de gestión de bluetooth Bluesoleil 2.3 en el PC Servidor. Sin embargo, no hace falta ninguna librería extra en Android ya que la plataforma nos da el soporte necesario.

Las clases Android que intervienen en el proceso son:

- BTSearchDevice es la actividad que se encarga de buscar los dispositivos para conectarnos. En la figura(10) podemos ver una imagen detallada de la interfaz.
- Sistema es una clase que guarda entre otras cosas los datos para la conexión bluetooth.
- Envíos es una tarea asíncrona encargada de lanzar los envíos.
- BTClientThread es un thread encargado de enviar los datos y recibir la respuesta del servidor pc.

En la parte del servidor pc tenemos:

- WaitThread es un thread que gestiona las conexiones. Por seguridad sólo admite una conexión. Cuando recibe una petición de conexión activa el thread de conexión.
- ProcessConnectionThread es el thread encargado de recibir las órdenes del dispositivo móvil y enviar la respuesta.

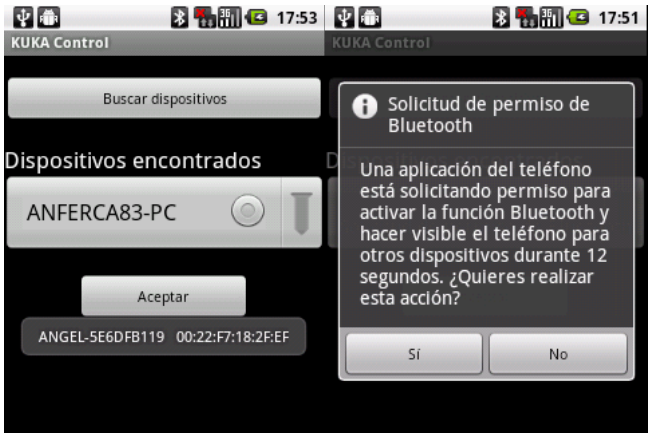

Figura 10: Interfaz de la conexión bluetooth

Para identificar la aplicación al hacer la conexión utilizamos un UUID en nuestro caso el 00001101000010008000000805F9B34FB. Las dos aplicaciones la de Android y la Java del Servidor PC la comparten. Hay que decir que la primera vez que se emparejen dos dispositivos se pedirá una clave. Esa clave se comparte entre los usuarios de los dispositivos y es un sistema de seguridad del sistema operativo ajeno al proyecto.

5.2. Movimientos del robot

```
<ExternalData>
        <Position>
                <A1>61</A1>
                <A2>-68</A2>
                <A3>190</A3>
                <A4>180</A4>
        </Position>

        <Ventosa>
                <Activada>false</Activada>
        </Ventosa>

        <Speed>
                <Valor>30</Valor>
        </Speed>

        <Movimiento>
                <Tipo>1</Tipo>
        </Movimiento>

        <PositionCar>
                <X>109</X>
                <Y>-67</Y>
                <Z>190</Z>
                <ALPHA>43</ALPHA>
        </PositionCar>

</ExternalData>
```

Figura 11: Formato de órdenes

Las órdenes que mandamos se corresponden con movimientos en el robot. Las órdenes se escriben en formato de fichero xml. Los ficheros son texto por lo que cuesta muy poco mandarlos y están estructurados de forma jerárquica por lo que la información es fácilmente encontrada. Por todo esto son un método idóneo de representación de órdenes. Un ejemplo de las órdenes que utilizamos es el de la figura(11).

- La etiqueta position engloba las posiciones de las articulaciones A1...A4
- La etiqueta ventosa nos indica si la herramienta está activada (true) o desactivada (false)
- La etiqueta speed nos indica el % de velocidad a la que se moverá el robot.
- La etiqueta position de la herramienta del robot en el espacio cartesiano que tiene como origen de coordenadas la base del robot.
- La etiqueta movimiento nos indica el tipo de movimiento 1 si es movimiento por articulaciones y 0 para movimiento por los ejes cartesianos.

Uno de los problemas más complejos del proyecto ha sido resolver el cambio de modo (articulaciones,coordenadas cartesianas) ya que ha supuesto la creación de toda la comunicación de vuelta entre el robot y el dispositivo móvil. A continuación paso a relatar los dos modos:

5.2.1. Por articulaciones del robot

El robot Scara posee 4 articulaciones. El movimiento de las articulaciones se ha limitado por seguridad de forma que la aplicación móvil controla los umbrales a los que llegará el robot. En la figura (12) se detalla cual es el movimiento de cada articulación.

- La articulación A1 se mueve circularmente, se puede pensar que mueve el hombro del robot.
- La articulación A2 se mueve de forma circular pero limitada,es decir no puede dar un giro completo. Se puede pensar en ella como en el codo del robot.
- La articulación A3 se mueve de arriba a abajo.
- La articulación A4 permite hacer movimientos circulares completos sobre la herramienta. Se puede pensar en ella como en la muñeca del robot.

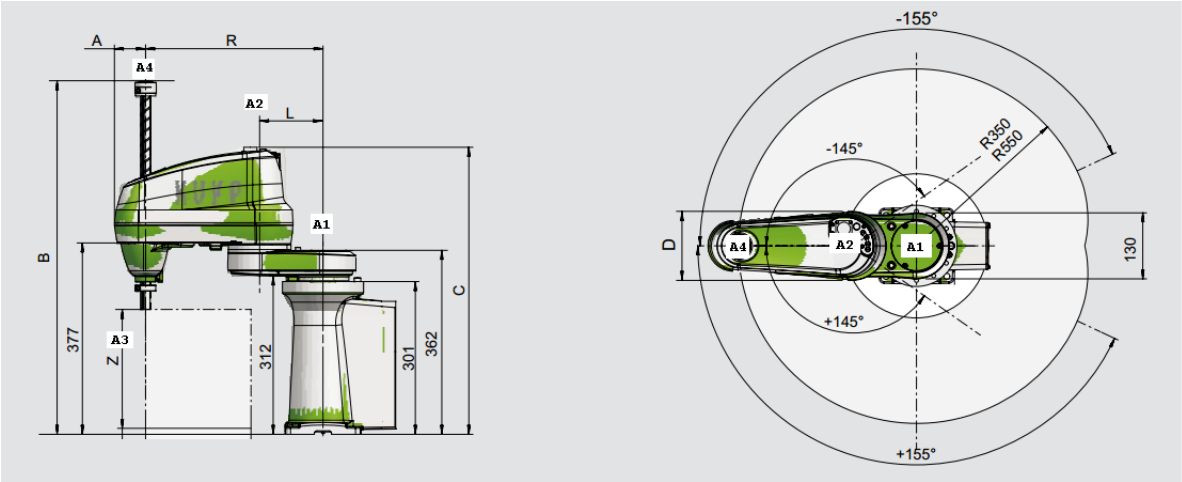

Figura 12: Articulaciones del robot

En el modo movimiento por articulaciones el usuario puede mover las articulaciones desde el modo teclado y táctil. Cuando el usuario introduce una orden esta se envía a la clase robot la cual la pondrá en la lista de órdenes a cumplir. Dichas órdenes se envían poco a poco al robot el cual las va cumpliendo.

5.2.2. Por ejes cartesianos

En el modo de movimiento por ejes cartesianos, el origen de coordenadas está en la base del robot. Con estas coordenadas representamos la posición de la herramienta en el área de trabajo. Este modo de movimiento sólo está disponible en el modo teclado por la complejidad de encontrar movimientos gestuales que fueran representativos de este tipo referencia. En la figura(13) se ilustra la representación del sistema.

Para terminar este apartado hay que hablar de la actividad Opciones. Dicha actividad permite al usuario configurar algunas características. Concretamente, se utiliza para:

- Configurar la velocidad a la que opera el robot.
- El tamaño de los incrementos del movimiento del robot en los modos táctil y teclado.

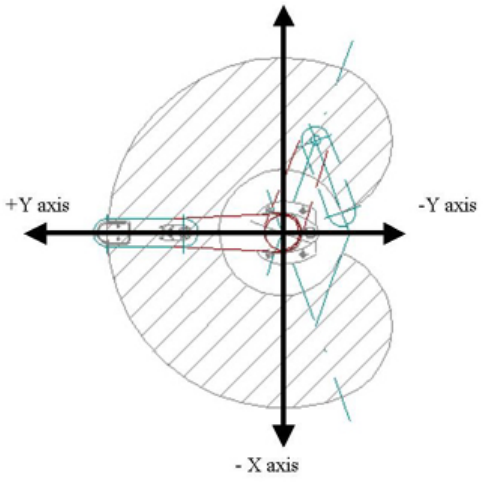

Figura 13: Área de trabajo del robot

- Permite eliminar los textos inscritos en pantalla de la aplicación táctil. Atención, por seguridad si se desactivan los textos de pantalla en la aplicación táctil tan sólo permanecerá activa la funcionalidad de realidad aumentada.
- Elegir el tipo de movimiento en el modo teclado.
- Elegir el tipo de robot scara.

Figura 14: Interfaz de la pantalla de opciones

5.3. Interacción con el usuario

La interacción con el usuario es una de las cosas más importantes de la aplicación ya que si la experiencia de usuario es buena conseguiremos que sea popular y realmente útil.

La interacción en nuestro proyecto se puede hacer de dos maneras:

- Modo teclado.
- Modo táctil.

5.3.1. Modo teclado

En este modo el usuario introduce las órdenes mediante un teclado dibujado en la pantalla. Es una forma rápida y fácil de introducir órdenes.En la figura(15) se muestra la interfaz de teclado para el movimiento por coordenadas cartesianas.

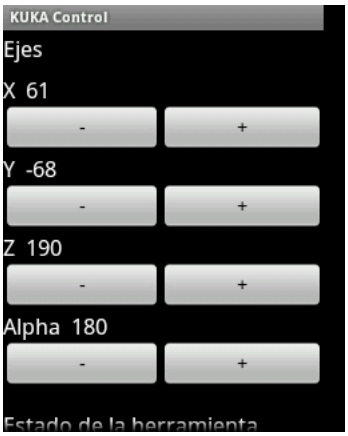

Figura 15: Interfaz modo teclado coordenadas cartesianas

En este modo interviene las actividades de las interfaces que son las encargadas de recoger los datos. Hay que recordar que la interfaz cambiará dependiendo si estamos en el modo de movimiento por ejes cartesianos o articulaciones del robot. Estos datos son enviados a la clase Robot para que sean encoladas en las órdenes pendientes de enviar.

5.3.2. Modo táctil

El modo táctil es el más complejo pero también el más divertido de utilizar. En la pantalla se muestra la información sobre la última

orden dada. Las órdenes sobre el movimiento se introducen mediante gestos. En la figura(16) se muestran los diferentes gestos y en la figura(17) la interfaz del modo táctil. Por otra parte, en esta pantalla cabe destacar que el fondo corresponde a la imagen que nos proporciona la cámara. Durante este modo, si enfocamos cualquier marcador de realidad aumentada se nos mostrará la imagen correspondiente al marcador.

Esta interfaz sólo permite los movimientos de tipo articulación. Es la encargada de recoger los datos y mostrar las características de realidad aumentada. Los datos recogidos por la interfaz son enviados a la clase Robot para que sean encoladas en las órdenes pendientes de enviar.

Las órdenes gestuales son:

- Mover articulación A1. El movimiento curvo hacia la derecha aumenta el valor de la articulación. Si el movimiento es en el sentido contrario se disminuye el valor.
- Mover articulación A2. El movimiento necesario es una recta hacia la derecha para incrementar el valor. Si el movimiento es en el sentido contrario se disminuye el valor.
- Mover articulación A3. El movimiento necesario es una recta hacia la arriba para incrementar el valor. Si el movimiento es en el sentido contrario se disminuye el valor.
- Mover articulación A4. Para incrementar el valor se pulsa en la superficie de la derecha. Si queremos disminuir el valor pulsamos en la región de la izquierda.
- Activar/Desactivar herramienta. La herramienta se activa/desactiva al agitar el dispositivo.

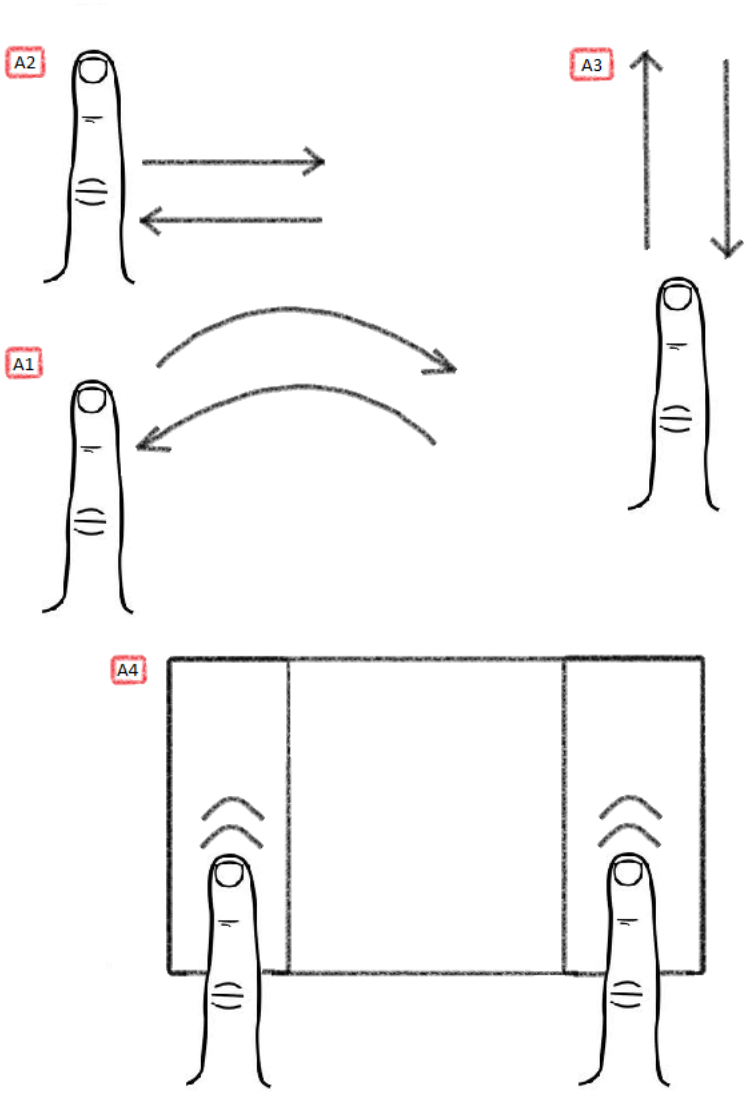

Figura 16: Ordenes gestuales

Figura 17: Interfaz modo táctil

5.4. Grabación y reproducción de órdenes

La funcionalidad del reproductor permite:

- Grabar órdenes
- Reproducir órdenes
- Listar órdenes grabadas

Cuando pulsamos la tecla grabar estamos indicando al programa que a partir de ese momento todas las órdenes que enviemos queremos que sean guardadas. Si introducimos cualquier orden en cualquiera de los modos, estas pasarán a ser reproducidas normalmente pero ahora también se guardarán en el servidor. Para parar de guardar imágenes pulsaremos en stop. Si queremos reproducirlas pulsaremos en reproducir. Cuando se pulsa en reproducir se envía una orden al servidor para que envíe las órdenes grabadas cíclicamente al robot. El robot cumplirá las órdenes hasta que se le indique que pare.

Hay varios puntos que hay que tener en cuenta en esta aplicación. El primero es que cuando se envían órdenes al servidor pc se gasta tiempo de comunicación. Es por ello que al reproducir veremos un aumento en la velocidad con la que el robot atiende las órdenes. Un segundo punto a tener en cuenta es que aveces se puede dar el caso de situaciones extremas en las que el robot para cumplir una orden tenga dos formas de llevarla a cabo. Para evitar estas situaciones de incertidumbre se añaden órdenes (puntos de control) que evitan este problema.

En esta funcionalidad interviene la clase Robot con la cola de órdenes grabadas y la interfaz de reproducción para indicar el tipo de orden (grabar,reproducir,parar). Intervienen el resto de interfaces de introducción de órdenes de usuario como las del modo teclado y las del

envío por bluetooth. No obstante el trabajo más pesado lo hace el servidor pc ya que es el que finalmente envía las órdenes al robot y gestiona las paradas, índica que orden es la siguiente a reproducir, introduce órdenes de control si es necesario...

Figura 18: Interfaz del reproductor.

Para terminar en la figura(18) se puede ver la sencilla interfaz del reproductor.

5.5. Parada de emergencia y recuperación

En todas las pantallas de introducción de órdenes, reproductor y menú principal se pueden observar dos iconos como los de la figura(19). El icono rojo indica la parada de emergencia y el verde la reanudación. Paso a describir cada una de las situaciones a continuación:

Emergencia: si pulsamos el botón rojo entraremos una situación de emergencia. Esto hará que el robot se pare y que no se pueda enviar ninguna nueva orden. Sí que se podrá navegar por las distintas interfaces para ver la situación del robot.

Reanudar: reactiva el robot cuando la situación de emergencia se ha solventado. Después de apretarlo toda la funcionalidad de la aplicación estará disponible.

Figura 19: Símbolos de parada de emergencia y reanudar.

En el tratamiento de las emergencias entran a trabajar tanto el pc servidor y la tarea asíncrona del dispositivo Android ya que son los encargados de enviar la orden de parada o reanudación y cortar o abrir la comunicación.

6. Propiedades Android del proyecto a resaltar

6.1. Tareas asíncronas en Android

6.1.1. Definición

La clase AsyncTask nos ayuda a los desarrolladores a ejecutar operaciones en segundo plano y definir los resultados en la interfaz de usuario. Para utilizar AsyncTask debemos crear una subclase de AsyncTask, comúnmente como una clase interna privada dentro de la actividad en la que estemos trabajando. Luego, sobrescribir uno o más métodos de AsyncTask para poder realizar el trabajo en segundo plano. Para iniciar la tarea asíncrona debemos llamar al método execute(). Existen varios métodos en una tarea asíncrona, de entre ellos cabe destacar:

- onPreExceute(). Se invoca inmediatamente después de que la tarea es ejecutada. Este paso se utiliza normalmente para configurar la tarea.
- doInBackground(Params...). Se invoca en el subproceso en segundo plano inmediatamente después de que onPreExecute() termina de ejecutarse. Se le pueden pasar parámetros. Es el método que más trabajo lleva pues es en este donde se hace lo que realmente queremos hacer en background.
- onPostExecute(Result...). Se invoca cuando el proceso en segundo plano ha sido terminado. El resultado devuelto por todo el proceso se pasa a este método como parámetro.

6.1.2. Envío de órdenes

En nuestro proyecto tenemos una Asyntask que no para de observar si hay alguna orden por enviar. Si se cumplen las circunstancias para poder enviar la orden, es decir, estamos conectados por bluetooth y no hay ningún tipo de emergencia, la envía llamando al thread BTClientThread. Este thread es el encargado de enviar los datos y esperar la respuesta del pc servidor. Cuando se están enviando datos se bloquea el acceso de manera que no surjan conflictos al intentar acceder a un mismo recurso. Al recibir el mensaje del pc servidor y dejar libres los recursos necesarios se desbloquea el acceso.

6.1.3. Visualizador de órdenes

Por otro lado, tenemos una Asyntask que se encarga de avisar a la interfaz del display para que actualice los valores de las acciones que está llevando a cabo el robot en cada momento.

El proceso que se lleva es el siguiente. Cuando se envía una orden desde el thread BTClientThread se espera a que la aplicación del pc servidor envíe la posición actual del robot. Una vez se recibe el mensaje, se formatea este y se actualizan los valores de la clase Visualizador. Después de la actualización se avisa a la Actividad DisplayActivity mediante un manejador.

6.2. Versatilidad Android

El mundo de los dispositivos móviles es muy complejo. Existe multitud de hardware diferente. Debemos hacer que nuestras aplicaciones sean internacionales y multidispositivo. Android presenta una serie de características que ayudan al programador para que esta labor sea más fácil. Nosotros en este punto hablaremos de las relativas al idioma e interfaces en general.

6.2.1. Idiomas

Android nos permite crear nuestras aplicaciones con soporte a varios idiomas. En nuestro caso la aplicación soporta Inglés y Español.

En algunos otros entornos de desarrollo, esta característica resulta "tediosa" porque es necesario hacer uso de librerías, scripts extras o archivos dedicados a indicar cuáles cadenas de texto serán las que se traducirán según la configuración de idioma que defina el usuario. Afortunadamente, en Android es algo muy sencillo de implementar.

En Android podemos definir los textos que aparecen en pantalla en un archivo llamado strings.xml. Dentro de este archivo tenemos datos pares (nombre y valor). Dicho archivo debe situarse en el directorio res/values/ . El archivo values/strings.xml podemos decir que incluye el idioma por defecto de la aplicación.

Si queremos añadir otro idioma debemos crear otro archivo strings.xml cuyos datos sean los mismos que el strings.xml del idioma por defecto pero en el par (nombre , valor) debemos cambiar el valor por el del idioma deseado. Para que la aplicación sepa a que idioma corresponde debemos meter el archivo en la carpeta res/values-en/ para inglés por ejemplo. El -x es un modificador Android que sirve a la aplicación para diferenciar entre idiomas. El modificador de idioma es el correspondiente al listado ISO 639-1.

6.2.2. Layouts

Las interfaces se sitúan en la carpeta res/layout/. Para crear una interfaz estáticamente basta con crear un xml en este directorio y

enlazarlo desde el código.

Si queremos crear diferentes tipos de interfaz para distintos dispositivos debemos crear en la carpeta res con carpetas layout definidas con modificadores de pantalla (small, normal, large, xlarge,land,etc...). Se pueden acumular los modificadores, por ejemplo, /layout-small-es/ indica que esta carpeta contiene las interfaces de la aplicación cuando la pantalla es pequeña y su idioma es español. Las carpetas contendrán los xml de las interfaces. No es necesario repetir todas las interfaces para cada modo indicado. Esto es gracias a que si no encuentra una pantalla en el tipo indicado cargará la por defecto, es decir, la de la carpeta sin modificadores. En mi caso las interfaz del main se descuadraba al cambiar el idioma, como al resto no le sucedía nada, en la carpeta layaout-en sólo incluí una nueva interfaz.

En nuestra aplicación hemos creado interfaces según los idiomas y tamaños de pantalla. Para así permitir que la aplicación funcione en tabletas y móviles tanto en español como en inglés.

Por último me gustaría destacar que en Android también hay modificadores para las imágenes que se guardan en res/drawable/. Los modificadores más usualmente utilizados son los relativos a la densidad de la pantalla: low (ldpi), medium (mdpi), high (hdpi) y extra high (xhdpi).

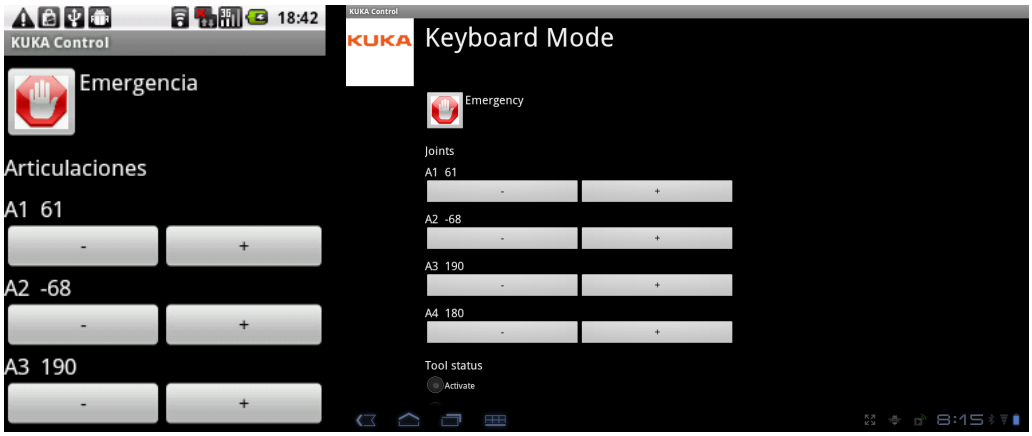

Figura 20: Interfaz de la introducción de órdenes mediante teclado

En la figura(20) se muestran unas interfaces correspondientes a la misma pantalla pero adaptadas a diferentes tipos de dispositivos e idiomas.

7. Conclusiones y trabajos futuros

7.1. Conclusiones

La aplicación desarrollada es útil para las necesidades de la empresa KUKA, ya que esta pretende su uso para la captación de clientes en ferias. No obstante, desde la perspectiva de eficiencia el robot se mueve mucho más lento cuando lo maneja un humano que cuando es dirigido por código programado. Todo esto es debido a que el robot inicialmente no fue creado para esta finalidad. Esto que inicialmente parece un defecto es una virtud muy buena ya que la utilización y combinación de elementos que inicialmente no fueron creados con esa finalidad potencia la evolución de la tecnología.

Personalmente este proyecto a supuesto un gran crecimiento personal y profesional. Me ha permitido, trabajar con profesionales del sector de reconocido renombre y la creación de una aplicación muy original que servirá de base para próximos proyectos de otros estudiantes. Este proyecto combina varias materias de gran actualidad como son la robótica, los dispositivos móviles Android y la realidad aumentada. Por otra parte,poder hacer un plan teórico y verlo realizado ha sido emocionante. Espero seguir consiguiendo participar en proyectos de este calibre.

7.2. Trabajos futuros

En un trabajo futuro podríamos:

- añadir conexión wifi controlando que el ordenador se conecte únicamente con un dispositivo móvil.
- eliminar el ordenador servidor y conectarnos directamente con el pc del robot.
- intentar conseguir una aplicación lo más próxima posible a una aplicación de tiempo real.
- aumentar el número de robots que puedan ser manejados.
- aumentar la complejidad de los programas que puedan ser programados desde el dispositivo.

Actualmente el alumno Julián Martínez Jiménez del Máster en Automatización e Informática Industrial de la UPV está desarrollando una aplicación apoyada en la nuestra. La finalidad de su proyecto es poder controlar un robot industrial, empleando la técnica de realidad aumentada como un elemento de sensorización. De esta forma se pretende poder controlar el robot, su posición y los obstáculos o

elementos que están en su entorno de trabajo. El robot de esta forma podrá evitar obstáculos generados mediante realidad aumentada o ir a estos objetos para situarse sobre ellos y cogerlos.

8. Referencias

- http://fjep.sourceforge.net/fjeptutorial.html Describe como crear .jar que puedan ser ejecutados en cualquier máquina con java.
- http://luugiathuy.com/2011/02/android-java-bluetooth/ Es un código ejemplo de comunicación bluetooth en Android.
- http://obviam.net/index.php/texture-mapping-opengl-android-displaying-images-using-opengl-and-squares/ Enseña como dibujar polígonos y superponer texturas en OpenGL.
- http://www.java2s.com/Code/Android/2D-Graphics/LoadBitmapwithContext.htm Muestra como cargar texturas de resources de Android y convertirlas a bitmap.
- http://suniandroid.blogspot.com.es/2011/03/crear-el-paquete-apk-para-el-market.html Describe como crear .apk's con identificador de autoría.
- http://www.vogella.com/articles/AndroidGestures/article.html#resources_databinding Programa Android para el reconocimiento de gestos.

9. Bibliografía

- http://www.kuka-robotics.com/spain/es/company/ Página de la empresa Kuka.
- http://code.google.com/p/andar/ Página oficial del proyecto de realidad aumentada AndAR.
- http://developer.android.com/index.html Página de referencia para los desarrolladores de aplicaciones Android.
- http://www.javabeat.net Página con información sobre Java.
- http://androideity.com Página con información útil en español para iniciarse en la programación Android.