Proyecto Fin de Carrera

Título: Unidad inalámbrica de programación para robots educativos con dispositivos móviles
Autor: David Guillermo Morales Sáez
Fecha: Septiembre de 2013
Tutor: Enrique Fernández García

Agradecimientos

Agradezco a mi tutor todo el apoyo que me ha dado, ayudándome en las fases más críticas del proyecto y aportándome ideas que han facilitado la creación de este Proyecto Fin de Carrera, además le doy las gracias por tener la paciencia de ayudarme en sus vacaciones de Agosto.

No puedo ignorar las ayudas recibidas para la realización del proyecto, y debo agradecer a Beatriz Correas Suárez por cederme una tableta para poder realizar el proyecto, a Ricardo Pérez García por la cesión de los chip MAX232, y a Pedro Medina Rodríguez, por todo el conocimiento aportado sobre los robots y la robótica en general.

También quiero agradecer a mi familia el apoyo dado; a mis padres por darme todo lo que he necesitado, siendo un gran pilar en el que he podido sostenerme en los momentos más duros; y a mi hermana, por darme ánimos cuando más débil estaba y por no dudar un momento en ayudarme siempre que ha podido.

Agradezco a mis todos compañeros con los que he compartido parte de este gran camino que ha sido el estudio de esta carrera, Ingeniería Informática, pero sobre todo agradezco a Daniel, por ayudarme y acompañarme gran parte de la misma, a Alberto, por soportarme en esas tediosas tardes haciendo prácticas, y a Gabriel, por todos esos momentos (tanto buenos como malos) que hemos compartido, trabajando duro e intentando hacerlo todo de una forma diferente a la usual, innovando siempre que hemos podido.

Índice

1. Resumen

El objetivo de este Proyecto de Fin de Carrera es diseñar e implementar una herramienta software que nos permita controlar de forma inalámbrica un brazo robótico. Tras el desarrollo del proyecto, se obtendrá una aplicación que permitirá al usuario realizar las acciones de las que dota una paleta estándar del brazo robótico, como pueden ser controlar directamente el brazo robótico o interactuar con la memoria EEPROM[1] del mismo, además de permitir el desarrollo de rutinas o programas para el mismo sin requerir un entorno de desarrollo externo e interactuar con la memoria interna del brazo robótico para permitir el envío y recepción de rutinas.

La aplicación ha sido desarrollada para Android debido a las facilidades para su desarrollo, como la gran portabilidad que dota el sistema operativo y por el bajo coste de desarrollo. Por otro lado, se necesita un puente que una la aplicación Android y el brazo robótico, y para lo que se ha utilizado una placa Arduino con un módulo Bluetooth, el cual permite que una tableta Android pueda comunicarse con la placa.

Para la realización del proyecto, se han utilizado múltiples herramientas, tanto software como hardware. Centrándonos en el software, se han utilizado dos entornos de desarrollo distintos, uno para el desarrollo de la aplicación para Android y otro para el programa de control de flujo para el Arduino. Por otro lado, ha sido necesario utilizar una tableta Android para poder depurar la aplicación, así como una placa Arduino para la depuración de la interconexión entre la tableta y el Brazo Robótico. No hay que olvidar que para realizar las pruebas fue necesario utilizar un Brazo Robótico, por lo que se ha utilizado el RHINO XR4.

La motivación principal para el desarrollo de este proyecto fue la idea de poder acercar el mundo de la robótica a aquellas personas ajenas al mismo, además de dotar de un sistema de control para el día de mañana, evitando problemas de software caduco y antiguo, cosa que empieza a suceder a día de hoy, obligando a utilizar máquinas virtuales para poder trabajar.

Cabe destacar la amplitud del trabajo, tratando problemas de diversa índole, que van desde la implementación de una aplicación sobre un sistema táctil y portable hasta el estudio y dominio de la comunicación inalámbrica Bluetooth. Ambos son temas que no forman parte de los conocimientos adquiridos a lo largo de la carrera, por lo que me han permitido comprender mejor aspectos de la informática que permanecían velados para mí.

La memoria está dividida en varias partes y capítulos. Se comienza con los capítulos del "Resumen" y la "Introducción", donde se da una visión de conjunto y se plantea una introducción a los principales temas tratados en el proyecto, además de exponerse las motivaciones y objetivos, seguido por un capítulo dedicado al "Estado actual del arte", donde se dota de una visión más pormenorizada del campo en cuestión, explicándose las soluciones software ya existentes.

[1] La memoria EEPROM es un tipo sólo de memoria no volátil de lectura que puede ser programada, borrada y reprogramada eléctricamente

El siguiente apartado es un capítulo dedicado a la “Metodología, recursos y plan de trabajo” del proyecto, seguido de un “Análisis” exhaustivo del proyecto, mostrando el modelo de dominio y los distintos casos de uso. Se completa esta sección con el “Diseño” tanto de la aplicación como del programa para la placa Arduino, donde se realiza una descripción completa de las soluciones adoptadas para que el proyecto cumpla con todos los requisitos.

Tras completar estos apartados se encuentra la “Implementación”, donde se detallan las herramientas utilizadas para completar el proyecto, además de aquellos elementos que se han tenido que añadir para garantizar el correcto funcionamiento del mismo. Lo siguiente que encontrarán serán las “Pruebas y resultados”, donde se detalla la validación del sistema y, por último, finaliza con un apartado dedicado a las “Conclusiones y perspectiva de futuro”, en el que se muestran las conclusiones a las que se ha llegado con la realización de este proyecto y las posibles ampliaciones del mismo en un futuro.

2. Introducción

Para entender los distintos ámbitos en los que se ha trabajado en este proyecto, es necesario comprender una serie de elementos muy diferentes que, unificándolos, me ha permitido diseñar e implementar un sistema de control inalámbrico para brazos robóticos.

2.1 Sistema Robótico

En términos generales, podemos hablar de la robótica como la conexión inteligente entre la percepción y la acción, donde la integración de información sensorial ocupa un lugar privilegiado. En el ámbito de la docencia, la robótica no tiene tanto alcance como el que dispone en el ámbito industrial, por lo que el soporte de los terminales es más limitado. Esto no evita que existan sistemas diseñados para la docencia. Algunos de esos sistemas son el RHINO, de la compañía Rhino Robotics, o el SCORBOT, de la compañía Intelitek.

Este tipo de robot, o brazo robótico, están diseñados y enfocados para la docencia, por lo que, si bien palidecen frente a los robots industriales debido a la complejidad de estos últimos, cumplen perfectamente su cometido. Su utilidad va desde enseñanzas de cinemática y cálculos de trayectorias hasta el dominio de las dinámicas y el control de un sistema robótico. Para poder controlar este tipo de sistemas, se necesita un controlador que transforme las órdenes de programación en señales eléctricas para los motores.

Para cumplir con esta necesidad, se han desarrollado varios controladores, como el RHINO MK-IV de la compañía Rhino Robotics. Una vez conectados el brazo robótico y el controlador, ya se pueden desarrollar rutinas para el mismo sin necesidad de dominar las señales para que las rutinas puedan ejecutarse correctamente en el hardware.

Por ello, el controlador suele implementar una serie de órdenes sencillas, que permiten a todo usuario controlar el brazo robótico con unos requerimientos menores en cuanto a sus motores. Estas órdenes están codificadas en ASCII de 7 bits, con dos bits de parada y de paridad impar, transmitiéndose a 9600 bits por segundo. Para controlar el robot, suelen haber dos modos diferenciados: control con una paleta y control con una rutina.

Una paleta es un pequeño dispositivo que permite un control básico de los motores y sensores internos y externos disponibles. El control suele ser directo y sin mostrar al usuario las operaciones que se están haciendo en segundo plano. De esta forma, un usuario inexperto puede controlar el brazo robótico, moviendo sus distintos motores o escribiendo en las salidas, sin dominar el lenguaje del controlador.

Por otro lado, el usuario puede desarrollar una rutina para que se ejecute en el controlador. Estas rutinas son almacenadas en una memoria interna del controlador, para que el usuario pueda ejecutarlas cuantas veces desee. Para poder usar esta funcionalidad, el

usuario debe enviar el código fuente de la rutina al robot, generalmente con una orden predefinida. Una vez hecho esto, el brazo robótico está listo para funcionar.

Estos robots no suelen ser demasiado grandes, facilitando su uso y su mantenimiento, mientras que el número de grados de libertad[2] no tiende a ser excesivamente elevado, como en el caso del RHINO XR4, que dispone de cinco grados de libertad, además de una pinza. Elevar el número de grados de libertad, el tamaño o la complejidad no facilitaría que se realizase la labor docente para la cual han sido diseñados.

La conexión con el controlador puede ser de muchas formas, pero para conectarnos con el controlador RHINO MKIV es necesario un cable serial RS-232[3], por donde podremos intercambiar información y órdenes con el controlador. Pero, para este proyecto, necesitamos algún elemento que nos permita enviar y recibir datos mediante una conexión inalámbrica. Para ello, se empleará una placa Arduino.

2.2 Comunicación Robot-Tabletas

La comunicación que posee el entorno del robot está basada en el estándar RS-232, propio de dispositivos del momento en el que fue construido. Sin embargo, las tabletas y prácticamente todos los dispositivos móviles actuales poseen Bluetooth como medio de comunicación con dispositivos externos muy variados, desde consolas hasta auriculares. Por este motivo, se ha considerado deseable realizar las acciones necesarias para que el robot pueda comunicarse mediante Bluetooth gracias a un enlace entre el estándar RS-232 y el Bluetooth, tanto físicamente como a nivel de protocolo.

2.2.1 Bluetooth

Bluetooth es un estándar de tecnología inalámbrica para el intercambio de datos a cortas distancias. Fue creado por Ericsson en 1994, concebido como una alternativa inalámbrica a los cables RS-232 (también conocido como cable serie) y fue estandarizado bajo el código IEEE 802.15.1 y es dirigido por el Bluetooth Special Interest Group (SIG) [4], si bien el estándar ya no tiene soporte.

---

[2] Un número de grados de libertad indica el número de motores que pueden moverse independientemente del resto.

[3] La interfaz RS-232 es un estándar de intercambio de datos binarios entre un equipo terminal de datos y un equipo de comunicación de datos. Es un conector de 25 pines, aunque se ha popularizado el conector de 9 pines.

[4] El SIG es una asociación privada sin ánimo de lucro fundada en 1998 y está formado por unas 18000 empresas que se encargan del desarrollo de los estándares de Bluetooth y de la licencia de las tecnologías y marcas asociadas a la misma.

Utiliza transmisiones de radio de onda corta, entre los 2400 y 2483.5 MHz para crear una red personal privada. Para su funcionamiento, se define un dispositivo Maestro, que será quien gestione la red. Este dispositivo tiene un número máximo de 7 conexiones con distintos dispositivos. Los dispositivos a los que se conecta son llamados dispositivos Esclavos, si bien se puede cambiar los roles de forma dinámica.

Inicialmente se consideró, como procedimiento de conexión más sencillo, usar un "adaptador Bluetooth-RS232" para este proyecto. Sin embargo, tiene varios inconvenientes que van desde la no disponibilidad en el mercado local y, sobre todo, que no permite intervenir en la comunicación y, a priori, en el desarrollo del proyecto interesa tener el control total del canal de comunicación.

Por el motivo anterior, se prefirió ampliar los objetivos del PFC poniendo un dispositivo programable que permitiese una comunicación con el robot y que dejase abierto el sistema para mayor control de la comunicación o ampliación futura del sistema. En este sentido, se consideró como candidato ideal el Arduino para servir de puente entre el robot y la paleta.

2.3 Arduino

Arduino es una plataforma de hardware libre, basada en una pequeña placa con un microcontrolador Atmel AVR y con distintos puertos de entrada y salida. Este sistema puede utilizar esos puertos para obtener información de sensores u otro tipo de señales y puede hacer uso de actuadores como luces o motores. La programación del mismo se hace gracias a un entorno de desarrollo basado en Processing [5] y no requiere conexión con ningún dispositivo de control externo para ejecutar el programa almacenado en su memoria.

El proyecto Arduino empezó en el año 2005, en Ivrea, Italia. Iniciado por Massimo Banzi y David Cuartielles, lo llamaron el Arduino de Ivrea, en referencia al personaje histórico de la ciudad, y empezaron a producir las placas en una fábrica de la ciudad. El proyecto Arduino es una rama de la plataforma open source Wiring [6], por lo que utiliza un lenguaje basado en Wiring para la programación a bajo nivel.

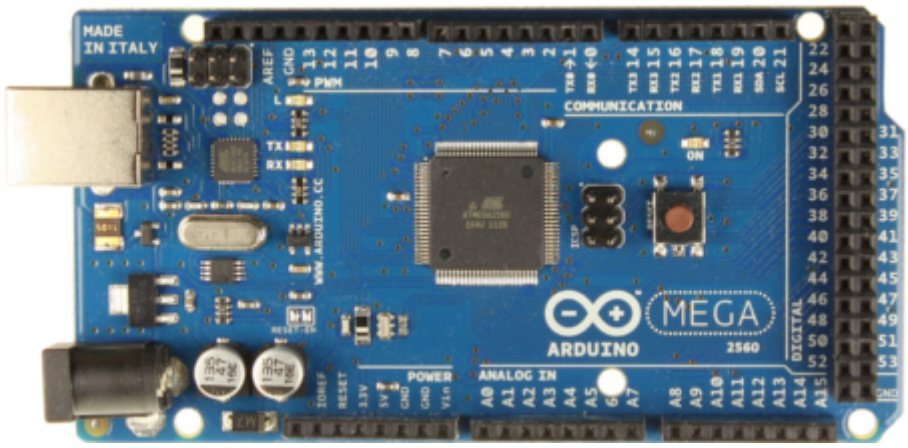
Arduino está desarrollado para ser escalable a nivel hardware, es decir, para poder añadir distintos módulos según sean necesarios, desde un módulo con una pantalla LCD hasta uno con una antena Bluetooth. Esto permite que los desarrolladores sólo tengan el hardware que necesiten, sin malgastar energía en una serie de prestaciones a las que no se les va a dar uso alguno.

[5] Processing es un lenguaje y un entorno de programación open source para el desarrollo de aplicaciones gráficas, dirigido a diseñadores gráficos y a la enseñanza de la computación gráfica.

[6] Wiring es una plataforma de prototipos electrónicos open source compuesta por un lenguaje de programación, un entorno de desarrollo integrado y un microcontrolador. Comenzó su andadura en el 2003, por Hernando Barragán.

La plataforma Arduino se ha posicionado rápidamente en entornos universitarios gracias a su bajo coste, al ser multi-plataforma, disponer de un entorno de programación sencillo, ser de código abierto y altamente escalable, por lo que los proyectos desarrollados en entornos universitarios se ven fácilmente atraídos a este tipo de soluciones.

Se ha utilizado la placa Arduino Mega 2560 que estaba disponible en el centro. Esta placa dispone del microcontrolador[7] ATmega2560 de Atmel, con una memoria flash de 256KB y una memoria SRAM[8] de 8KB, 16 pines analógicos de entrada y 54 pines digitales de entrada y salida. Los niveles de tensión de estos pines oscilan entre los 0 y los 5 voltios, mientras que los niveles de tensión con los que trabaja el controlador del brazo robótico oscilan entre los -12 y los 12 voltios, por lo que es necesario construir una adaptación externa de niveles, como se describirá más adelante en esta memoria.



Hay que comentar que, para poder conectarnos vía Bluetooth con un dispositivo externo, hace falta algún elemento que permita al Arduino realizar ese tipo de conexiones y, para ello, hemos utilizado un módulo llamado Xbee Shield. Este módulo, desarrollado en colaboración con Libelium, permite a una placa Arduino conectarse de forma inalámbrica utilizando distintos protocolos, entre ellos el Bluetooth.

Esta placa dispone de dos jumpers[9] donde se define el tipo de conexión que se va a hacer, ya sea USB (la placa se conecta directamente con un ordenador), Xbee (la placa recibirá y enviará datos con el chip Arduino, pero éste no podrá enviar o recibir datos vía USB) o un modo intermedio, donde la placa recibirá y enviará datos del chip Arduino y éste podrá enviar y recibir información vía USB. Este modo es el que se ha utilizado en el proyecto debido a las

[7] Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria.
[8] Memoria Estática de Acceso Aleatorio (Static Random Access Memory o SRAM) es un tipo de memoria basada en semiconductores capaz de mantener los datos sin necesidad de un circuito de refresco. Sin embargo, es un tipo de memoria volátil, por lo que hay que mantener la alimentación eléctrica para mantener los datos.
[9] Un jumper es una pequeña funda de plástico que conectan dos o más pines.

necesidades del mismo. Por otro lado, para programar esta placa es necesario utilizar los comandos AT[10].

2.4 Tableta Android

Dentro de este apartado, podemos diferenciar dos elementos importantes, las tabletas y el sistema operativo, Android.

2.4.1 Tabletas

Las tabletas son unos dispositivos portátiles mayores que un teléfono móvil o PDA[11] con la que el usuario interactúa utilizando sus dedos o un pequeño lápiz o *stylus*. La primera tableta que se puso a la venta fue la Microsoft Tablet PC, desarrollada por Microsoft en el año 2001. Pero su popularidad fue bastante marginal y este tipo de dispositivo no tuvo mucho éxito. Esto cambió en el año 2010, cuando Apple, de mano de Steve Jobs, presento el iPad. Este producto revolucionó el mercado, haciendo que gran cantidad de fabricantes decidiesen sacar productos con unas capacidades similares.

Tras este exitoso lanzamiento, comenzó una "guerra" entre los distintos fabricantes. En este conflicto económico, podemos ver tres grandes contendientes: Apple, con su gama de iPads y el sistema operativo iOS; Microsoft, con su gama Surface y con Windows RT o Windows 8; y Google, con todas las tablets del ecosistema Android.

Las tabletas y los teléfonos inteligentes actuales, además de ser dispositivos táctiles de visualización excelentes, poseen varios sensores de gran diversidad, como lo son los sensores de posición y aceleración, lo que su uso puede ser muy interesante para futuros proyectos en entornos robóticos.

La tableta utilizada en este proyecto ha sido una Asus Eee Pad Transformer TF101, que fue cedida por doña Beatriz Correas Suárez. Esta tableta dispone de una pantalla LED de 10.1 pulgadas, un procesador NVIDIA Tegra2, una salida Mini HDMI, además de disponer de acelerómetro, sensor de luz, giroscopio y GPS.

[10] Los comandos AT (o comandos Hayes) es un lenguaje desarrollado por la compañía Hayes Communications para configurar y parametrizar módems.

[11] PDA, u ordenador portátil, es un pequeño ordenador de mano originalmente diseñado como agenda electrónica. A día de hoy, sus funcionalidades son muy superiores a las ideadas inicialmente, permitiendo la lectura de correos electrónicos, la escritura de documentos o incluso el uso de videojuegos.

2.4.2 Android

El sistema operativo Android es el sistema operativo móvil más extendido a día de hoy. Basado en Linux, fue diseñado para dispositivos con una pantalla táctil. Inicialmente fue desarrollado por Android Inc., hasta que Google decidió comprar la empresa y dirigir su camino. Android fue presentado en el año 2007, pero hasta el año siguiente no salió ningún dispositivo con este sistema operativo. Este dispositivo fue el HTC Dream, que salió en octubre de 2008.

Android está construido sobre el lenguaje C[12], con algunas librerías en C++[13], pero la inmensa mayoría de los programas, o aplicaciones, desarrollados para el mismo están escritos en Java[14], lo que implica que las aplicaciones trabajan sobre una máquina virtual, no sobre el propio dispositivo, es decir, que un desarrollador puede diseñar una aplicación para un dispositivo y utilizarla en otros dispositivos sin necesidad de recompilar o modificar el código fuente.

Además, se basan en la filosofía de software libre, lo que ha permitido que el ecosistema Android crezca exponencialmente, consiguiendo copar la tasa de mercado de teléfonos inteligentes o *smartphones*. Gracias a esta filosofía, cualquier desarrollador puede diseñar e implementar una aplicación para cualquier dispositivo, lo que facilita el aprendizaje e incrementa el número de aplicaciones diseñadas para el mismo.

[12] El lenguaje C es un lenguaje de programación creado por Dennis Ritchie en 1972 en los Laboratorios Bell. es un lenguaje imperativo, estructurado y poco tipado, muy orientado al desarrollo de Sistemas Operativos, entre ellos, Unix y Linux.

[13] El lenguaje C++ es un lenguaje de programación creado por Bjarne Stroustrup en los años 80. Inicialmente, fue una extensión del lenguaje C, incluyendo mecanismos de manipulación de objetos.

[14] El lenguaje Java es un lenguaje de programación creado por James Gosling en 1995 en Sun Microsystems. Es un lenguaje orientado a objetos y con una gran portabilidad, ya que trabaja sobre una máquina virtual, no sobre la propia máquina.

Si bien Android empezó con desventaja frente a su competencia directa, Apple, su gran crecimiento ha conseguido que el número de dispositivos Android supere drásticamente a los dispositivos de Apple. Esto ha permitido que la comunidad de desarrolladores de Android sea extensa, facilitando, aún más si cabe, el aprendizaje y desarrollo de aplicaciones. Sabiendo esto, es lógico pensar la razón por la cual se decidió utilizar un dispositivo basado en Android para este proyecto:

- El desarrollo es gratuito, no requiere pagar ningún tipo de tasa
- Hay gran cantidad de información y el desarrollo de las aplicaciones no es complejo
- Al trabajar sobre una máquina virtual, una aplicación puede funcionar sobre múltiples dispositivos

3. Estado actual del Arte

En los últimos años, la informática ha crecido de forma asombrosa, llegando a hitos que antes sólo estaban en sueños. Pero en el ámbito de la robótica orientada a la docencia disponible en el Departamento de Informática y Sistemas, hace años que no se actualiza. Uno de los robots disponibles es el RHINO XR4, de Rhino Robotics Inc. Para trabajar con el controlador RHINO MK-IV, disponemos el software RobotTalk, un software diseñado por la misma compañía para comunicarse con el controlador. Este software tiene soporte hasta la versión Windows XP, por lo que, con los sistemas actuales, puede dar problemas en la ejecución.

Este software permite al usuario interactuar con el controlador, trabajando como si de la paleta se tratase, enviando o recibiendo programas y ejecutando la rutina almacenada en la EEPROM. Si bien cumple con la mayoría de las funciones que buscamos para este proyecto, no es inalámbrico, es más, se requiere un ordenador con una versión antigua de Windows (o un emulador instalado) para su funcionamiento, complicando aún más la posibilidad de disponer en un futuro del mismo sin temor a encontrarse con un fallo o sin poder utilizarlo.

Debido a esta situación, se ha planteado este proyecto, buscando facilitar y permitir el uso del controlador y el brazo robótico para un futuro no demasiado lejano. Al diseñar un nuevo sistema inalámbrico, evitamos vernos obligados a disponer de un ordenador continuamente conectado al controlador del robot, además de que, al ser una aplicación para un dispositivo Android, puede ser ejecutada sobre cualquier dispositivo Android, por lo que no habrá problemas al trabajar con ella sobre dispositivos futuros.

4. Metodología, recursos y plan de trabajo

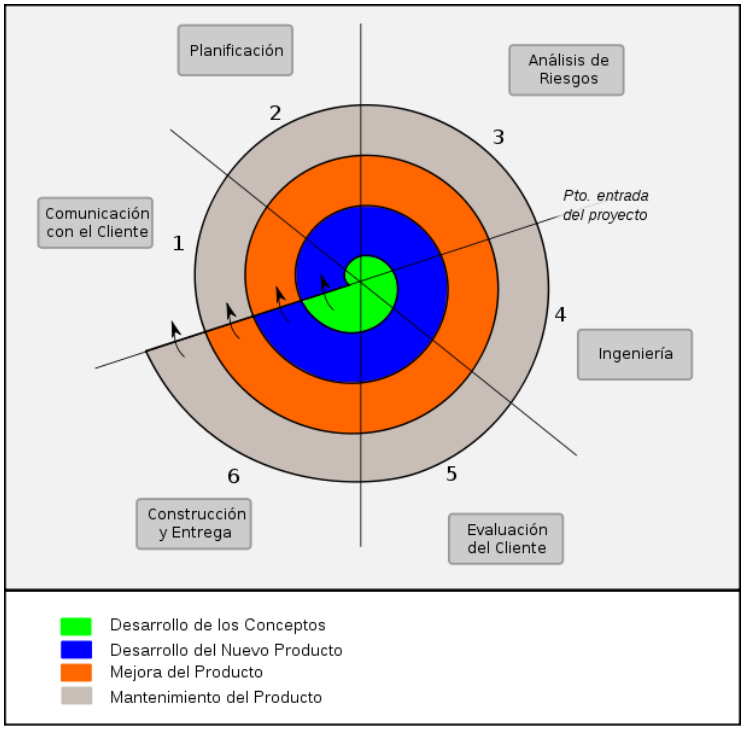
4.1 Metodología

4.1.1 Definición del Modelo

En todo proyecto informático, es recomendable definir un modelo de desarrollo, un ciclo de vida de la realización del mismo. Esto se debe a que es muy recomendable normalizar los procesos del proyecto para conseguir una mejor estructuración y definición del mismo. La elección del modelo a seguir depende de las necesidades y restricciones del proyecto, por lo que hay que ser cuidadosos al escoger, ya que una mala elección puede resultar catastrófica en una fase tardía del desarrollo, llegando a obligar a empezar desde cero gran parte del trabajo por un fallo en la concepción inicial.

Al definir la metodología de trabajo de este proyecto, hubo que evaluar las necesidades y restricciones del mismo. Debido a que es un proyecto altamente segmentable, inicialmente se consideró realizar un desarrollo por prototipos, definiendo cada fase como un estado distinto del proyecto, incrementando sus funcionalidades con cada prototipo.

Sin embargo, analizando con mayor profundidad el proyecto, se puede ver que gran parte de sus componentes están entrelazados, teniendo distintos módulos que hacen uso de los mismos elementos, por lo que se optó por un desarrollo en espiral, el cual permite ir añadiendo funcionalidades con el tiempo, a la vez que se facilita el mantenimiento o adaptación de las funcionalidades ya desarrolladas.



El modelo de proceso de software en espiral o evolutivo, combina la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. En este modelo, el software se desarrolla en una serie de versiones incrementales, inicialmente con modelos muy sencillos y básicos, pero en las últimas iteraciones, las versiones generadas son cada vez más completas, acercándose y, finalmente, igualando al modelo deseado.

En cada ciclo, se identifican los objetivos a conseguir en el ciclo actual, las distintas alternativas posibles que hay para obtener esos objetivos, y las restricciones con las que esperamos encontrarnos. Una vez identificados, se formula una estrategia efectiva para resolver las distintas fuentes de riesgos posibles y se inicia el desarrollo. Una vez completado el prototipo, se plantea el prototipo para el próximo ciclo. En cada ciclo se completa una revisión que incluye todo el ciclo anterior y el plan para el siguiente.

4.1.2 Aplicación del Modelo

Una vez decidido el modelo a utilizar, se iniciará la aplicación del mismo en el proyecto. Para ello, se plantearon una serie de objetivos genéricos, divididos en los dos distintos sistemas en los que se trabajaría: la tableta Android y la placa Arduino. Los objetivos del desarrollo en la tableta Android fueron los siguientes:

1. Definir el estándar a utilizar y crear una clase funcional que lo utilice correctamente
2. Crear una actividad que, partiendo de la clase definida en el paso anterior, permitiese crear instrucciones completas
3. Definir el modelo de comunicación Bluetooth e implementarlo

4. Conseguir enviar instrucciones vía Bluetooth a la placa Arduino correctamente
5. Conseguir enviar programas vía Bluetooth a la placa Arduino correctamente
6. Definir e implementar la actividad que simulase la Paleta del Brazo Robótico correctamente

Estos objetivos fueron definidos inicialmente, y se ha trabajado en base a ellos. Con respecto a los objetivos para la placa Arduino, fueron los siguientes:

1. Comprender el funcionamiento de las comunicaciones en Arduino
2. Definir y desarrollar la comunicación vía Bluetooth con la placa Bluetooth
3. Enviar y recibir mensajes correctamente vía Bluetooth con la tableta Android
4. Definir y desarrollar la comunicación vía Serial con el Brazo Robótico
5. Enviar y recibir mensajes correctamente vía Serial con el Brazo Robótico
6. Unificar ambas comunicaciones, creando un puente entre ambas

Con estos objetivos planteados, se fue desarrollando la aplicación Android y el programa del Arduino por separado, hasta que, llegados a un punto, hubo que igualar los desarrollos para realizar las pruebas. Esto permitió descubrir fallos y problemas de rendimiento, lo que permitió redistribuir la carga de trabajo sin sufrir grandes retrasos y modificaciones.

4.1.3 Recursos necesarios

A continuación se describen los recursos que se han utilizado, tanto hardware como software, para la realización del presente PFC y para la experimentación y validación de los resultados obtenidos.

4.1.3.1 Recursos Hardware

Para la realización del proyecto, se han requerido los siguientes materiales:

- Ordenador Portátil:
  - MacBook Pro mid 2009
  - Procesador: Intel Core 2 Duo 2.53GHz
  - Memoria RAM: 4GB DDR2 a 1066MHz
  - Disco Duro: 500GB Serial ATA a 5400 rpm

- Tableta Android
  - Asus Eee Pad Transformer TF101
  - Sistema Operativo: Android 3.2
  - Pantalla: 10.1” LED (1280x800)

    - Salidas: USB 2.0 y MiniHDMI

- Placa Arduino
    - Arduino Mega 2560
    - Microcontrolador: ATmega2560
    - Memoria Flash: 256KB
    - Memoria SRAM: 8KB

- Placa Bluetooth
    - Communication Shield XBee Pro
    - Bluetooth Module for Arduino

- Brazo Robótico
    - Robot educativo Rhino4
    - Cable Serial 232
- Placa Protoboard
    - 2x Chip Max232
    - 8x Condensador 1µF
    - Cables Cobre
    - Osciloscopio

- Grabación
    - Cámara de Video Sony Handycam HDR-SR11E
    - Capturadora de Video Hauppauge HD PVE 2 GE Plus
    - Trípode Manfrotto

De todos estos materiales, tanto el ordenador portátil y el equipo de grabación es propio, mientras que el resto fue cedido por la Escuela de Ingeniería Informática de forma temporal para la realización del proyecto de fin de carrera.

4.1.3.2 Recursos Software

Para el desarrollo del proyecto, se han requerido los siguientes recursos software:

- Sistema Operativo Mac OS X 10.8 Mountain Lion

- Entornos de Desarrollo
    - Android Studio
    - Entorno de desarrollo de Arduino

- Editores de Texto
    - Microsoft Word 2007
    - TextEdit

- Entorno UML
    - ArgoUML

- Utilidades de Grabación
    - HDPVRCapture 3.0
    - iMovie

4.2 Plan de Trabajo

Partiendo de la metodología usada y simplificando, ya que no se mostrarán las distintas iteraciones que se realizarán, si no que se mostrará el camino seguido en el desarrollo, por lo que el plan de trabajo que se ha seguido para la realización de este proyecto de fin de carrera ha sido el siguiente:

1. Documentación y comprensión del funcionamiento del Brazo Robótico
2. Documentación y búsqueda de entorno de desarrollo
3. Definición del estándar del robot
4. Diseño de la Aplicación
    a. Análisis de Requisitos
    b. Diseño
    c. Búsqueda de herramientas
    d. Implementación
5. Diseño del programa Arduino
    a. Análisis de Requisitos
    b. Diseño
    c. Implementación
6. Pruebas y conclusiones
    a. Pruebas de las funcionalidades básicas
    b. Pruebas completas de las funcionalidades
    c. Análisis de resultados
7. Presentación y defensa del PFC
    a. Grabación de su funcionamiento
    b. Montaje de Video para mostrar su funcionamiento
    c. Confección de la memoria del PFC
    d. Preparación y defensa oral del PFC

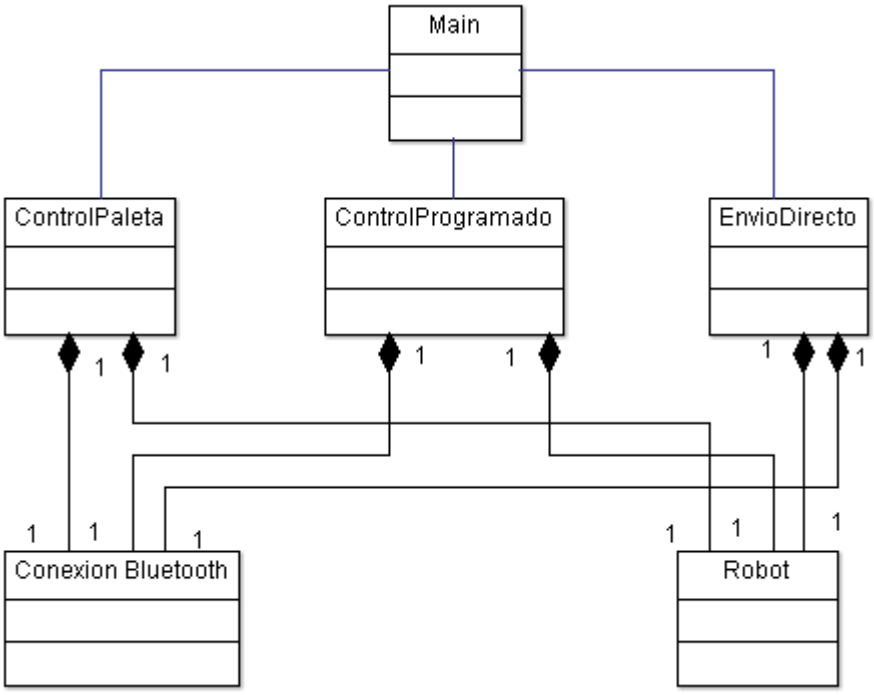
5. Análisis

5.1 Modelo del dominio

Con el fin de capturar claramente los tipos más importantes de objetos del sistema, se ha empleado el modelo de dominio. Este modelo de dominio está compuesto por objetos que representan los elementos que existen o los eventos que suceden en el entorno en el que trabaja el sistema.

Con este proyecto, el usuario podrá tanto controlar directamente el Brazo Robótico como desarrollar sus propios programas para controlarlo, además de utilizar distintos programas, usándolos en la aplicación. La aplicación sólo podrá tener una conexión activa en total, por lo que habría que cortar la conexión establecida con un robot para poder trabajar con otro. Lo mismo sucede con la placa Arduino, sólo podrá estar conectada a una tableta y a un Brazo Robótico. La aplicación de la paleta dota al usuario de tres modos de ejecución con distintas funcionalidades, mientras que la placa Arduino no dispone de múltiples capacidades, si no que siempre tendrá la misma funcionalidad. El modelo de dominio desarrollado dispone de las siguientes entidades:

- Main: representa el estado inicial, punto desde el que parten todas las funcionalidades
- ControlPaleta: representa el estado donde el usuario puede controlar directamente el robot
- ControlProgramado: representa el estado donde el usuario puede crear sus propios programas
- EnvioDirecto: representa el estado donde el usuario puede enviar programas al robot
- Conexión Bluetooth: representa el puente entre la tableta y el Arduino vía Bluetooth
- Robot: representa la entidad que gestiona todos los datos del robot



Modelo de Dominio

Desde la interfaz inicial, el usuario dispone de tres posibles acciones diferentes: controlar el robot directamente, desarrollar un programa o enviar un programa ya hecho. Todas estas acciones requieren de un puente Bluetooth entre la tableta y la placa Arduino, y para ello utilizan una conexión Bluetooth ya establecida. Además, el estándar del robot estará almacenado en la entidad Robot, lo que permitirá un fácil acceso a los datos necesarios para su correcto funcionamiento.

5.2 Modelo de casos de uso

Los casos de uso han sido adoptados casi universalmente para la captura de requisitos de sistemas software y son la herramienta fundamental cuando se identifican y especifican clases, subsistemas e interfaces, cuando se identifican y especifican casos de prueba y cuando se planifican las iteraciones del desarrollo y la integración del sistema. Estos casos de uso proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales con un énfasis especial en el valor añadido para cada usuario individual o para cada sistema externo.

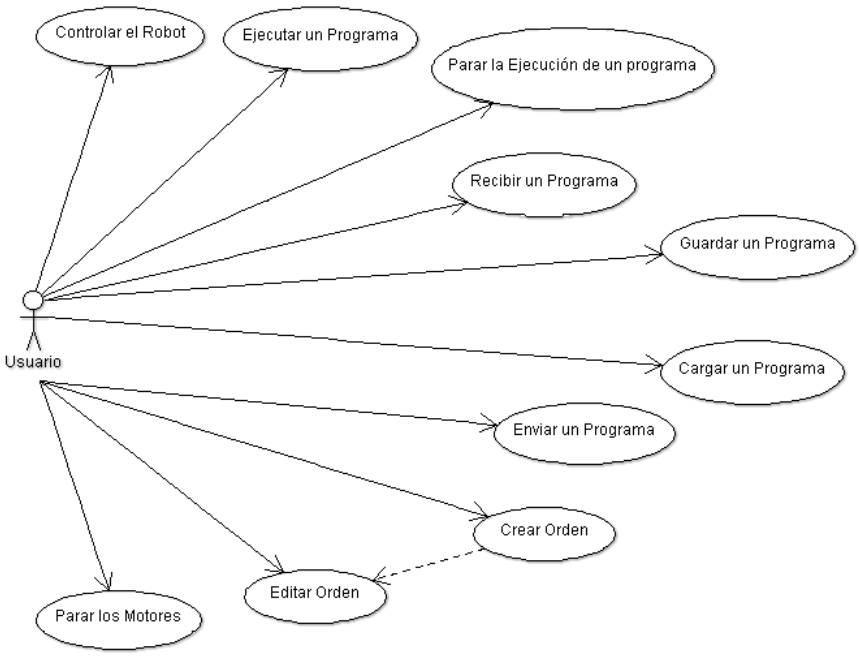
5.2.1 Actores

El modelo de casos de uso describe las capacidades que dispone cada usuario para trabajar con el sistema. Cada uno de éstos se representa con uno o más actores, al igual que representan a cada sistema externo con el que interactúa el sistema. Un actor juega un papel por cada caso de uso con el que interviene. En este proyecto, todas las acciones pueden ser llevadas por un único actor, que es el usuario. Este actor puede tanto controlar el robot directamente como crear programas y enviarlos o recibirlos del robot.

5.2.2 Casos de uso

Cada forma en que los actores usan el sistema se representa con un caso de uso. Los casos de uso son “fragmentos" de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. De manera más precisa, un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia.

El usuario interactuará con los casos de uso del sistema, teniendo como opciones controlar el robot manualmente, crear, cargar, recibir, guardar y ejecutar un programa, utilizando para ello las herramientas de creación, edición y eliminación de órdenes y parar el robot como medida de emergencia.



Casos De Uso del Usuario

A continuación se describen los casos de uso con los que interactúa el actor Usuario:

Caso de uso	**Controlar el Robot**
Actores	Usuario
Tipo	Básico
Resumen	Controla el Brazo Robótico manualmente
Precondiciones	Que esté definido el estándar
Postcondiciones	Ninguna.
Flujo Principal	Se controlará el brazo robótico manualmente utilizando las opciones disponibles.
Flujos alternativos	Ninguno.
Excepciones	Ninguno.

Caso de uso	**Ejecutar un Programa**
Actores	Usuario
Tipo	Básico
Resumen	Ejecuta el programa de la EEPROM del Brazo Robótico
Precondiciones	Que exista un programa en la EEPROM
Postcondiciones	Se inicia la ejecución del programa
Flujo Principal	Se ordenará al Brazo Robótico que inicie la ejecución del programa que está almacenado en la EEPROM.
Flujos alternativos	Ninguno.
Excepciones	Puede suceder que no haya ningún programa en la EEPROM, en cuyo caso no hará nada.

Caso de uso	**Parar la Ejecución de un programa**
Actores	Usuario
Tipo	Básico
Resumen	Para la ejecución del programa que esté ejecutándose
Precondiciones	Debe haber un programa ejecutándose
Postcondiciones	No habrá ningún programa en ejecución
Flujo Principal	El usuario parará la ejecución del programa que esté en funcionamiento en el Brazo Robótico.
Flujos alternativos	Ninguno.
Excepciones	Ninguno.

Caso de uso	**Recibir un programa**
Actores	Usuario
Tipo	Básico
Resumen	Recibe un programa del Brazo Robótico
Precondiciones	Debe haber un programa en la EEPROM.
Postcondiciones	Se dispone del programa almacenado en la EEPROM.
Flujo Principal	El usuario solicitará el programa que está almacenado en la EEPROM .
Flujos alternativos	Ninguno.
Excepciones	Si no hay ningún programa en la EEPROM, no se recibirá nada.

Caso de uso	**Guardar un programa**
Actores	Usuario
Tipo	Básico
Resumen	Almacena un programa en un fichero en la memoria de la tableta.
Precondiciones	Debe haber un programa.
Postcondiciones	Se dispone de un fichero con el programa.
Flujo Principal	El usuario seleccionará el fichero donde se almacenará el programa y éste se creará o se sobreescribirá.
Flujos alternativos	Ninguno.
Excepciones	Si no hay espacio en la memoria, no podrá almacenarlo.

Caso de uso	**Cargar un programa**
Actores	Usuario
Tipo	Básico
Resumen	Lee un programa de un fichero.
Precondiciones	Debe existir el fichero.
Postcondiciones	Se dispone del programa almacenado en el fichero.
Flujo Principal	El usuario seleccionará el fichero donde está almacenado el programa y se leerá.
Flujos alternativos	Ninguno.
Excepciones	Puede que el fichero esté corrupto, en cuyo caso puede haber problemas con la lectura o con la ejecución, pudiéndose generar una excepción.

Caso de uso	**Enviar un programa**
Actores	Usuario
Tipo	Básico
Resumen	Se enviará un programa al Brazo Robótico.
Precondiciones	Debe existir el programa.
Postcondiciones	El Brazo Robótico tendrá el programa guardado en la EEPROM.
Flujo Principal	El usuario enviará el programa al Brazo Robótico, que lo almacenará en la EEPROM.
Flujos alternativos	Ninguno.
Excepciones	Ninguno.

Caso de uso	**Crear Orden**
Actores	Usuario
Tipo	Básico, Inclusión
Resumen	Se creará una nueva Orden.
Precondiciones	Ninguna.
Postcondiciones	Se obtendrá una orden correctamente formada.
Flujo Principal	El usuario creará una nueva orden, escogiendo todos sus parámetros.
Flujos alternativos	Ninguno.
Excepciones	Si el estándar está mal definido, pueden haber problemas con la ejecución, pudiéndose generar una excepción.

Caso de uso	Editar Orden
Actores	Usuario
Tipo	Básico
Resumen	Se editará una Orden.
Precondiciones	Debe existir la orden.
Postcondiciones	Se obtendrá la orden modificada.
Flujo Principal	El usuario, tras escoger la orden, la modificará como lo desee.
Flujos alternativos	Ninguno.
Excepciones	Si el estándar está mal definido, pueden haber problemas con la ejecución, pudiéndose generar una excepción.

Caso de uso	Parar los motores
Actores	Usuario
Tipo	Básico
Resumen	Se pararán los motores.
Precondiciones	Ninguna.
Postcondiciones	Los motores del Brazo Robótico se pararán.
Flujo Principal	El usuario pulsará el botón de emergencia para parar los motores del Brazo Robótico. En ese momento, se mandará la orden para paralizarlo.
Flujos alternativos	Ninguno.
Excepciones	Ninguno.

5.3 Requisitos del sistema

El proyecto se puede dividir en dos partes bien diferenciadas, por un lado la aplicación Android y por otro el programa Arduino. Estos subproyectos disponen de distintos requisitos, ya que difieren en las funcionalidades requeridas, por lo que sus requisitos serán tratados por separado.

5.3.1 Requisitos de la Aplicación Android

Con el fin de que la aplicación cumpla con las necesidades básicas del proyecto, debe cumplir una serie de funcionalidades que permitan al usuario realizar las acciones necesarias para su labor con el control del Brazo Robótico:

- Debe permitir controlar manualmente el Brazo Robótico, simulando las funcionalidades de la paleta ya existente
- Debe permitir desarrollar un programa válido para que el Brazo Robótico lo ejecute correctamente

- Debe permitir la comunicación con la memoria interna del Brazo Robótico, para la carga y descarga de programas
- Debe permitir la utilización de distintos estándares para usarla con distintos Brazos Robóticos
- Debe realizar la conexión con los Brazos Robóticos vía Bluetooth

5.3.2 Requisitos del Programa Arduino

Las necesidades para el programa Arduino son más escasas, ya que su funcionalidad se basa en la idea de diseñar un mero puente de comunicación entre las dos plataformas:

- Debe permitir el envío y recepción de mensajes vía Bluetooth
- Debe permitir el envío y recepción de mensajes vía Serial
- Debe controlar el flujo de los mensajes, conectando los puertos Serial con los puertos Bluetooth

5.3.3 Requisitos de la Interfaz de Usuario

Mención aparte requieren las necesidades de la interfaz de usuario, ya que es el único contacto que tendrá el usuario con nuestra aplicación. Dicha interfaz debe estar adaptada para que sea fácilmente entendible por cualquier usuario, independientemente de su nivel de conocimientos.

La interfaz de usuario debe permitir hacer uso de manera clara y sencilla todas las funcionalidades de la aplicación. Para ello, se ha desarrollado una interfaz intuitiva donde se podrán hacer uso de las funcionalidades básicas y definir las instrucciones directamente. Por lo tanto es necesario que todos los casos de usos definidos para el Usuario puedan ser ejecutados de manera gráfica. La organización de los elementos en la pantalla debe seguir una línea de ordenación intuitiva, para que el esfuerzo del usuario para adaptarse sea mínimo.

Por otro lado, han de distinguirse claramente los modos de ejecución con el fin de evitar confusiones a la hora de controlar directamente el Brazo Robótico o de crear un programa. Finalmente, se debe permitir el acceso al sistema de archivos con facilidad, así como permitir definir la conexión vía Bluetooth sin necesidad de grandes dificultades.

6. Diseño

6.1 Interfaz de Usuario

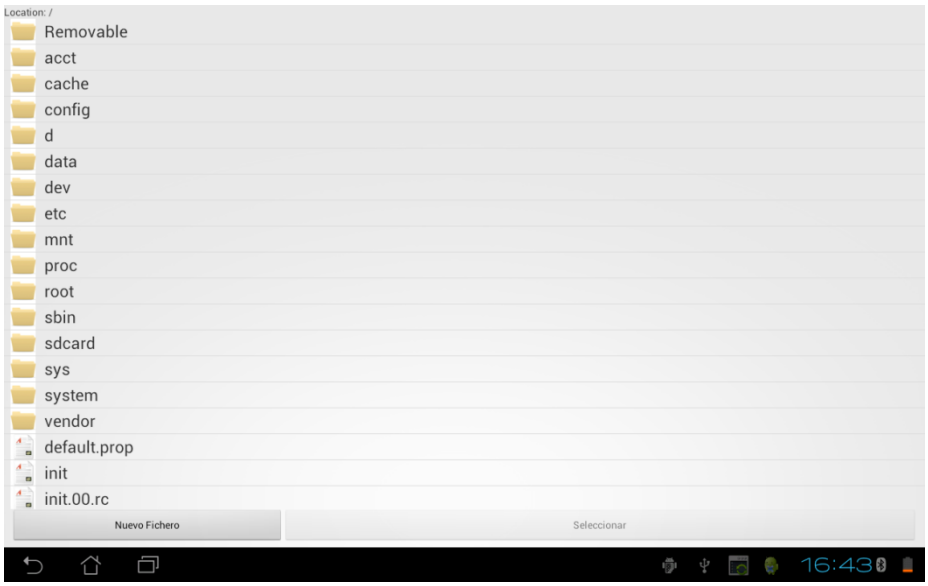
Con el fin de conseguir una mejor experiencia para el usuario al utilizar la aplicación, se ha definido una interfaz basada en botones, simplificando y ocultando las operaciones más complejas o arduas al usuario. Por otro lado, se han definido distintas pantallas con una funcionalidad bien definida, separando los distintos elementos entre sí.

6.1.1 Pantalla Principal

La pantalla principal es la primera pantalla que se muestra al usuario. En esta pantalla, el usuario podrá escoger entre los tres distintos modos de ejecución.

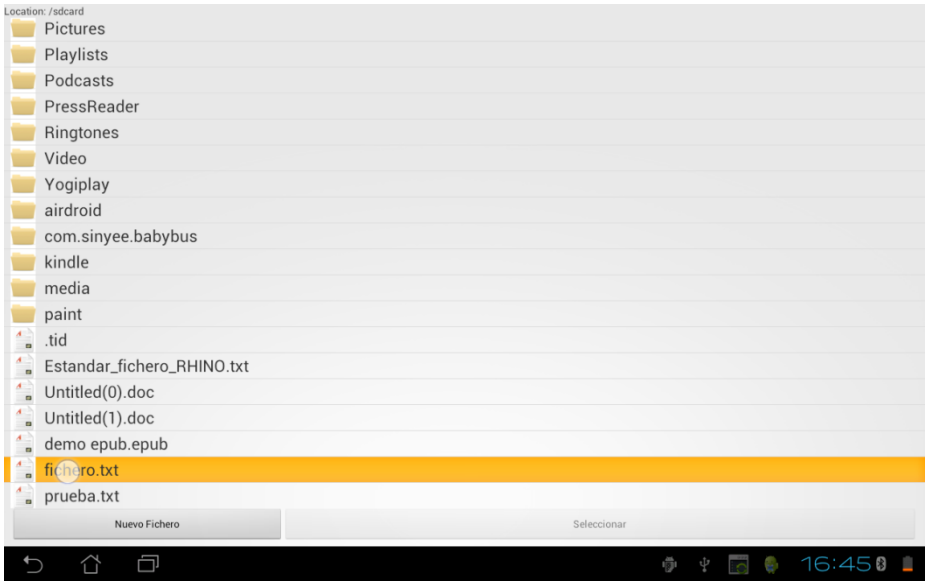
6.1.2 Selección de Fichero

A la hora de escoger un fichero para conocer su ruta, veremos la siguiente interfaz:



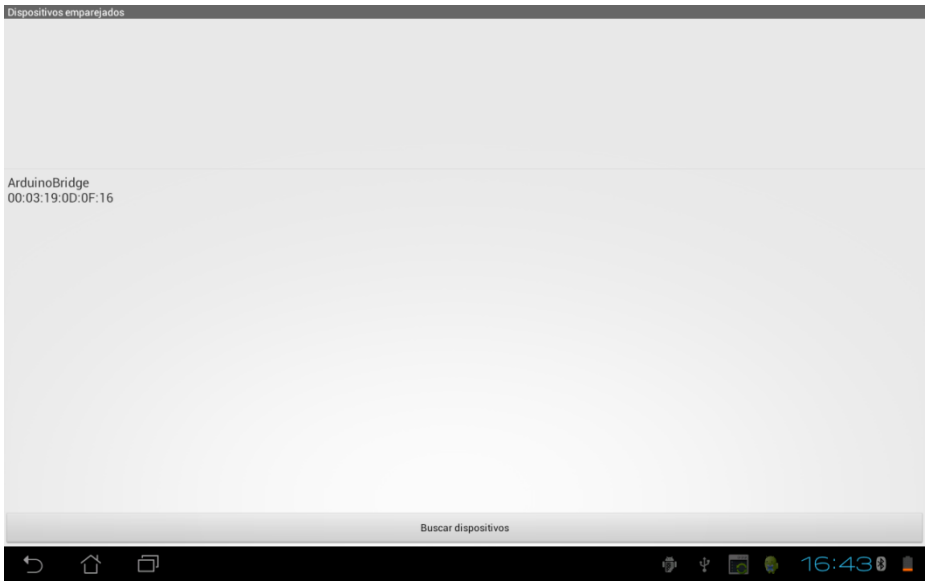
En esta interfaz, el usuario podrá navegar por el sistema de archivos, pudiendo entrar en los distintos directorios pulsando en ellos. Si se desea crear un nuevo fichero, basta con pulsar el botón *Nuevo Fichero* para que se muestre un pequeño cuadro de texto para insertar el nombre del fichero deseado.

En este punto, puede crearlo o cancelar, en cuyo caso podrá seguir navegando. Si desea seleccionar un fichero ya creado, basta con pulsar sobre el fichero que deseamos y escoger la opción *Seleccionar*.

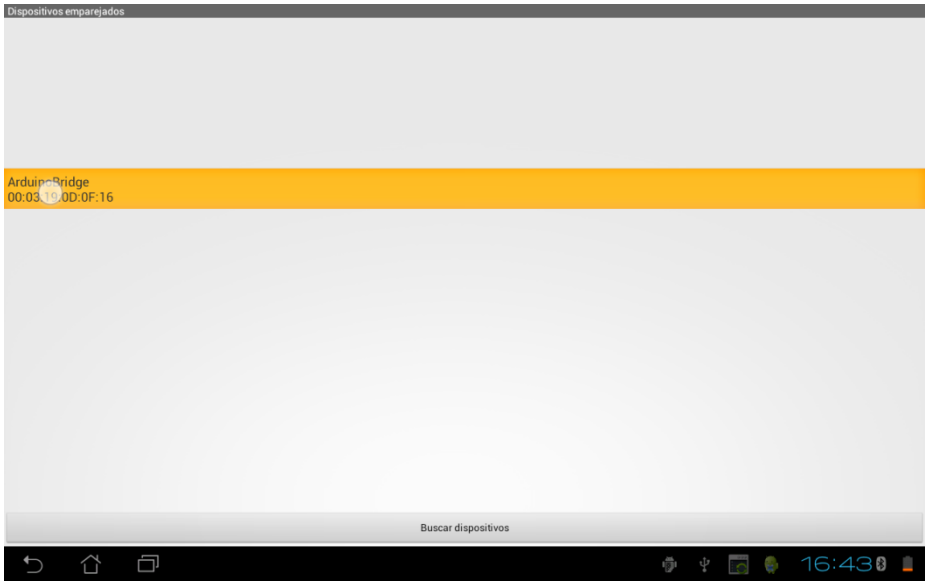


6.1.3 Selección de un dispositivo

Para seleccionar un dispositivo para conectarnos vía Bluetooth, se mostrará una ventana con los distintos dispositivos a los que ya nos hemos conectado. También hay un botón donde podemos obtener todos los dispositivos cercanos que sean visibles para nuestra antena Bluetooth.

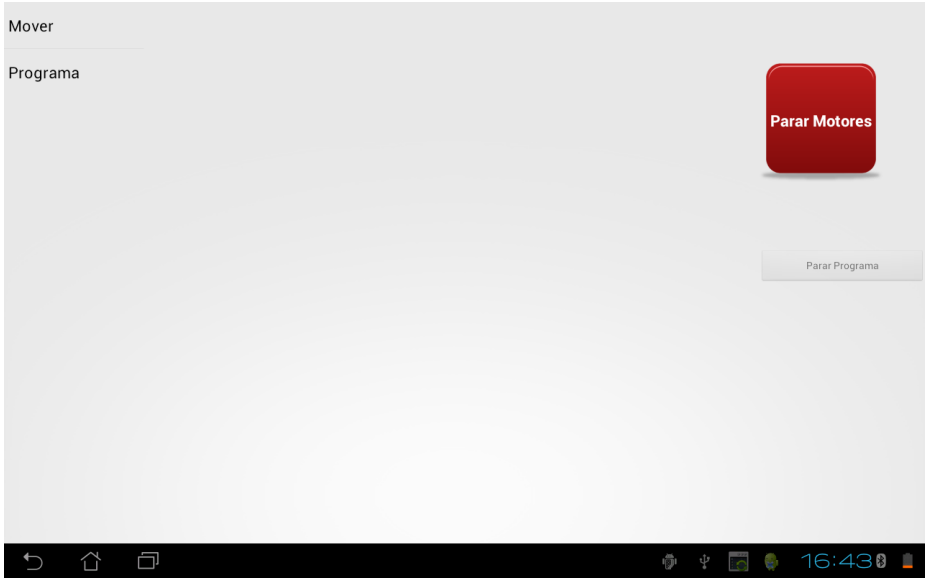


Para seleccionar un dispositivo, basta con pulsar en él para intentar conectarnos a él. Si la aplicación no consigue conectarse, volverá a mostrar los dispositivos disponibles.



6.1.4 Modo Paleta

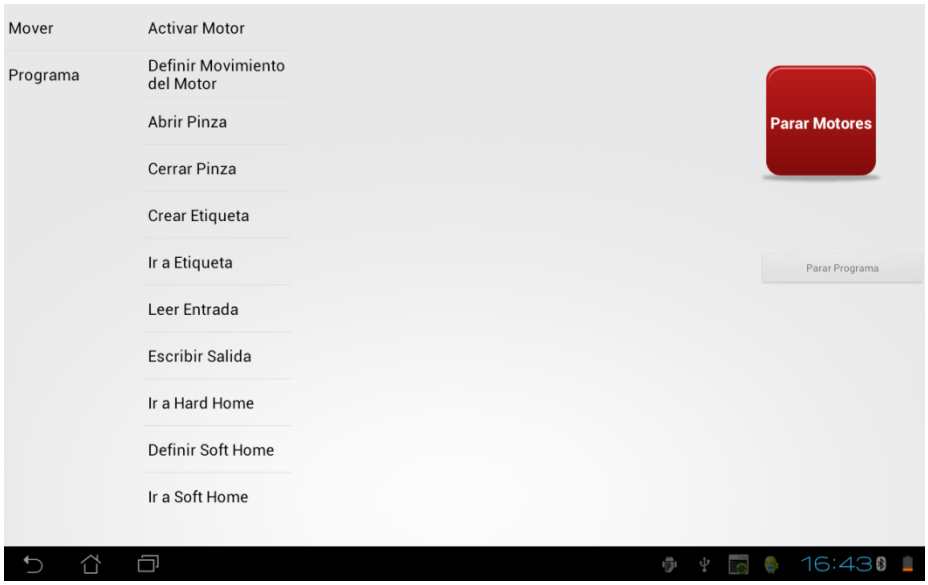
La pantalla del "Modo Paleta" muestra los elementos necesarios para un correcto control del Brazo Robótico de forma interactiva manual, emulando a la "paleta" física conectada al controlador del robot.



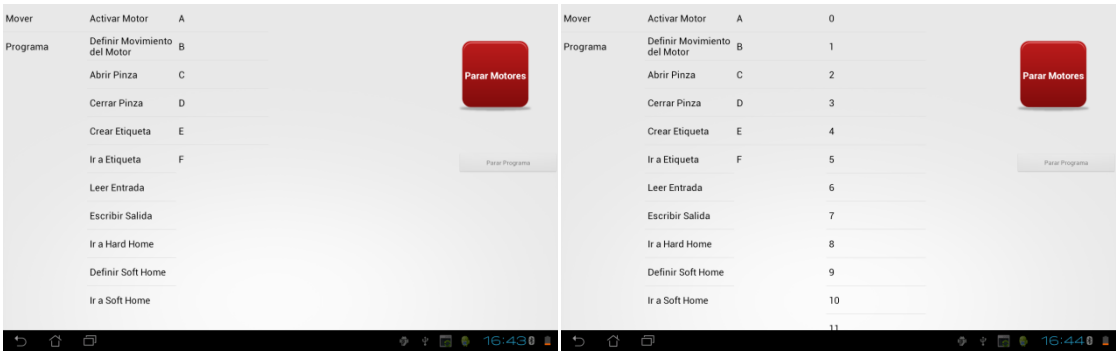
Como se puede ver, hay cuatro elementos principales:

- Un botón de emergencia para parar los motores del Brazo Robótico
- Un botón de parada de rutina
- Un conjunto de órdenes para el movimiento del Brazo Robótico
- Un conjunto de órdenes para el control de la memoria EEPROM del controlador del Brazo Robótico

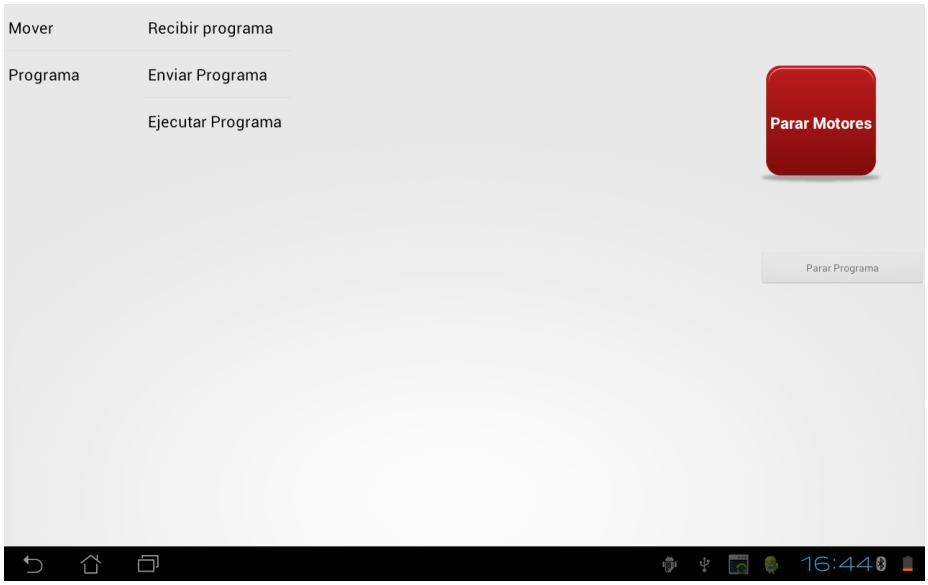
De estos elementos, hay que comentar los dos conjuntos de órdenes, *Mover* y *Programa*. El conjunto de órdenes *Mover* almacena todas las ordenes que implican un control directo del Brazo Robótico y de las entradas y salidas digitales del sistema.



Algunas órdenes disponen de varios parámetros, por lo que al pulsarlas, se mostrará el siguiente parámetro a escoger. La orden se enviará al controlador una vez que se pulse el último parámetro (o, en el caso que no tenga parámetros, cuando se pulse la orden).



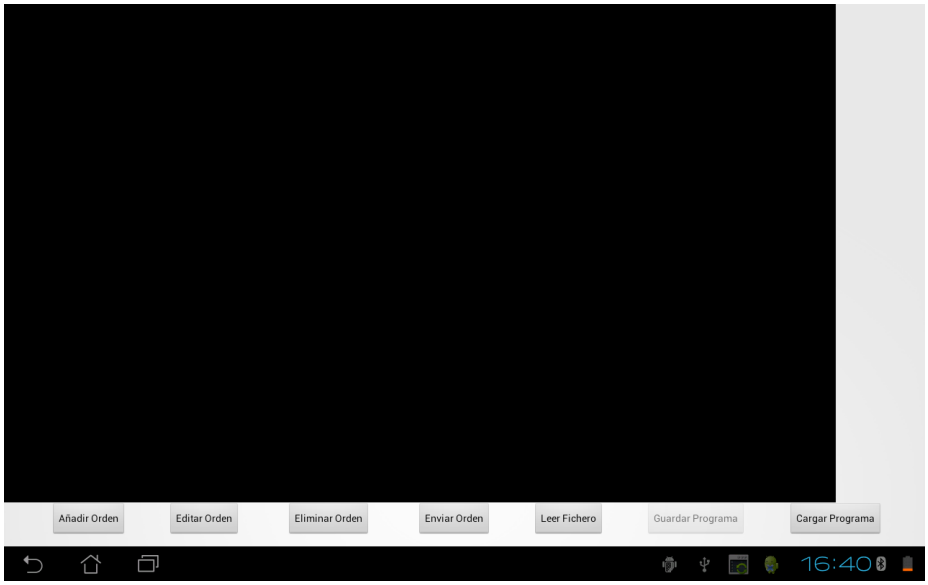
Por otro lado, el conjunto de órdenes *Programa* almacena todas las ordenes que interactúan con la memoria EEPROM del controlador, ya sea enviando, recibiendo o ejecutando rutinas del mismo.



6.1.5 Control Programado

La interfaz del Control Programado busca facilitar la labor de creación y edición de una rutina apta para el controlador del Brazo Robótico. Para ello, dispone de los siguientes elementos:

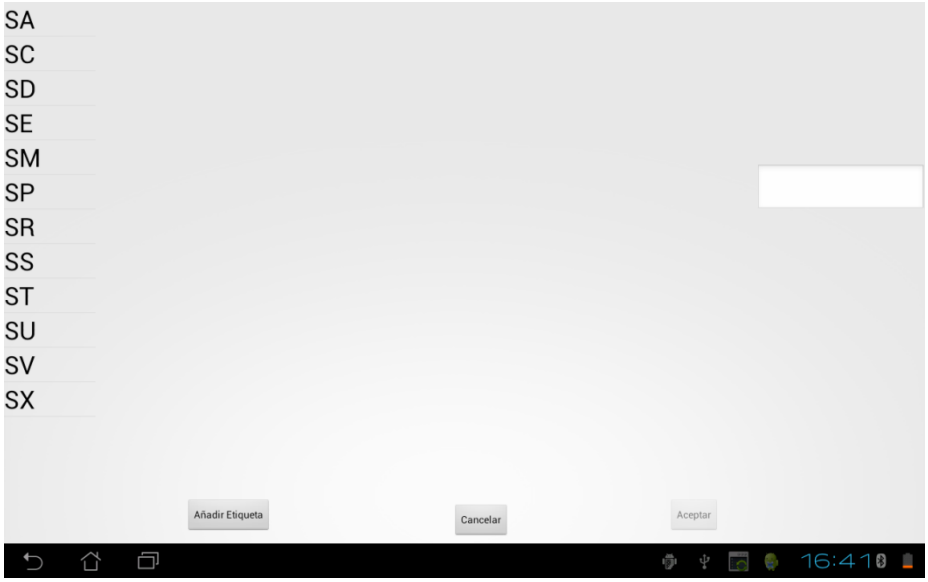
- Botón de inserción de orden
- Botón de edición de orden
- Botón de eliminación de orden
- Botón de envío de órdenes
- Botón de lectura de fichero
- Botón de almacenamiento en un fichero
- Botón de recepción de una rutina
- Rutina actual



A medida que vayamos insertando órdenes, éstas se irán mostrando en la franja gris de la derecha. Si se desea editar o eliminar una orden, basta con pulsar en la orden en cuestión y escoger la opción con los botones ubicados en la parte inferior de la interfaz.

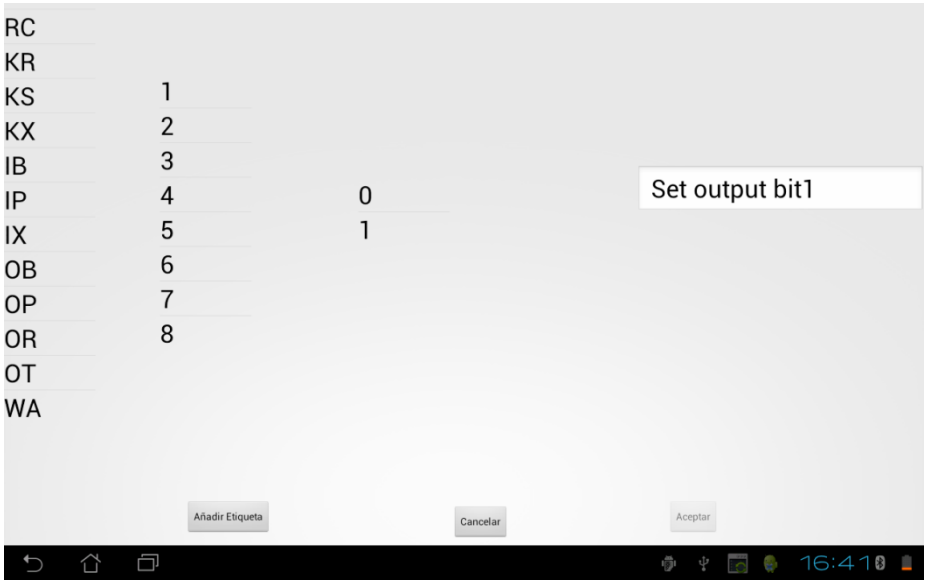
6.1.6 Selección de Orden

Para seleccionar una orden y sus distintos parámetros, veremos una lista con todas las órdenes disponibles a la izquierda de la interfaz.



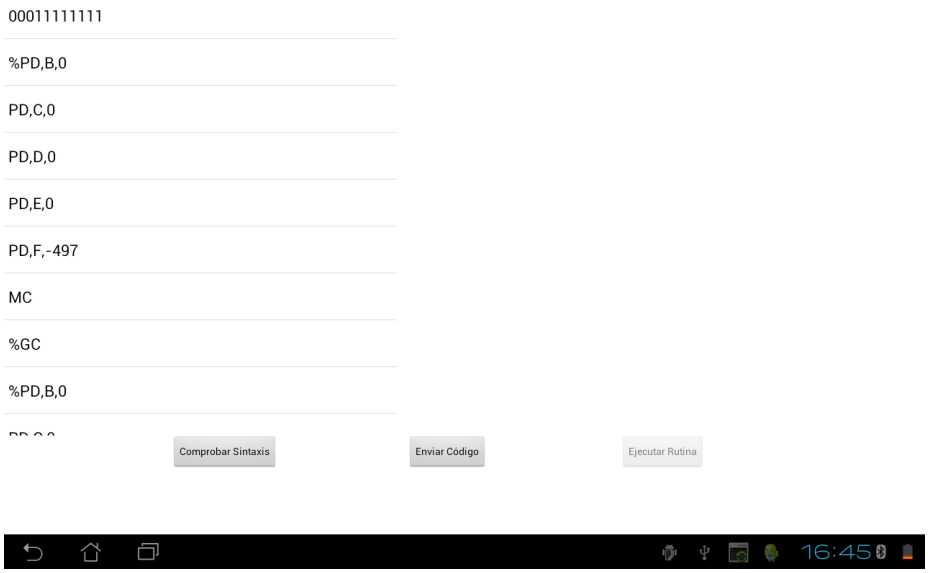
Una vez que se pulse en la orden deseada, se mostrarán todos los posibles valores del primer parámetro (si lo tiene) y un pequeño comentario explicando la funcionalidad de la orden.

Cada vez que el usuario pulse en un parámetro, se mostrarán los valores del siguiente parámetro. Una vez que haya seleccionado el último parámetro, se habilitará el botón *Aceptar*. Además, desde el principio estará disponible el botón *Añadir Etiqueta*, que permite incluir una marca que indique la posición de una etiqueta, y *Cancelar*, que vuelve al estado anterior sin devolver ninguna orden.



6.1.7 Ejecutar Rutina

La interfaz para el envío y ejecución de una rutina mostrará inicialmente el código de la rutina la izquierda de la interfaz. Una vez cargado el código, el usuario puede pulsar el botón de *Comprobar sintaxis* y, si no hay ningún fallo, pulsar en *Enviar Código* para enviar la rutina al controlador. Tras enviar la rutina, por fin se podrá pulsar en *Ejecutar Rutina* para ejecutar la rutina almacenada en la memoria EEPROM.



6.2 Diagrama de Clases de la Aplicación

Con el fin de diseñar las clases necesarias, hemos de tener en cuenta que se está empleando el lenguaje Java, por lo que se utilizan los paquetes. Estos paquetes contienen aquellas clases cuya distancia es menor, es decir, aquellas que más cercanas sean. En este caso, se han creado los siguientes paquetes que contienen todas las clases creadas en el proyecto:

- Activities
- Bluetooth
- Utilidades

La estructura de las clases puede verse en el diagrama de Clases. En primer lugar, el paquete Activities contiene las actividades principales del sistema, es decir, aquellas actividades[15] que representan los modos de funcionamiento de la aplicación, siendo el núcleo del sistema. El segundo paquete, Bluetooth, contiene las clases relacionadas con la conexión Bluetooth del sistema, donde se incluyen todas las funciones para la gestión de la conexión. Por último, el paquete Utilidades, que incluye aquellas clases necesarias para gran parte del trabajo del sistema, como la obtención de la ruta de los ficheros o el almacenamiento de los datos necesarios del robot, pero que no son el núcleo del mismo.

6.2.1 Activities

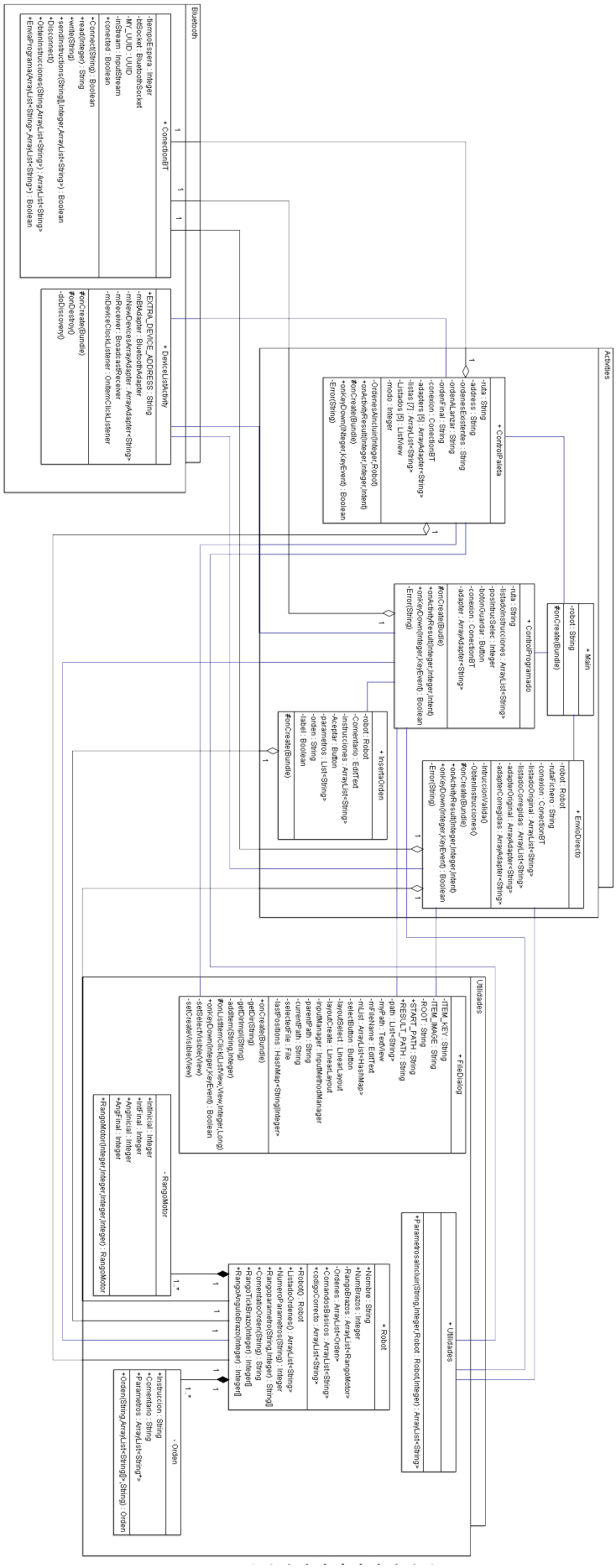
El paquete Activities contiene las clases básicas del sistema, es decir, el núcleo funcional del mismo. Estas clases son las actividades que lanza el Sistema Operativo Android para que las aplicaciones funcionen y que sustentan la mayor carga de trabajo del proyecto. Las actividades que contiene este paquete son:

6.2.1.1 Main

La actividad Main es la primera actividad en ser lanzada al abrir la aplicación. Su funcionalidad es básica, simplemente muestra los elementos necesarios para entrar en los tres modos de ejecución básicos de la aplicación.

[15] En el sistema operativo Android, las actividades son un tipo de clase que representa a los hilos de ejecución de las aplicaciones.

Diagrama de Clases



6.2.1.2 ControlPaleta

La actividad ControlPaleta es la actividad donde se permite un control directo del robot desde la tableta. En esta clase se cargan las órdenes básicas del robot y permite trabajar con ellas, desde controlar el robot directamente hasta interactuar con su memoria EEPROM, descargando o enviando programas. El manejo de las órdenes se hace de forma dinámica, es decir, a medida que el usuario va escogiendo las órdenes, sus parámetros se desbloquean a la derecha de la orden, permitiendo escoger los parámetros sin interferencias.

Para poder llevar a cabo la comunicación con el dispositvo, contiene una conexión Bluetooth gracias a la clase ConectionBT, la cual inicializa obteniendo la dirección con la clase DeviceListActivity. Por otro lado, contiene a la clase Robot, necesaria para conocer las ordenes ejecutables. Por último, hay que comentar que llama a la clase FileDialog para poder obtener las rutas de los ficheros necesarios.

6.2.1.3 ControlProgramado

En esta actividad, el usuario puede crear o modificar un programa. Aquí, el usuario irá añadiendo o modificando órdenes gracias a la clase InsertaOrden, que devolverá la instrucción completamente formada que el usuario desee, leyendo o almacenando en ficheros obtenidos con la clase FileDialog y enviándolos o recibiéndolos por Bluetooth gracias al paquete Bluetooth. Todo esto permitirá al usuario desarrollar un programa nativo del robot sin necesidad de emplear otros dispositivos.

6.2.1.4 InsertaOrden

La actividad InsertaOrden es la actividad donde se puede generar una orden válida siguiendo los parámetros definidos en el estándar del robot. Para ello, obtendrá las órdenes disponibles de la clase Robot y mostrará mediante una lista en la pantalla. Cada vez que el usuario seleccione una orden, se le mostrarán los parámetros disponibles de forma ordenada, es decir, cuando seleccione, el primer parámetro, se le mostrará el segundo, si existe, y así consecutivamente. Una vez que la orden esté completa, si el usuario acepta, devolverá la orden formada.

6.2.1.5 EnvioDirecto

Por último, la clase EnvioDirecto permite al usuario enviar un programa ya hecho a la EEPROM del robot, además de poder analizar su correctitud respecto al estándar definido. Para ello, se conecta vía Bluetooth gracias al paquete Bluetooth, carga el fichero obtenido de la clase FileDialog en pantalla y, si el usuario lo desea, lo valida comparando cada orden con el fichero estándar. Esto añade una etapa previa al envío con el fin de garantizar la sintaxis de las órdenes e impedir que se envíen órdenes que puedan causar errores en el robot. Una vez completado el paso anterior, puede enviarse al Brazo Robótico.

6.2.2 Bluetooth

6.2.2.1 ConectionBT

La clase ConectionBT es la clase que gestiona el enlace Bluetooth, desde la conexión hasta la desconexión con el dispositivo escogido, pasando por el envío y recepción de datos con el mismo. Dispone de funciones para enviar tanto ordenes sueltas como programas completos, además de permitir la recepción del código almacenado en la EEPROM para su posterior tratamiento o almacenamiento.

6.2.2.2 DeviceListActivity

La segunda clase del paquete, DeviceListActivity, se encarga de obtener y mostrar los dispositivos a los que la aplicación puede conectarse y conseguir la dirección de aquel dispositivo al que el usuario desee conectarse. Además, discrimina entre aquellos dispositivos ya enlazados (que muestra inicialmente) con los nuevos dispositivos descubiertos, para los cuales hay que realizar una búsqueda.

6.2.3 Utilidades

6.2.3.1 FileDialog

La primera clase con la que nos encontramos en este paquete es FileDialog. Esta clase gestiona la obtención de las rutas de cada uno de los ficheros necesarios para el funcionamiento de la aplicación. Permite tanto obtener la ruta de ficheros ya existentes como

de definir el nombre de nuevos ficheros para su posterior creación, todo esto mientras facilita la navegación por el sistema de archivos.

6.2.3.2 Utilidades

Esta clase se definió como clase para disponer de aquellas funcionalidades que fuesen requeridas por múltiples clases, evitando así que no se vieran duplicadas en el código. Finalmente, esta clase almacena la función ParametrosaIncluir, que devuelve una lista de elementos que contiene todos los posibles valores del parámetro dado para la orden escogida.

6.2.3.3 Robot

Esta clase almacena y refleja los valores definidos en el estándar del robot. Contiene tanto las órdenes del sistema como los rangos de los distintos motores del brazo robótico, inclusive el nombre del brazo robótico y el número de brazos que dispone. Además, almacena aquellas órdenes básicas para el funcionamiento de la paleta de forma separada. Esto facilita su acceso para el control en el modo Paleta.

6.2.3.4 RangoMotor

Esta clase almacena el rango de un motor, tanto en valores enteros como angulares, siempre y cuando estén definidos en el estándar. En caso contrario, se almacenarán los valores "-1,-1", indicando que no dispone de los mismos.

6.2.3.5 Orden

Por último, se encuentra la clase Orden, la cual almacena los datos de cada orden, desde la propia orden, hasta una breve explicación de la misma, pasando por sus parámetros. Éstos serán una lista con los rangos de los mismos si no son dependientes del motor, o los códigos "IRANGE" o "ARANGE" si dependen de los mismos, siendo el primero para los valores enteros y el segundo para los valores angulares.

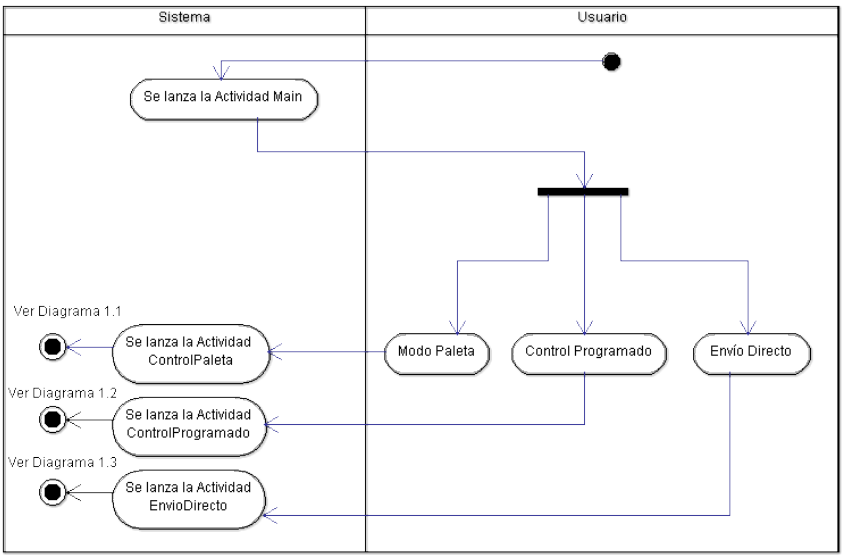
6.3 Diagrama de Actividades de la Aplicación

La complejidad del sistema, a simple vista, parece mínima, ya que está planteada bajo un diagrama sencillo. Pero esto no es así, ya que el sistema permite generar bastante trabajo y muy variado con pocos movimientos, por lo que el diagrama de Actividades se ve incrementado de forma elevada y por ello se ha subdivido en el control de flujo de cada clase principal, facilitando su presentación y comprensión.

6.3.1 Main

Al iniciarse la actividad, se prepara la interfaz para que el usuario pueda escoger el modo que desea ejecuta entre los tres disponibles, como muestra el Diagrama 1.0.

Diagrama 1. 0 - Main

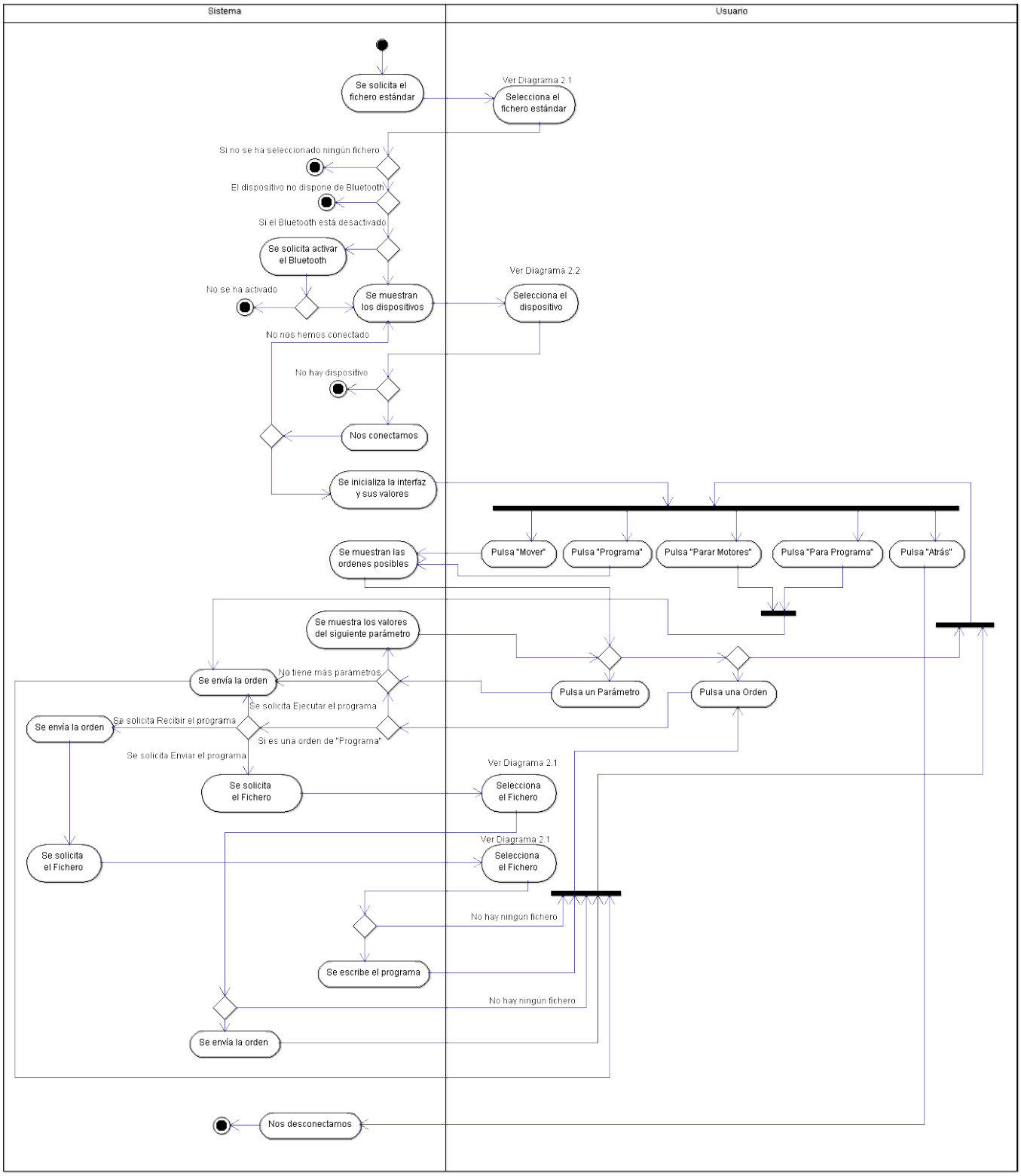


6.3.2 ModoPaleta

En la ejecución del Modo Paleta se aprecia que inicialmente hay un intercambio de información entre el usuario y el sistema, definiendo la ruta del estándar del robot y la dirección del dispositivo al que vamos a conectarnos. Si se produjera un fallo en este proceso, como podría ser no seleccionar un fichero o que el sistema no disponga de Bluetooth, se volverá a la actividad anterior, es decir, Main.

Una vez hecho esto, y si no ha ocurrido ningún problema, el usuario puede controlar el brazo robótico como desee (dentro de las limitaciones del funcionamiento del mismo). En este punto se puede desde mover el brazo robótico, utilizando las órdenes disponibles, como trabajar con la EEPROM del brazo robótico, enviando o recibiendo un programa. Además, puede parar los motores o el programa en ejecución como medida de emergencia. Si retrocede volverá a la actividad Main.

Diagrama 1. 1 - Modo Paleta



6.3.3 Control Programado

En esta actividad se puede apreciar que la inicialización es similar que en el Modo Paleta pero en cambio, una vez que finaliza, hay muchas más opciones para el usuario a la hora de trabajar. Se podrá añadir instrucciones o modificarlas, gracias a la actividad InsertaOrden; eliminar las órdenes escogidas o leer un fichero de la memoria de la tableta. Por otro lado, puede interactuar con la memoria EEPROM del brazo robótico, obteniendo el programa que está en el mismo o cargando un propio. Si se desea volver a la actividad principal, basta pulsar el botón de retroceso.

6.3.4 EnvioDirecto

En esta clase, además de realizar los mismos pasos que en las anteriores actividades en la inicialización, se solicita la ruta del fichero para cargar el programa. Una vez cargado todo el programa, el usuario dispone de tres opciones: comprobar la sintaxis del programa con respecto al estándar del robot, enviar el programa al robot, o retroceder a la actividad Main.

Diagrama 1. 2 - Modo Programado

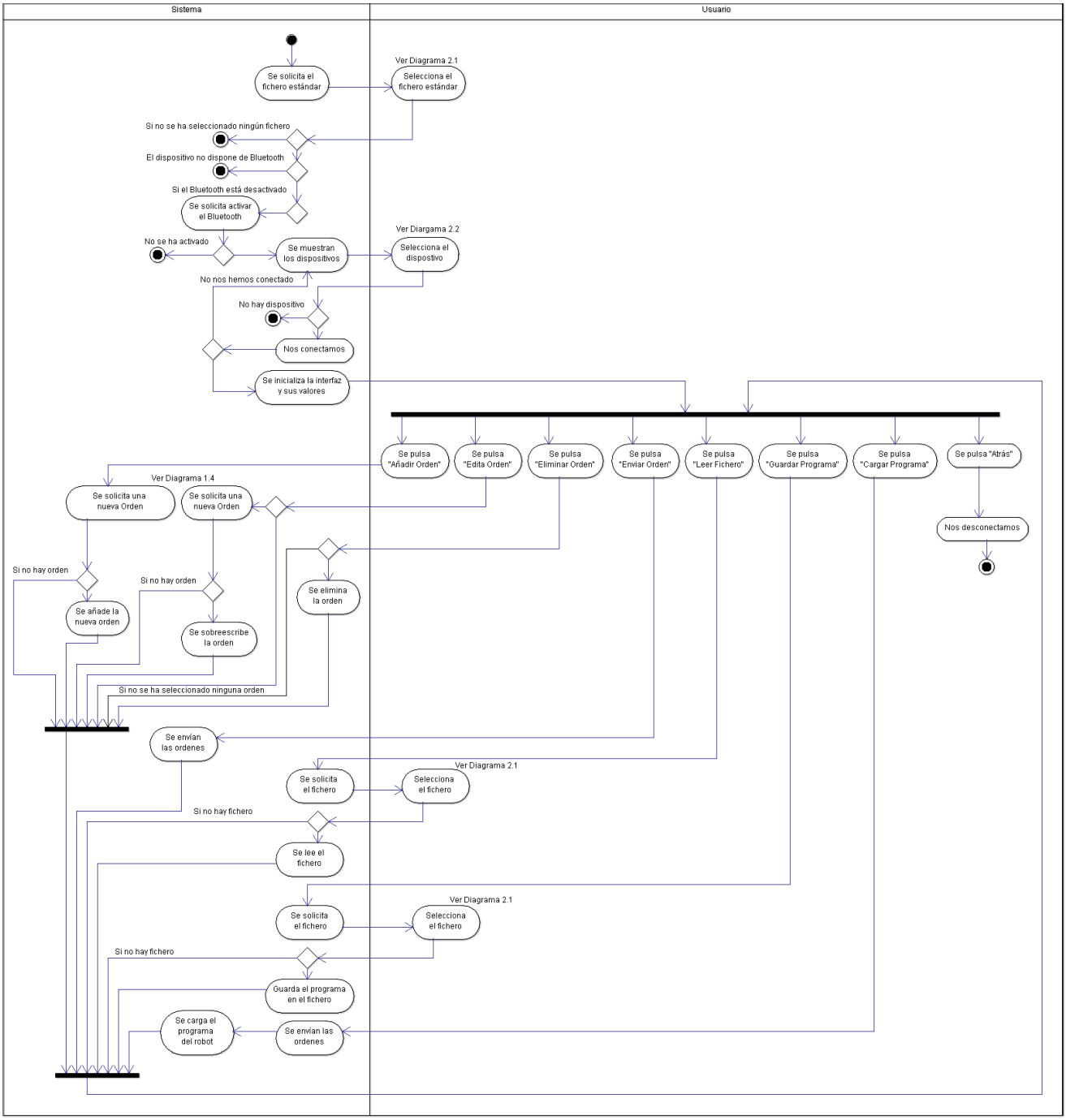
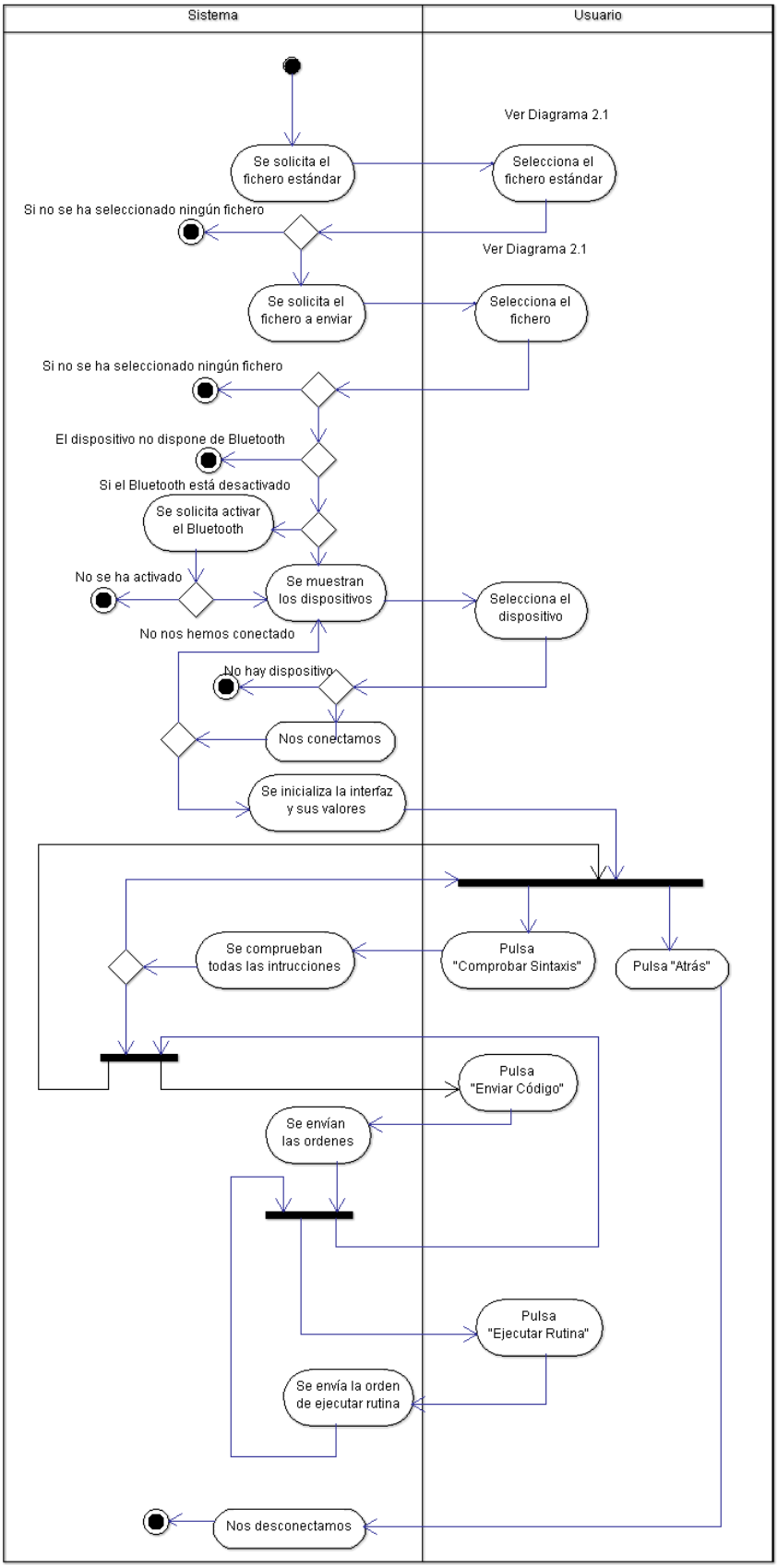


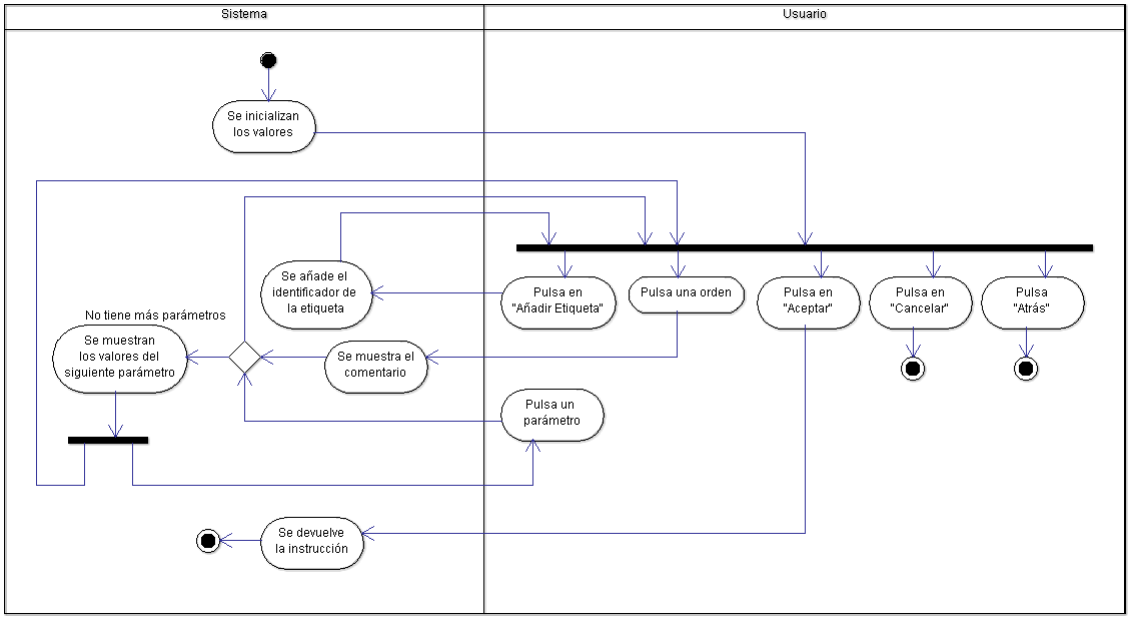
Diagrama 1. 3 - Envío Directo



6.3.5 InsertaOrden

En la clase InsertaOrden, el usuario puede definir los parámetros de una orden y, si está satisfecho con la misma, devolverla al proceso anterior, pero si el resultado no es el deseado, puede rehacerla sin ningún coste computacional, simplemente seleccionando otra orden u otro parámetro. Para ir rellenando la instrucción, se debe ir escogiendo cada parámetro en orden, y se le irá mostrando cada parámetro una vez que se haya seleccionado el parámetro anterior. Por último, retrocediendo o Cancelando se vuelve al proceso anterior sin devolver ninguna orden.

Diagrama 1. 4 - InsertaOrden

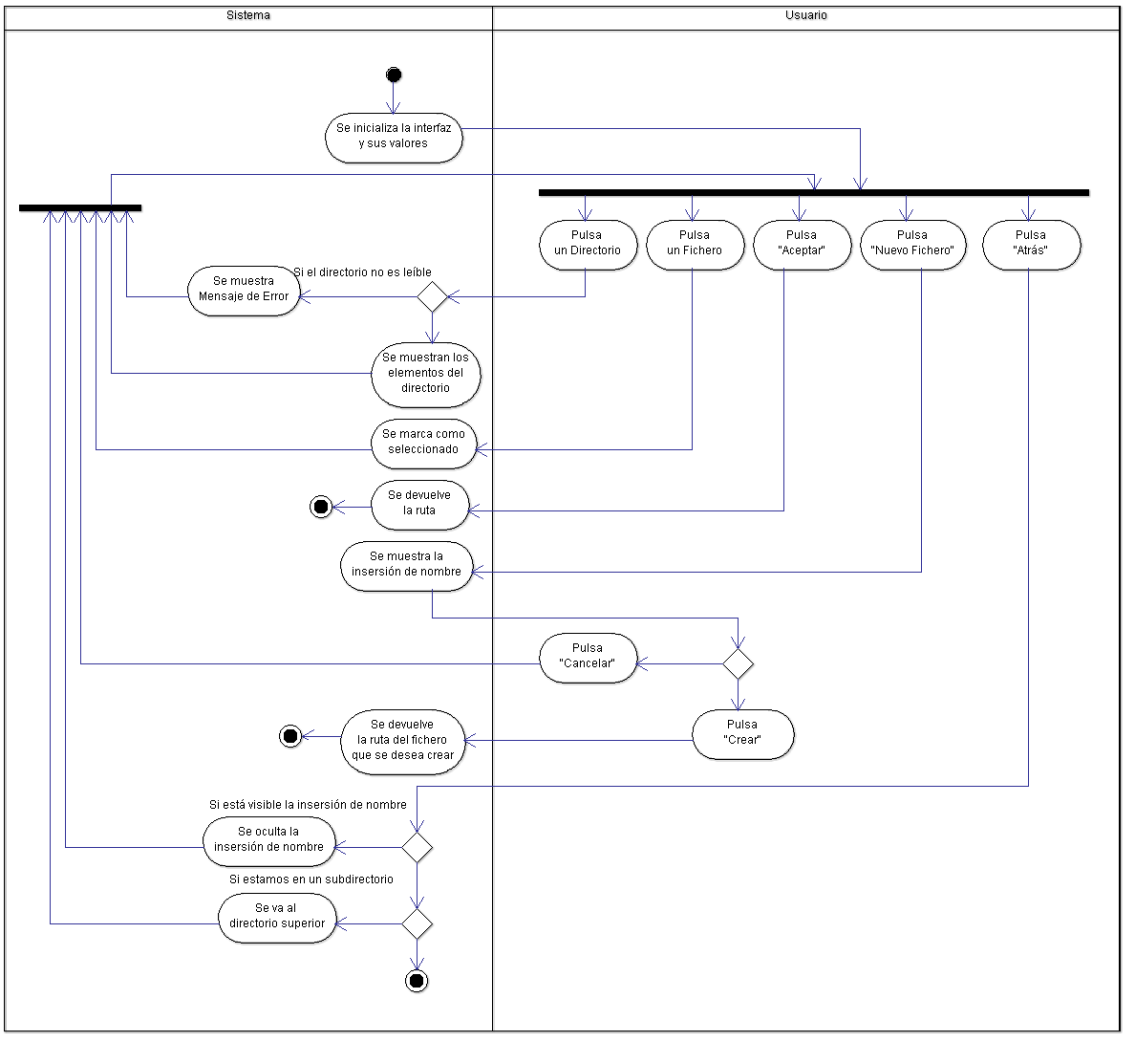


6.3.6 FileDialog

En la actividad FileDialog, el usuario puede navegar por el sistema de archivos mientras busca el fichero que desea seleccionar o el directorio donde desea almacenar el fichero. Si desea crear un nuevo fichero, pulsando el botón de creación se le muestra un cuadro de texto para indicar el nombre del fichero.

Una vez escrito, puede aceptar, en cuyo caso se devolverá la ruta del fichero que deseamos crear, o bien puede cancelar, en cuyo caso no se devolverá ninguna ruta. Si en lugar de esto, lo que desea es seleccionar un fichero ya existente, basta con que lo seleccione y pulse aceptar. En cambio, si pulsa el botón de retroceso, irá subiendo en el árbol de directorios hasta la carpeta raíz, en cuyo caso, si seguimos intentando retroceder, volverá a la actividad que lo lanzó.

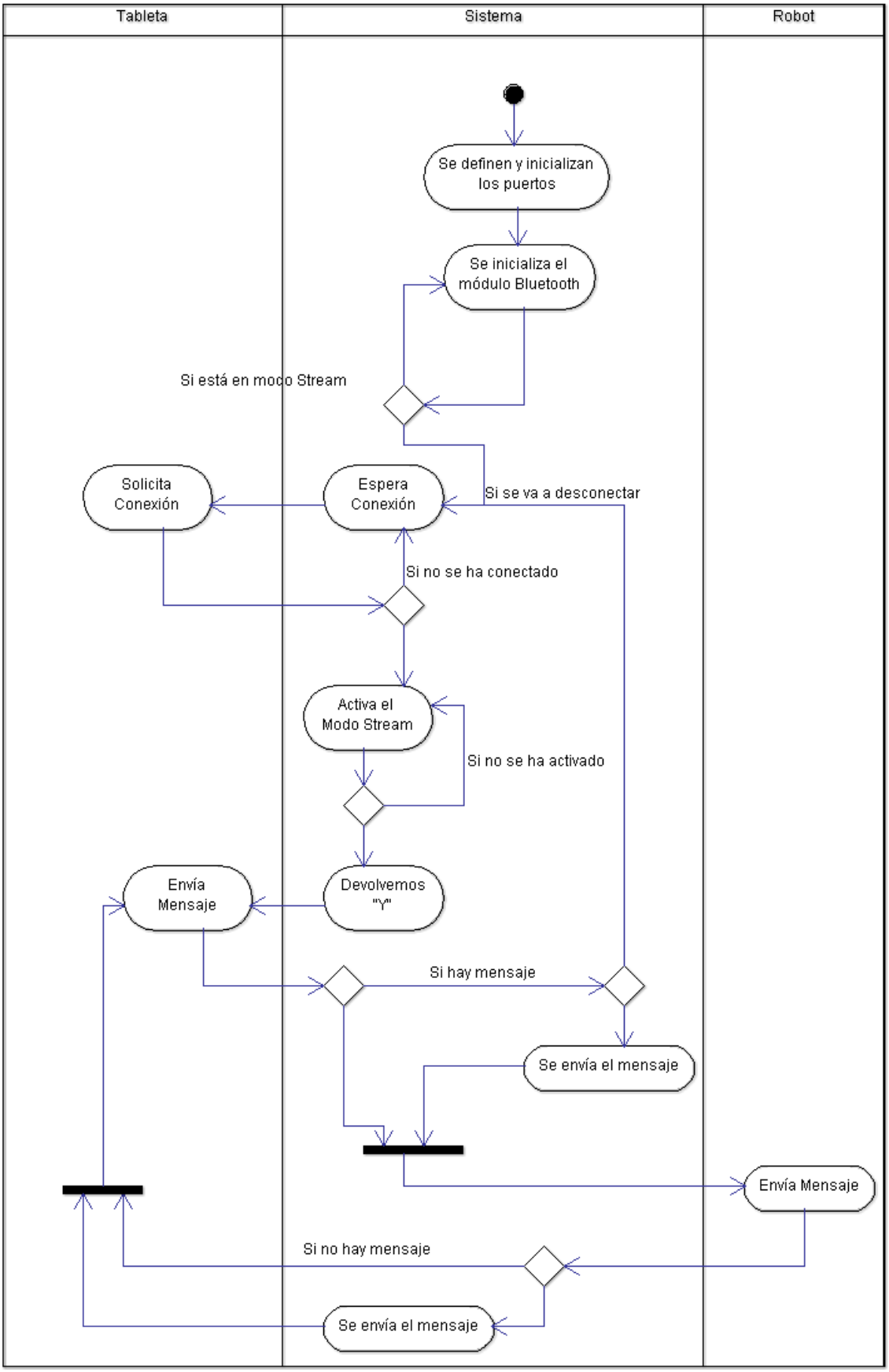
Diagrama 2. 1 - FileDialog



6.4 Diagrama de Actividades del Arduino

El programa interno del Arduino se encarga del control de tráfico entre la tableta, vía Bluetooth, y el Brazo Robótico, vía cable Serial. Para ello, inicialmente prepara al módulo Bluetooth para recibir una conexión y, una vez recibida, actúa como puente entre ambas plataformas, con la salvedad que, si recibe el carácter '#' (en ASCII es el número 35 en decimal), indica que la tableta va a desconectarse, por lo que prepara al módulo Bluetooth para recibir otra conexión.

Diagrama Arduino



7. Implementación

La implementación de la aplicación se ha desarrollado en Java, utilizando las características que disponen los lenguajes de programación orientados a objetos. Por otro lado, el programa de Arduino se ha desarrollado en el lenguaje nativo del entorno de desarrollo de Android, que está basado en Wiring[16] para la programación a bajo nivel. Si bien se puede desarrollar utilizando programación a bajo nivel, este lenguaje dispone de herramientas que facilitan drásticamente el trabajo.

El código fuente del proyecto completo, tanto el código de la aplicación de Android como el código del programa para la placa Arduino están adjuntados con el disco entregado junto a esta memoria.

7.1 Herramientas utilizadas

7.1.1 Android Studio

La implementación del proyecto se ha realizado utilizando el entorno Android Studio, debido a las grandes posibilidades que dispone, al tener un monitor nativo que facilita la depuración y revisión del código. Este entorno aún no está completo y está en constante revisión y actualización, pero aún así, es muy superior a su competidor directo, Eclipse, tanto en prestaciones como en rendimiento.

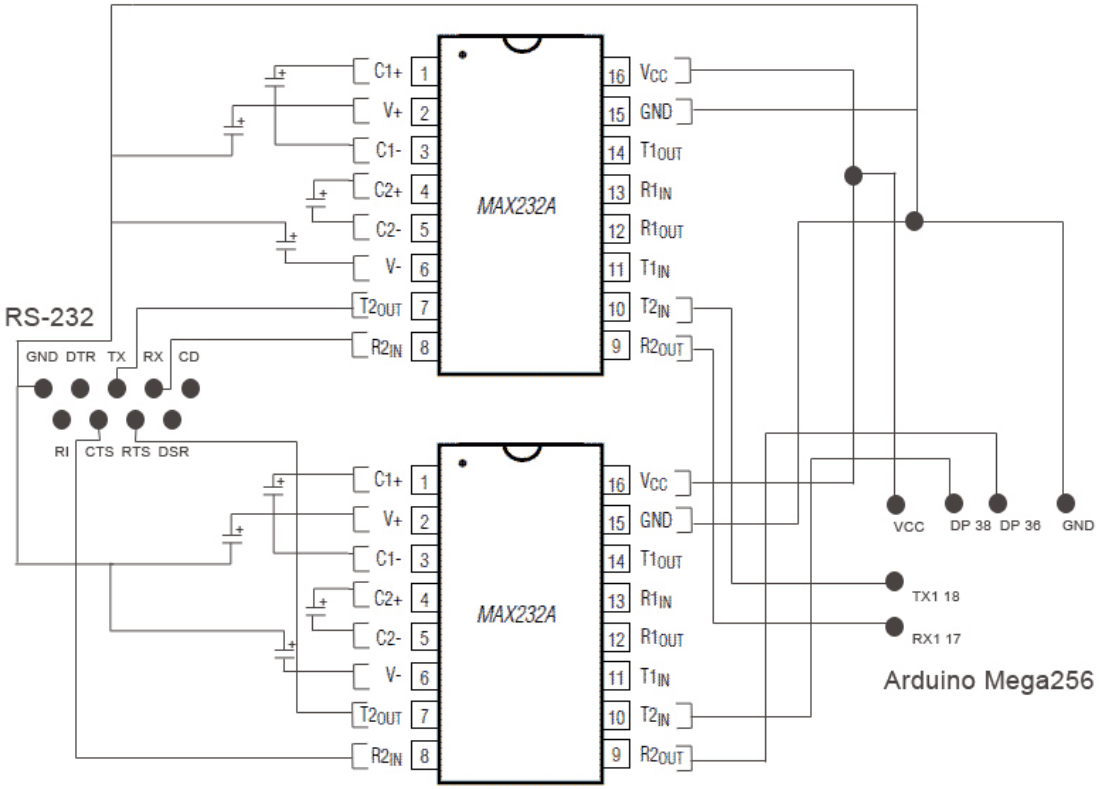
7.1.2 Arduino for Mac

El desarrollo del programa Arduino se ha llevado a cabo en el entorno de desarrollo oficial. Aunque, como sucede con el entorno Android Studio, está en constante actualización, se ha decidido no actualizarlo, por lo que se ha trabajado siempre con la versión 0016.

[16] Wiring es una plataforma de prototipos electrónicos open source compuesta por un lenguaje de programación, un entorno de desarrollo integrado y un microcontrolador. Comenzó su andadura en el 2003, por Hernando Barragán.

7.2 Adaptación de Voltaje en el canal serie

Una vez que se van a realizar las interconexiones del Brazo Robótico con la placa Arduino, se ha tenido que tener en cuenta que ambos sistemas trabajan con diferentes tensiones, trabajando la placa Arduino con voltajes comprendidos entre los 0 y los 5V, y el cable serial RS-232 conectado al Brazo Robótico con voltajes comprendidos entre los -12V y los 12V. Para compatibilizar los niveles de tensiones, se han utilizado los chips MAX232A, que convierten las señales de un puerto Serie RS-232 a señales compatibles con los niveles TTL de circuitos lógicos, que son los niveles equivalentes a los utilizados en las señales del Arduino. Para ello, el circuito que se ha implementado corresponde al siguiente esquema:



En el esquema se puede apreciar que las conexiones entre el cable serial RS-232 y la placa Arduino están unidas mediante dos chips MAX232A, uno para el paso de mensajes (TX y RX en el RS-232 y TX1 y RX1 en el Arduino) y otro para el control de flujo (CTS y RTS en el RS-232 y DP 36 y DP 38 en el Arduino). Se ha optado por esta solución de forma experimental para facilitar las conexiones del cableado y aumentar la flexibilidad, debido al bajo coste de los componentes, si bien en el diseño final se podría realizar, evidentemente, con un solo chip. Para que tenga un correcto funcionamiento, ha sido necesario añadir cuatro condensadores a

cada chip y conectarlos como se aprecia en el esquema. Se han utilizado, en cada chip, cuatro condensadores de un microfaradio, siguiendo las especificaciones del fabricante.

Por último, es necesario precisar que los puertos han sido codificados en el código fuente de la aplicación, por lo que habría que modificarlo si se desease utilizar otros, así como si se busca utilizar una placa distinta a la utilizada, la Arduino Mega 2560.

8. Pruebas y Resultados

En este capítulo veremos las pruebas realizadas con la aplicación y su entorno, analizando los resultados obtenidos en las mismas.

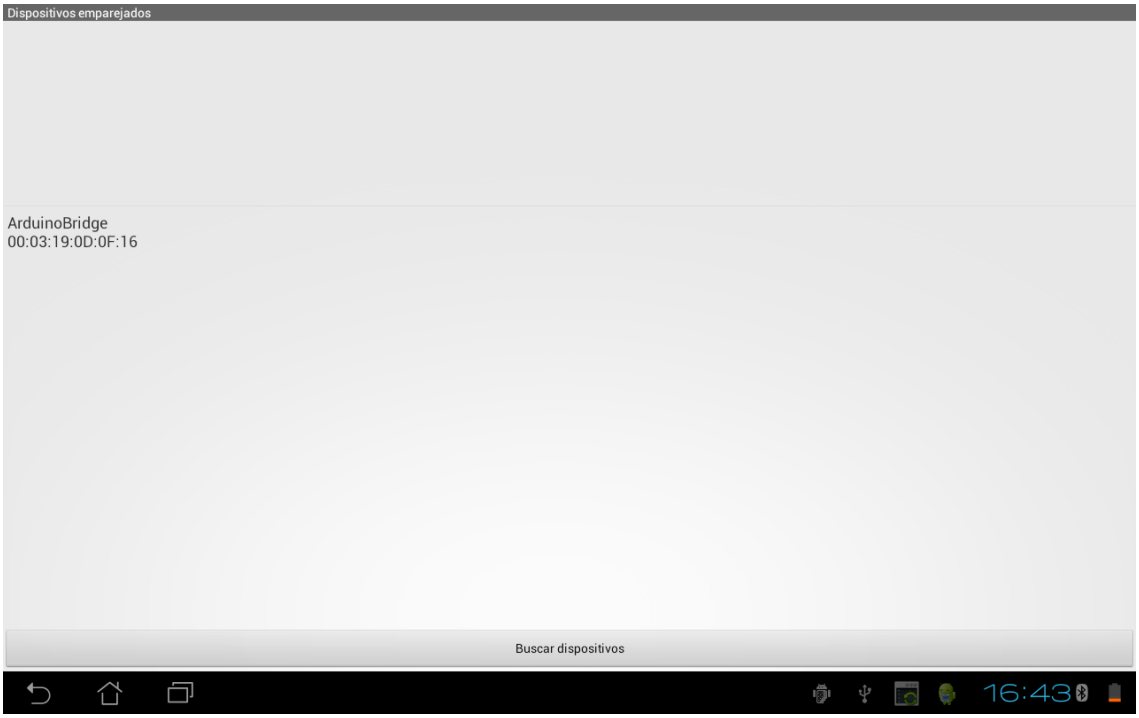
8.1 Pruebas básicas

En las pruebas básicas, se ha buscado comprobar la fiabilidad de los elementos básicos del sistema, como son la conexión, el envío de órdenes y la obtención de ficheros. Estas pruebas validan el funcionamiento completo de los componentes subyacentes, que son la conexión Bluetooth y el funcionamiento del paso de mensajes tanto vía Bluetooth entre la tableta y la placa Arduino como vía Serial entre la placa Arduino y el controlador del brazo robótico.

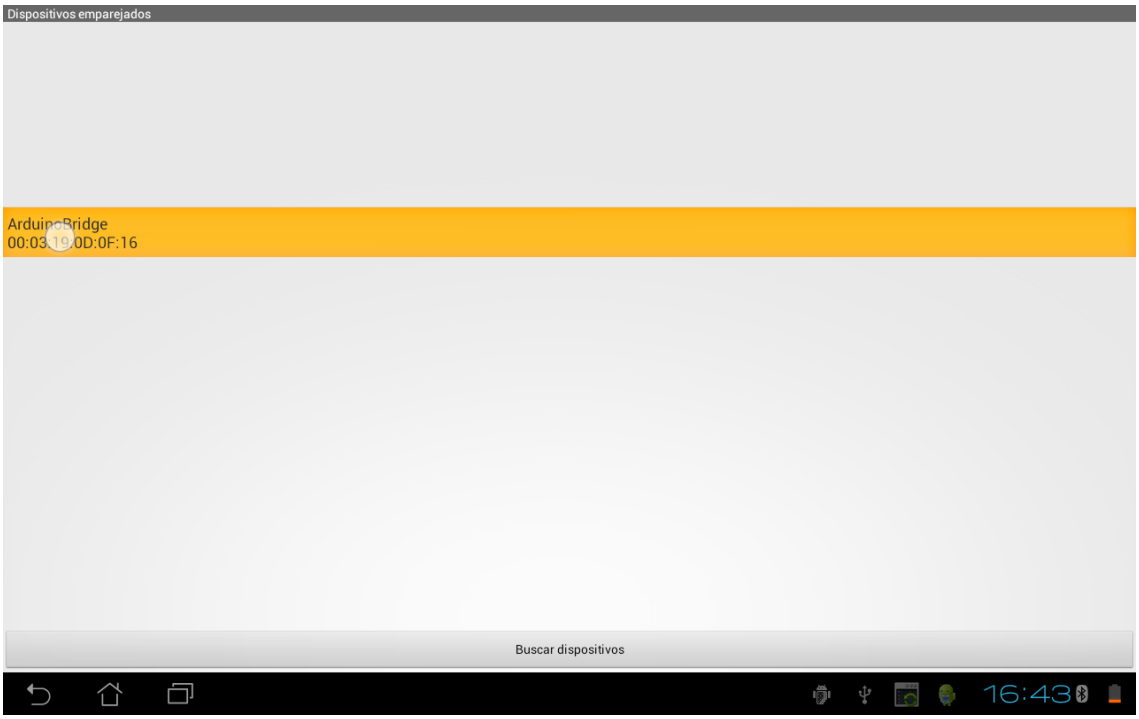
8.1.1 Prueba de conexión

Una de las principales pruebas a realizar es la prueba de conexión, donde se valida la conexión Bluetooth. Para ello, nos conectaremos a la placa Arduino desde la interfaz de la aplicación, y veremos qué sucede.

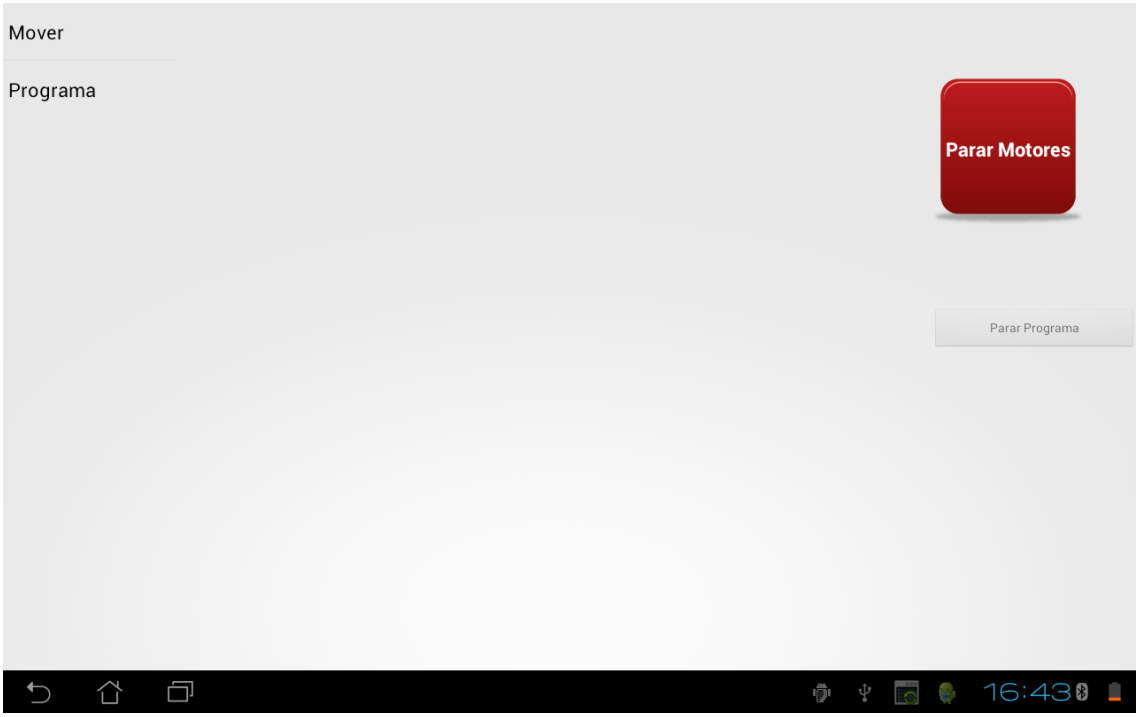
Primero, buscamos los dispositivos Bluetooth cercanos:


Dispositivos emparejados
ArduinoBridge
00:03:19:0D:0F:16
Buscar dispositivos
16:43

Una vez encontrado, pulsamos en el que deseemos, en este caso, ArduinoBridge:



Nos hemos conectado.



Con esto, podemos indicar que la conexión Bluetooth, está validada.

8.1.2 Prueba de Envío de Órdenes

Otra prueba principal a realizar es la validación del envío de órdenes al Brazo Robótico. Para ello, procederemos a realizar tres tipos de prueba, una orden simple, una orden con parámetros y una orden con retorno.

8.1.2.1 Prueba de una orden simple

Para la prueba de envío de una orden simple, se ha optado por la orden HH, que ordena al Brazo Robótico que se coloque en la posición de origen a nivel Hardware. Para realizar esta prueba, enviamos la orden "HH" y, tras procesarla, el Brazo Robótico se ha dirigido a la posición inicial.

8.1.2.2 Prueba de una orden con parámetros

Para la prueba de envío de una orden con parámetros, se ha escogido la orden OB, que pone la salida digital escogida al valor indicado. Para realizar esta prueba, enviamos la orden "OB,1,1", que debería poner la salida digital número 1 con un 1 lógico. Para poder comprobarlo, se puede observar la bombilla led[17] que dispone el controlador y, si se ilumina, es que se tendrá un 1 lógico. Tras enviar la orden, se ilumina la bombilla led.

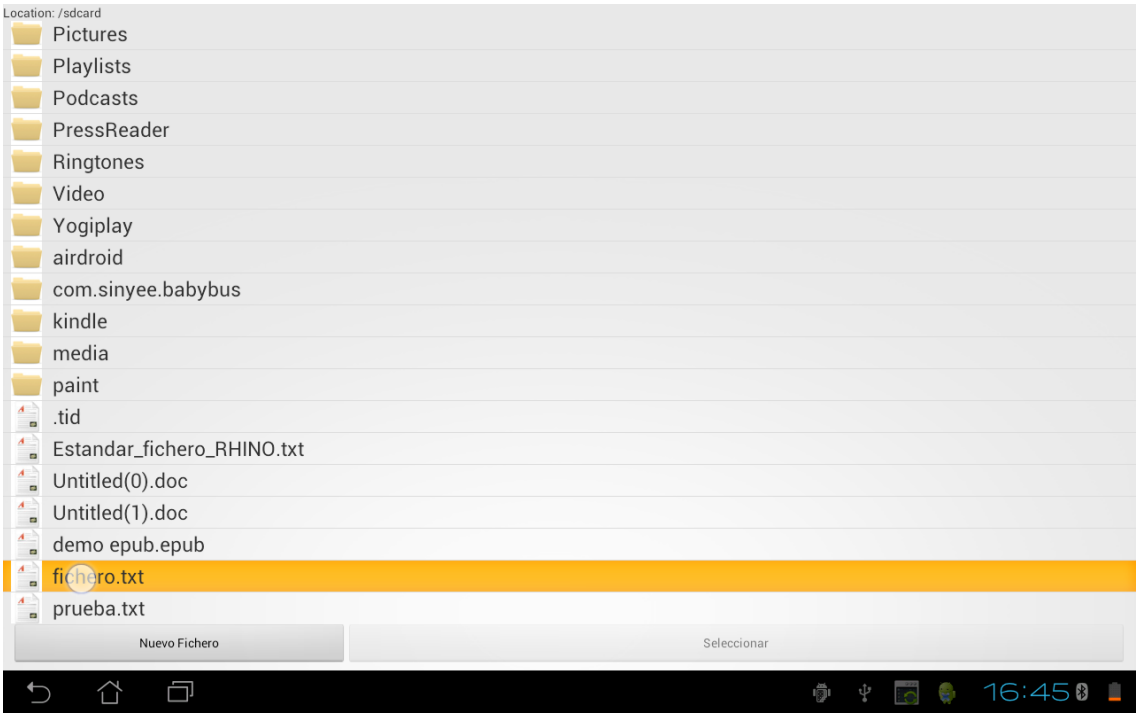
8.1.2.3 Prueba de una orden con retorno

Para la prueba de envío de una orden con retorno, se ha escogido la orden SE, que obtiene el código de error del controlador. Si este código es 0, indica que no hay ningún error, si el retorno toma otro valor, este valor indicará el tipo de error. Para realizar esta prueba, se envía la orden "SE" y obtenemos en la tableta el valor 0.

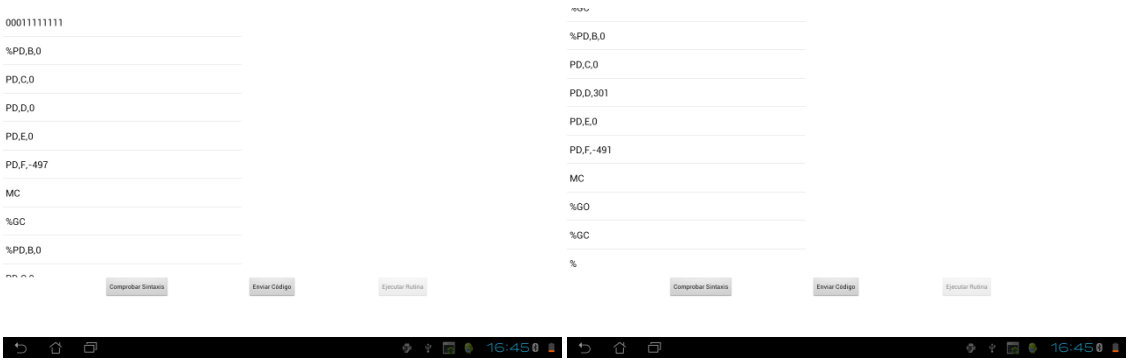
[17] La bombilla led es un diodo que emite luz. La palabra led proviene de las siglas en Diodo Emisor de Luz (Light-Emitting Diode en inglés).

8.1.3 Prueba de Obtención de Ficheros

Para la prueba de obtención de ficheros, obtendremos el código fuente almacenado en un fichero y lo mostraremos por pantalla (en la interfaz de *Enviar Rutina*). Para ello, primero seleccionamos el fichero:



Una vez seleccionado, pulsamos *Seleccionar*:



Y sabiendo que el fichero a leer era:

00011111111
%PD,B,0
PD,C,0
PD,D,0
PD,E,0
PD,F,-497
MC
%GC

%PD,B,0
PD,C,0
PD,D,301
PD,E,0
PD,F,-491
MC
%GO
%GC

Podemos afirmar que la obtención de ficheros ha sido validada

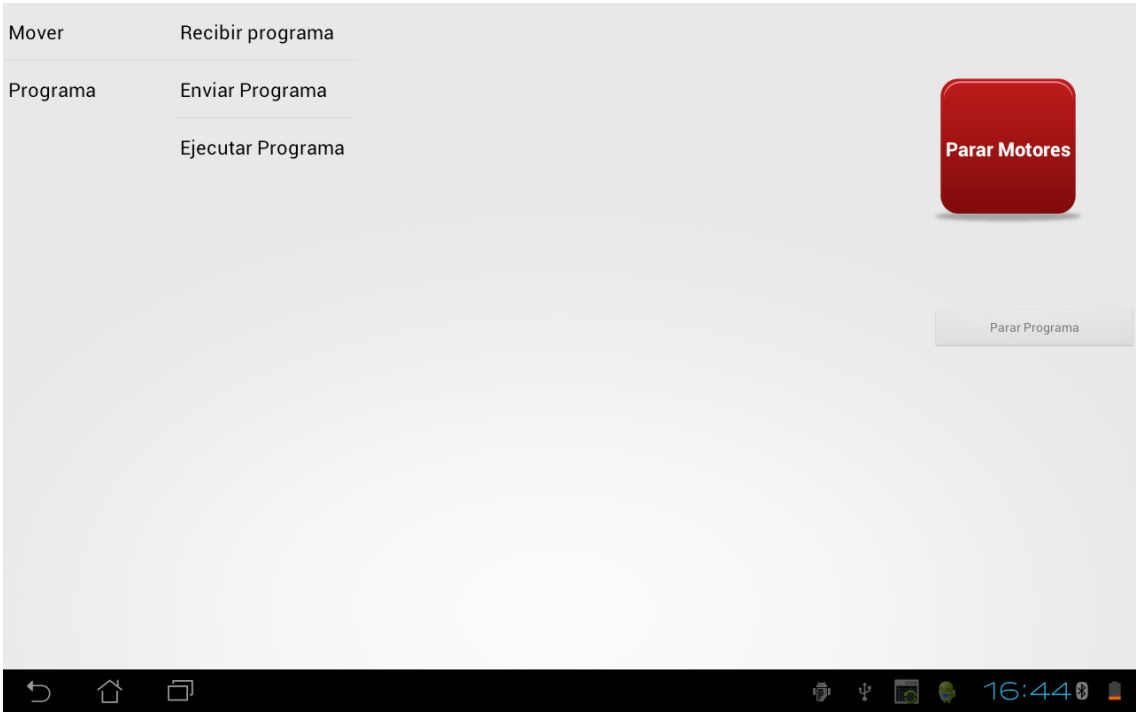
8.2 Pruebas Completas

8.2.1 Pruebas del Modo Paleta

Para probar el funcionamiento del modo Paleta hay que comprobar que funcionan los siguientes elementos:

- La recepción de rutinas
- La parada de motores
- La parada de rutina

Para las dos primeras pruebas, lo que se hará es recibir la rutina almacenada en la memoria EEPROM del controlador. Para ello, se ha pulsado el botón *Recibir Programa*:



Una vez hecho esto, hemos recibido la rutina y nos ha solicitado que indiquemos dónde deseamos almacenarla:

Tras estos pasos, hemos verificado que el funcionamiento es correcto. El siguiente elemento a comprobar es la parada de emergencia de los motores. Para poder comprobar su funcionamiento, hemos decidido mover un motor y, a mitad de su movimiento, pararlo. Para ello, enviamos la orden “PD,E,1000”, que define la posición de destino del motor E hasta la posición 1000. Tras enviarlo, ahora debemos activar el motor utilizando la orden “MS,E”. Tras enviarla, pulsamos el botón *Parar Motores* y, como era de esperar, se ha detenido.

Por último, debemos comprobar la parada de rutina. Para poder hacer las pruebas pertinentes, debemos ejecutar una rutina y para ello, se utilizará la orden “FX”, que ejecuta la rutina almacenada en la memoria EEPROM del controlador. Tras enviar la orden, pulsamos el botón *Parar Programa* y el brazo robótico se para.

Con estas pruebas, podemos indicar que el modo Paleta funciona correctamente y cumpliendo con los requisitos exigidos.

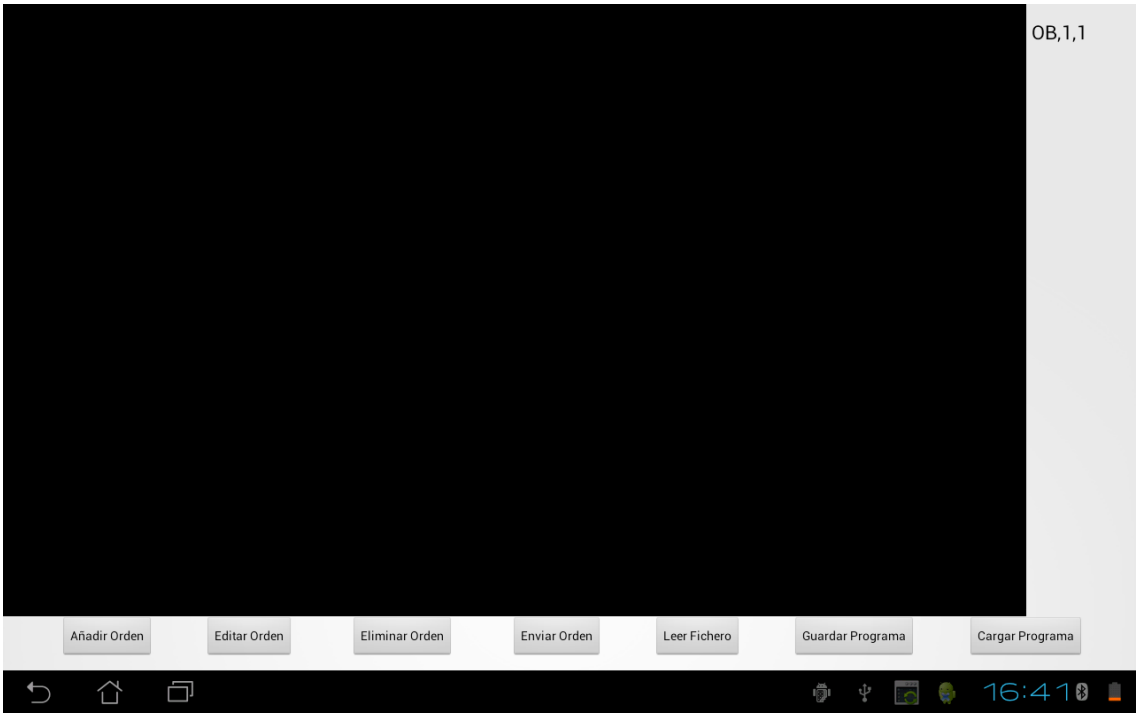
8.2.2 Pruebas del Control Programado

Para poder probar el funcionamiento del modo Programado, hay que comprobar los siguientes elementos:

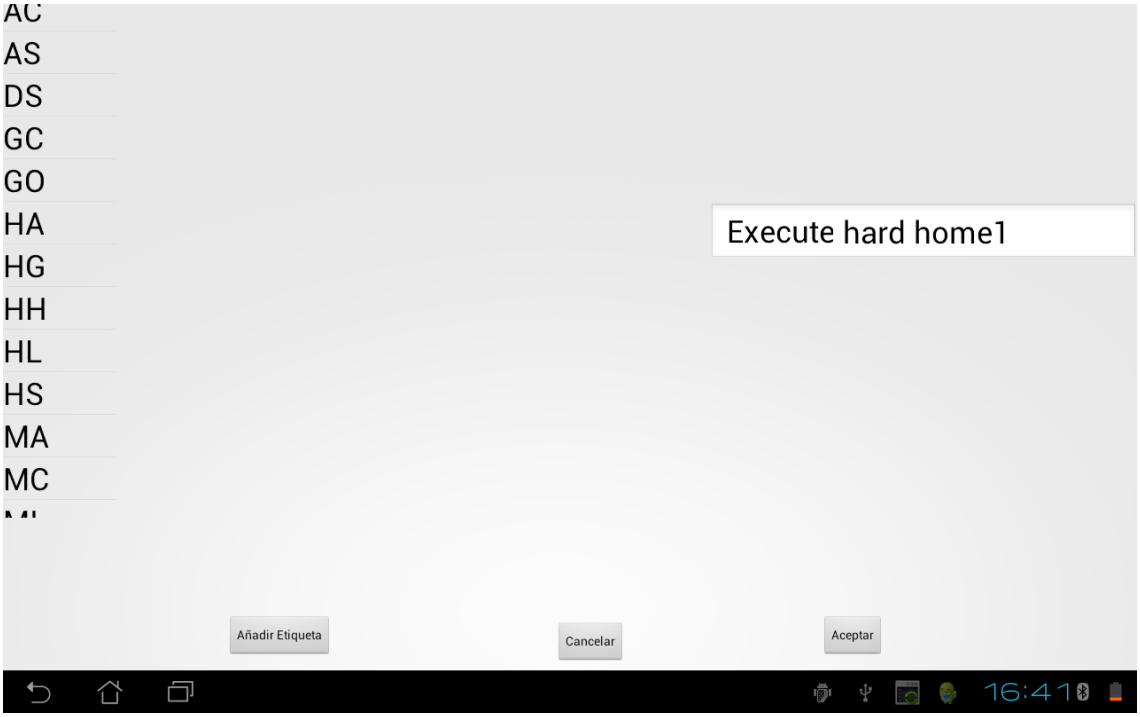
- La inserción de una orden
- La edición de una orden
- La eliminación de una orden

Para validar la inserción de una orden, hay que comprobar el funcionamiento de la interfaz de *InsertaOrden*. Para ello, pulsamos en el botón *Añadir Orden*:

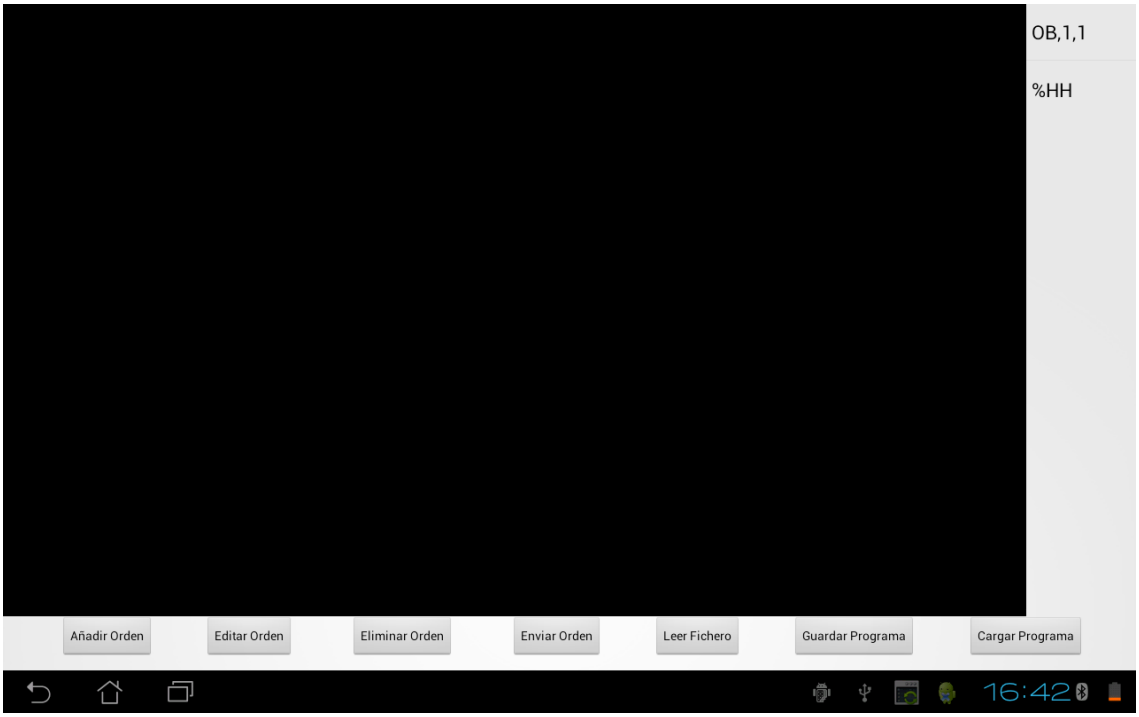
Y veremos la interfaz de selección de orden. Una vez dentro, navegaremos por las distintas órdenes y parámetros hasta escoger la que deseemos, en este caso “OB,1,1”:



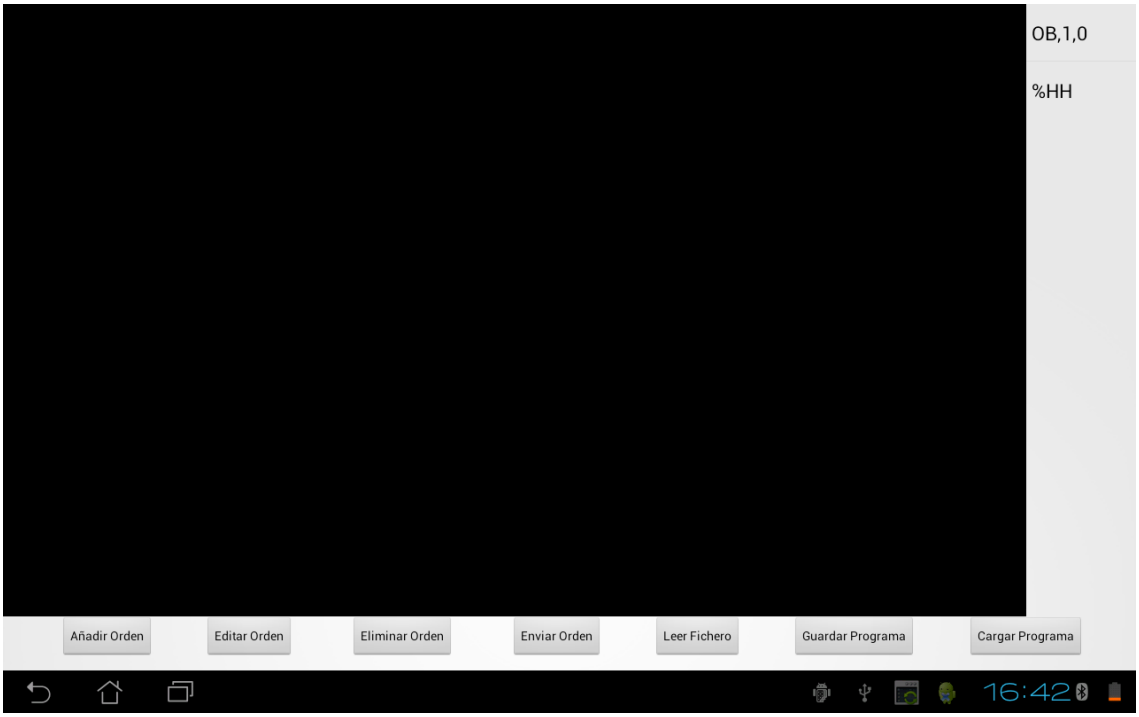
Ahora, comprobaremos también la inserción de la etiqueta:



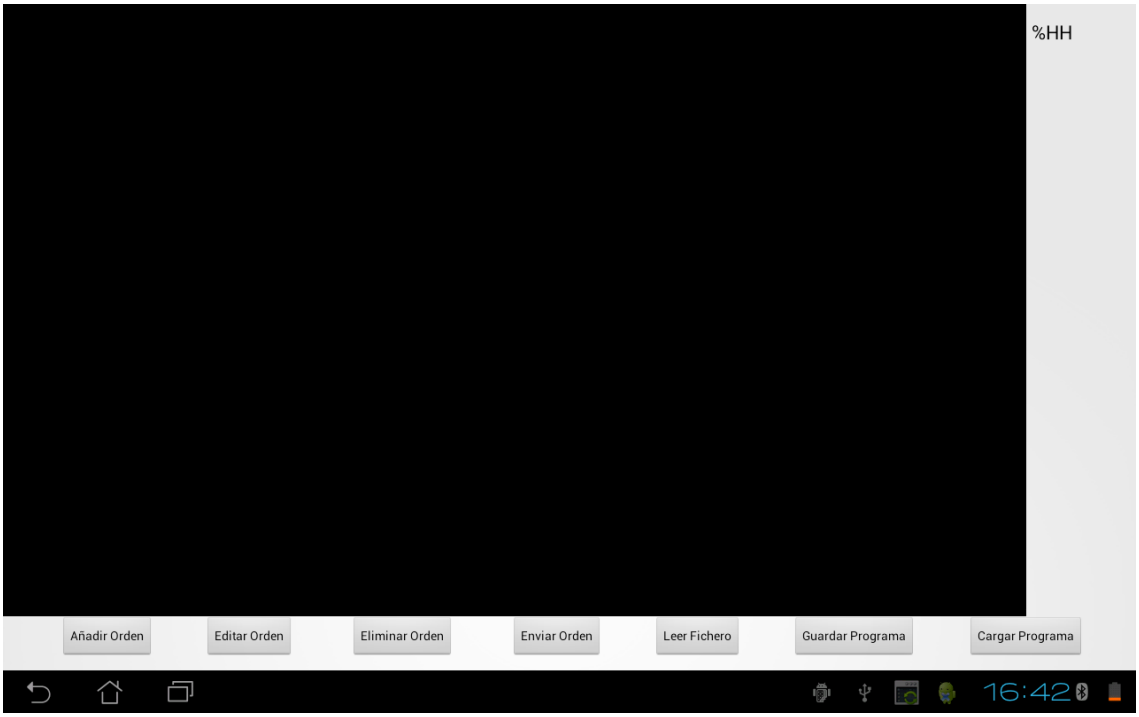
Y, como se puede comprobar, el símbolo de etiqueta ha sido añadido (el símbolo de la etiqueta para las rutinas del controlador RHINO MKIV es %):



Para validar la edición de una orden, seleccionaremos la primera orden, “OB,1,1”, y la modificaremos. Y, como podemos comprobar, la orden ahora es “OB,1,0”:

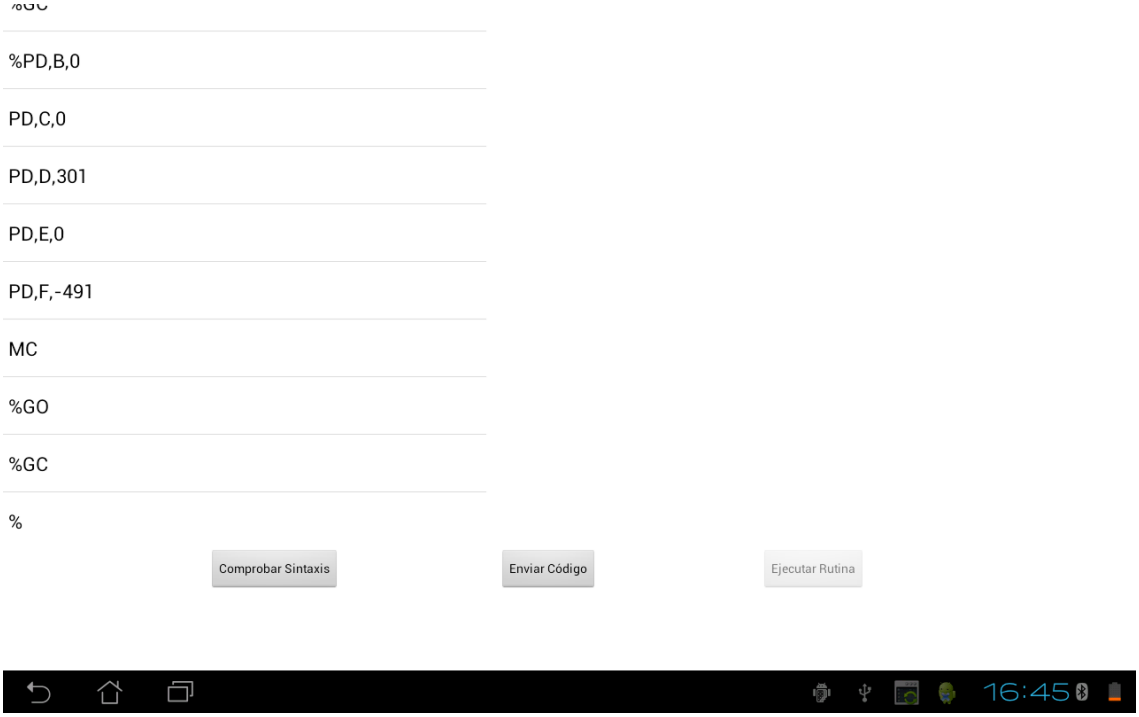


Por último, vamos a validar la eliminación de órdenes y, para ello, eliminaremos la primera orden, “OB,1,0”:



8.2.3 Pruebas del Ejecutar Rutina

Del modo Ejecutar Rutina, sólo nos falta validar un elemento, la validación de las rutinas a enviar. Para ello, y tras cargar un fichero:



Pulsamos en el botón *Comprobar Sintaxis* y, si no hay ningún elemento erróneo, nos lo indicará con un mensaje:

9. Conclusiones y perspectiva de futuro

9.1 Aportaciones y Conclusiones

Se ha creado un entorno de control y desarrollo para brazos robóticos en una plataforma táctil y basada en software libre. Cabe resaltar que no existe, a día de hoy, software de control robótico que integre el control y la programación de rutinas dentro de un dispositivo móvil.

Lo primero a destacar es la dificultad añadida de abordar un proyecto sin tener un trabajo previo sobre el que apoyarse, aumentando de manera considerable el tiempo requerido en la búsqueda de documentación, en la investigación de las distintas posibilidades y en la formulación de la solución que se ha implementado, además de los aspectos que se han englobado dentro de los posibles trabajos futuros.

Por otro lado, el trabajar con tecnologías nuevas para el alumno ha hecho que comprenda y descubra nuevos conocimientos más allá de los cursados a lo largo de la carrera, especialmente en el área de la robótica y de la programación orientada a dispositivos móviles.

Finalmente, especificar que se han alcanzado todos los objetivos especificados en la propuesta de proyecto (PFC-1) y se resumen en los siguientes puntos:

- Se ha preparado el desarrollo del PFC, estudiando el funcionamiento del Brazo Robótico y diseñando los componentes software que se desarrollarían
- Se ha programado la aplicación del PFC, diseñando e implementando los distintos modos de ejecución del mismo
- Se ha creado una batería de ejemplos prácticos que ilustran el funcionamiento del sistema

Además, se han aportado una serie de logros adicionales, más allá de los objetivos originales y que podemos resumir en los siguientes puntos:

- Se ha hecho énfasis en un desarrollo modular, escalable y eficiente. Así mismo se ha cuidado la eficiencia y su facilidad de mantenimiento.
- Se ha enfocado el sistema para permitir a usuarios sin conocimiento previo utilizar el sistema correctamente
- Se ha utilizado la comunicación con un sistema programable basado en un Arduino, aportando una gran flexibilidad futura al desarrollo.

9.2 Perspectiva de Futuro

Inicialmente, este proyecto surge de la conveniencia de actualizar los componentes software de comunicación de los sistemas robóticos del laboratorio de robótica, siendo el presente Proyecto Fin de Carrera una parte de la actualización deseada. Como continuación del desarrollo realizado, se abren un conjunto de propuestas dentro de esa línea de trabajo original que ha concluido con la creación de un entorno de control y desarrollo. A continuación, se destacan aquellas líneas de trabajo futuras que podrían ser abordadas:

- Interfaz de control manual del Brazo Robótico: este trabajo sería una actualización de la aplicación ya hecha. Trataría de una inclusión de un modo de control gráfico del Brazo Robótico, pudiendo mover los motores o el sistema en conjunto arrastrando las partes de una representación gráfica en la pantalla.
- Ampliación/generalización del entorno para gobernar más de un robot desde la tableta. Esto sería particularmente interesante en el laboratorio, ya que están disponibles un total de tres robots educativos, incluyendo un segundo robot Rhino tipo Scara).
- Desarrollo de la aplicación en otros sistemas: este trabajo consistiría en trasladar la aplicación ya desarrollada a otros sistemas, como pueden ser iOS de Apple o Windows 8 de Microsoft.
- Utilización de otros estándares de comunicación inalámbrica: este trabajo sería una actualización de la aplicación ya hecha y del programa en Arduino desarrollado. Se trataría de incluir la posibilidad de utilizar otros estándares, como puede ser WiFi[18], para conectarse con la placa Arduino, en lugar de Bluetooth.
- Control de dispositivos a larga distancia: este trabajo consistiría en modificar el sistema para permitir que se pudiera controlar brazos robóticos a larga distancia gracias a Internet.

[18] WiFi: es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica. Es una abreviatura del inglés *Wireless Fidelity*, y pertenece al estándar de redes inalámbricas de área local 802.11.

Anexo

Definición del Estándar

Para poder trabajar correctamente con el robot, se ha desarrollado una definición estándar del fichero a utilizar para el control del brazo robótico:

NOMBRE
NUM_MOTORES
RANGO_MOTOR_1_TICK,RANGO_MOTOR_1_ANGLE
RANGO_MOTOR_2_TICK,RANGO_MOTOR_2_ANGLE
...
RANGO_MOTOR_N_TICK,RANGO_MOTOR_N_ANGLE
ORDEN_CONEXION
ORDEN_HH
ORDEN_MOV_MOTOR
ORDEN_DEF_POS_MOTOR
ORDEN_CERRAR_PINZA
ORDEN_ABRIR_PINZA
ORDEN_RECIBIR_FICHERO
ORDEN_ENVIAR_FICHERO
MARCA/ORDEN_ETIQUETA
ORDEN_IR_ETIQUETA
ORDEN_INPUT
ORDEN_OUTPUT
ORDEN_DEF_SH
ORDEN_IR_SH
ORDEN_PARAR_MOTORES
ORDEN_EJECUTAR_RUTINA
ORDEN_PARAR_RUTINA
COD_CORRECTO_EN_ASCII
CABECERA_FICHERO
ORDEN_1,RANGO_PARAM_1,RANGO_PARAM_2,...,RANGO_PARAM_N _ EXPLICACION
ORDEN_2,RANGO_PARAM_1,RANGO_PARAM_2,...,RANGO_PARAM_N _ EXPLICACION
...
ORDEN_N,RANGO_PARAM_1,RANGO_PARAM_2,...,RANGO_PARAM_N _ EXPLICACION

Como ejemplo, disponemos del fichero utilizado para controlar el robot RHINO XR4 con el controlador RHINO MK-4:

RHINO
5
0..0
0..0
0..0
0..0
0..0
TH,TH,*SE
HH
MS
PD
GC
GO

TX,*SE,FT
TX,*SE,FR
$%
GL
IB
OB
HS
HG
MA
TX,*SE,FX
FA
48,13
00011111111
SA _ Read Motor Status
SC _ Read System Configuration
SD,0..3000 _ Stop/start delay timer
SE _ Read host error stack
SM, 0..3 _ Read motor mode
SP _ Read teach pendant error byte
SR _ Reset motor current limit circuitry
SS _ Read system status
ST _ Execute diagnostics
SU _ Read usage time
SV _ Read version and I.D. number
SX _ Execute diagnostics and return results
SZ _ Read the delay timer
CC,0..1 _ Set coordinate position
CG,0..1 _ Enable/disable gripper mode
CM,A..F,0..3 _ Set Motor Mode
CR,0..2 _ Set robot type
AR _ Read system acceleration
DR,A..F _ Read motor pam level and direction
GS _ Read gripper status
HR,A..F _ Read soft home position
PA,A..F _ Read actual position
PW,A..F _ Read destination position
PZ,x..z _ Read xyz destination position
RL _ Read limit switches
UA _ Read xyz rotation angle
UH,x..z _ Read xyz home position
UO,x..z _ Read xyz offset
UT _ Read xyz tool length
UY _ Read xyz height of the elbow rotation axis
VA,A..F _ Read motor actual velocity
VR,A..F _ Read motor desired velocity
VX _ Read system velocity
XR,A..F _ Read auxiliary port level and direction
AC,A..F _ Clear motor actual position
AS,0..100 _ Set system acceleration
DS,A..F,-100..100 _ Sets an open loop mode motor's pwm level and direction
GC _ Close Gripper
GO _ Open Gripper
HA _ Go to the hard home position
HG _ Go to de soft home position
HH _ Execute a hard home
HL,B..F _ Hard home on limit switch
HS _ Set soft home
MA _ Stop all motors and aux ports
MC _ Start coordinated move

MI _ Start all motors, immediate mode
MM,B..F _ Stop single motor
MS,B..F _ Start single motor
MX _ Start an xyz move
PD,B..F,-32767..32767 _ Set destination position, absolute
PR,B..F,-32767..32767 _ Set destination position, relative
PX,B..F,-1000..1000 _ Set xyz destination, absolute
PY,B..F,-1000..1000 _ Set xyz destination, relative
VG,0..100 _ Set system velocity
VS,B..F,-100..100 _ Set Motor Velocity
XA,-1000..1000 _ Set xyz rotation angle
XH,X..Z,-1000..1000 _ Set xyz home position
XO,X..z,-1000..1000 _ Set xyz offset
XS,1..2, -100..100 _ Set aux port level and direction
XT,-1000..1000 _ Set tool length
XY,-1000..1000 _ Set height of elbow rotation axis
FR _ Receive teach pendant file from host
FT _ Transmit teach pendant file to host
FX _ Execute teach pendant program
TA _ Abort teach pendant program
TC _ Clear teach pendant display
TE,0..1 _ Enable/disable teach pendant to move motors
KA,A..F,0..255 _ Set proportional gain
KB,A..F,0..255 _ Set differential gain
KC,A..F,0..255 _ Set integral gain
RA,A..F _ Read proportional gain
RB,A..F _ Read differential gain
RC,A..F _ Read integral gain
KR _ Restore user gains from EEPROM
KS _ Store user gains to EEPROM
KX _ Restore factory gains
IB,1..16 _ Read input or switch bit
IP _ Read input port
IX _ Read switch port
OB,1..8,0..1 _ Set output bit
OP,0..255 _ Set output port
OR _ Read output port
OT,1..8,0..1 _ Toggle output bit
WA _ Abort all waits
WI,1..16,0..1 _ Wait on input or switch
GL,1..50 _ Go to Label