

***Facultad***

***de***

***Ciencias***

**TITULO EN LETRA TIPO ARIAL ROUNDED**

**TAMAÑO 20**

(Title in English)

Trabajo de Fin de Grado

para acceder al

**GRADO EN INGENIERÍA INFORMÁTICA**

Autor: Santiago Sañudo Martínez

Director: Mario Aldea Rivas

Co-Director: Alejandro Pérez Ruiz

Junio – 2018

Índice de contenido

[Agradecimientos 4](#_Toc513652476)

[Resumen 5](#_Toc513652477)

[Abstract 6](#_Toc513652478)

[Introducción 7](#_Toc513652479)

[Motivación 7](#_Toc513652480)

[Objetivos 7](#_Toc513652481)

[Metodología 8](#_Toc513652482)

[Tecnología y Herramientas Utilizadas 9](#_Toc513652483)

[Lego Mindstorms 9](#_Toc513652484)

[1ª Generación, Bloque RCX 9](#_Toc513652485)

[2ª Generación, Bloque NXT 9](#_Toc513652486)

[3º Generación, Bloque EV3 10](#_Toc513652487)

[Motores y sensores 10](#_Toc513652488)

[Adaptador Pistorms Mindstorms 11](#_Toc513652489)

[Raspberry Pi 3 B 12](#_Toc513652490)

[AOSP (Android Open Source Project) 13](#_Toc513652491)

[ADB (Android Debug Bridge) 14](#_Toc513652492)

[Protocolo I2C 14](#_Toc513652493)

[Desarrollo del Sistema Operativo Android para la Raspberry 16](#_Toc513652494)

[Estado Final 16](#_Toc513652495)

[Proceso de Elaboración 16](#_Toc513652496)

[Problemas Encontrados 16](#_Toc513652497)

[Desarrollo de la Librería para Lego Mindstorms en I2C para Linux 16](#_Toc513652498)

[Estructura de la Librería 16](#_Toc513652499)

[Compilador Cruzado 16](#_Toc513652500)

[Modelo de Demostración de la Librería de Lego Mindstorms 17](#_Toc513652501)

[Proceso de Montaje 17](#_Toc513652502)

[Algoritmo 17](#_Toc513652503)

[Funcionalidad Final 17](#_Toc513652504)

[Problemas Encontrados 17](#_Toc513652505)

[Modelo de Demostración de la Tesis de Alejandro 19](#_Toc513652506)

[Proceso de Montaje 19](#_Toc513652507)

[Algoritmo 19](#_Toc513652508)

[Funcionalidad Final 19](#_Toc513652509)

[Datos Recogidos y Conclusión 19](#_Toc513652510)

[Bibliografía 19](#_Toc513652511)

# Agradecimientos

Antes de comenzar, me gustaría agradecer a mis familiares más cercanos los cuales han hecho posible que llegue hasta donde estoy. Por ello, un simple gracias es poco para mis padres y mi hermano.

Durante estos largos años, no solo he estado yo luchando por conseguir este objetivo, sino que al igual que yo había muchos otros. Entre ellos, los grandes amigos que hecho a lo largo de la carrera y que seguramente sin ellos esto habría sido muchísimo más difícil y por ello solo me queda daros las gracias.

Por último, agradecer también a todos y cada uno de los profesores que en mayor o menor medida han hecho posible que adquiera todos los conocimientos que tengo hasta la fecha sobre este bonito e interesantísimo campo que es la Ingeniería Informática. Entre ellos tengo que destacar especialmente a Mario Aldea, Alejandro Pérez y Héctor Pérez, ya que gracias a su increíble paciencia y su increíble conocimiento en la materia han conseguido convertir un tema totalmente desconocido como era este trabajo en un primer momento en una experiencia enriquecedora y realmente bonita.

# Resumen

# Abstract

# Introducción

## Motivación

Puedo decir con seguridad que he nacido en pleno desarrollo tecnológico en lo que a la informática concierne, y afortunadamente sigue manteniéndose en continua mejora siendo está a veces demasiado rápida para mantener el ritmo con ella. Esto hizo que mi infancia estuviera rodeada por pequeñas consolas, juguetes robóticos o pequeños diseños de electrónica que permitían que aquello que hoy es mi pasión naciera.

Entre estos campos se encuentra uno de los principales campos que es la robótica, un pilar fundamental en este trabajo final de grado. Lo que hace muchos años era una forma de entretenimiento se ha convertido con este proyecto en una manera de no solo aprender y comprender como realmente funcionan los robots, sino que además ha permitido hacerlo de una manera atractiva y distendida. Mencionar que de todos los posibles campos que abarca la robótica, el trabajo se centra en el funcionamiento de los motores y sensores de la plataforma desplegada por Lego, Lego Mindstorms [b], empleando para esto el adaptador Pistorms.

Otra de las grandes razones por las que me ha gustado especialmente llevar a cabo el proyecto es por el simple hecho de que todo giraba en torno al mismo componente, un miniordenador llamado Raspberry Pi[c]. Esta, será empleada para sustituir la propuesta del ordenador encargado del control de los distintos dispositivos y sensores de Lego, de manera que para ello necesita emplear el adaptador Pistorms o comúnmente llamado “*brick*”. Este miniordenador es el responsable de que naciera mi pasión por la programación ya que comencé a realizar mis primeros scripts de código en una versión muy joven de este modelo años previos a comenzar a la carrera. Dado que ya estaba familiarizado con el dispositivo, esto supuso una motivación extra por realizar el proyecto.

## Objetivos

El objetivo principal de este trabajo final de grado consiste en establecer una demostración práctica de la Tesis del profesor Alejandro Pérez. Esta tesis a simples rasgos consiste en demostrar porque Android no es un sistema operativo ideal para ejecutar aplicaciones de tiempo real por defecto debido a que el sistema toma el control de los procesadores haciendo que los tiempos de espera hasta que los hilos de ejecución de nuestras aplicaciones vuelvan a ejecutarse sean demasiado altos, pese a que les establezcamos prioridad de tiempo real, por culpa de interrupciones no controladas o bajo grandes cargas en el procesador.

Pero esto tiene solución empleando una lógica basada en el CPUSET, que consiste en aislar nuestro programa en un único núcleo del procesador de manera que todos los demás hilos de ejecución estén en los demás núcleos haciendo que las interrupciones a nuestra aplicación sean ninguna o casi ninguna. Para ello se propuso realizar una demo empleando los sensores y motores de Lego Mindstorm controlados por una Raspberry empleando el adaptador de Pistorms.

Por ello el primer objetivo fue claramente elaborar un sistema operativo Android ejecutable en la Raspberry.

Posteriormente fue elaborar una librería basada en el TFG del alumno Carlos Ayerbe González, de manera que esta fuera únicamente dependiente del protocolo I2C y estuviera ejecutada en un sistema operativo basado en Linux que permitiera ejecutar con los mismos tiempos de respuesta que el Kit original de Lego las operaciones y funciones que los dispositivos de Lego ofrecen. Una vez obtenida dicha librería, se elaboraría una pequeña demo con un robot simple para demostrar su correcto funcionamiento.

Por último, se elaboraría otra pequeña demo donde se demostraría obteniendo los tiempos de ejecución que efectivamente la tesis del profesor Alejandro funciona y se comporta de manera correcta incluso bajo grandes cargas en el procesador.

## Metodología

En este caso ha sido realmente sencilla, se ha asistido a la Universidad de Cantabria todos los días por la mañana durante el desarrollo del proyecto, de manera que a medida que ha habido grandes progresos, o en su defecto grandes problemas, se han comentado con el profesor pertinente y estos se han ido celebrando o solucionando respectivamente.

Dado que se ha considerado como un trabajo formal como podría ser el de cualquier empresa, las reuniones sobre como progresaba el trabajo se realizaban a primera hora y a última, antes de terminar. En las reuniones de por la mañana se comentaba el trabajo realizado en la tarde previa, así como la tarea a desarrollar ese mismo día y en las reuniones al terminar la mañana se comentaban los progresos obtenidos.

# Tecnología y Herramientas Utilizadas

## Lego Mindstorms

Es una plataforma de robótica orientada al desarrollo de pequeños juguetes para niños. Está pensada para unificar la construcción de objetos o estructuras con la programación. Esto se realiza con el fin de que los niños obtengan una experiencia interactiva respecto a lo que programen para que hagan sus diseños.

Esta plataforma está fabricada por la empresa LEGO, y actualmente consta de 3 generaciones.

1ª Generación, Bloque RCX

Es la primera generación y cuenta con la posibilidad de lectura de 3 sensores distintos, así como la capacidad de almacenar programas en su memoria interna. El microcontrolador de esta generación era un Hitachi H8/3292, y su gran desventaja es que no permitía la paralelización de código.



2ª Generación, Bloque NXT

Esta segunda generación presenta ciertas mejoras tanto en los sensores aumentado la calidad y la precisión de los mismos e incluyendo nuevos tipos. Pese a que se cambia el microcontrolador por un modelo cuyo procesador es un ARM7, seguimos sin poder paralelizar código de manera efectiva. Esta generación añade una nueva cualidad, la conectividad por Bluetooth además de una nueva interfaz USB, facilitando drásticamente la comunicación con el dispositivo.

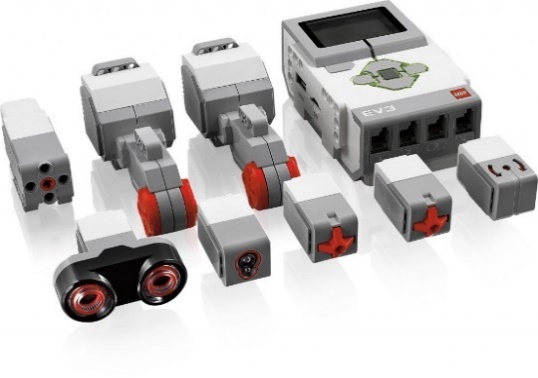


3º Generación, Bloque EV3

Esta última generación es una enorme mejora en todos los aspectos, ya que permite la paralelización de código de manera efectiva, hay 4 conexiones para sensores y 4 para motores e incluye un altavoz integrado. Además, se puede emplear una tarjeta microSD para comunicarse con el dispositivo o el wifi, sin excluir a las añadidas en la anterior generación, el bluetooth y el USB. Esta generación cuenta con un sistema operativo basado en Linux además de mejorar su procesador a una arquitectura ARM9. También incluye sensores de color, comunicación mediante infrarrojos, un mando a distancia y servomotores.



En nuestro caso, la idea fundamental del proyecto será sustituir el minicontrolador de la 3ª Generación, EV3 por la Raspberry Pi 3 b, con el adaptador de Pistorms de manera que realice la misma función que el mini controlador de Lego. Sin embargo, en lo que respecta a los motores y sensores utilizaremos todos los que están disponibles en un Kit de Lego de Mindstorms.



Motores y sensores

En estos Kits podemos encontrar 2 tipos de motores, uno grande y uno pequeño siendo la diferencia entre ambos la potencia y la capacidad de frenarse de manera no repentina.

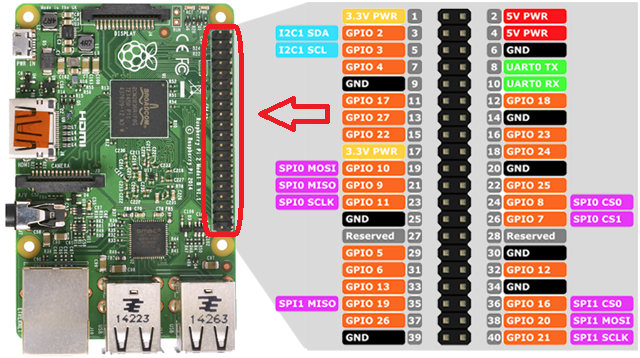
Respecto a los sensores encontramos:

* Sensores táctiles, permitiéndonos diferenciar cuando se pulsan.
* Un giroscopio, permitiéndonos saber el grado de inclinación que llevamos respecto del momento inicial de configuración o punto de referencia, así como la velocidad angular que el giroscopio lleva. Estas medidas solo se toman sobre un eje de coordenadas.
* Sensor de color, siendo capaz de distinguir entre 7 colores distintos, aunque la distancia de reconocimiento es bastante corta.
* Sensor ultrasonido, encargado de detectar la distancia con el objeto más próximo siendo muy útil y contando con una alta precisión.

## Adaptador Pistorms Mindstorms



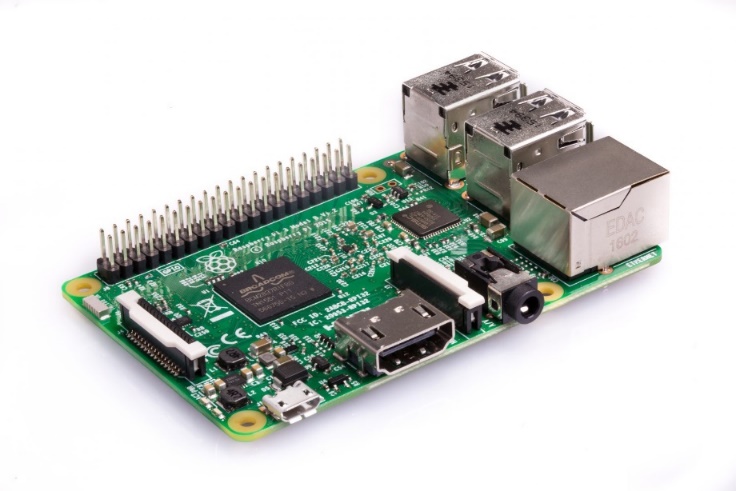
Este pequeño dispositivo es el responsable de establecer la comunicación de la Raspberry Pi con los sensores y motores EV3 de Mindstorms. Para ello, se conecta a las conexiones GPIO de la Raspberry Pi, tanto para alimentarse como para comunicarse con ella. Estas son las conexiones:



Este dispositivo para realizar la conexión con los sensores y motores de Mindstorms emplea conexiones de RJ12. Las conexiones se dividen en 2 bancos, el A y el B. En cada banco encontramos 2 conexiones para motores y 2 para sensores.

Para comunicarse la Raspberry con el Adaptador Pistorms toda la comunicación se realiza empleando el protocolo I2C que posteriormente comentaremos en que consiste.

## Raspberry Pi 3 B



La Raspberry Pi 3 B es la tercera generación de un modelo que inicialmente fue ideado como un miniordenador de bajo coste con los recursos mínimos necesarios para centrase en el aprendizaje y la enseñanza. Y así fue como salió al mercado en febrero de 2012 la primera versión. Este pequeño dispositivo consiguió con un coste muy bajo lanzar al mercado un dispositivo con alta conectividad y una capacidad de procesamiento más que suficiente para ejecutar pequeños programas. En la actualidad van por la versión 3 B+, sin embargo, nosotros utilizaremos la 3 B debido a que el lanzamiento de esta versión fue en pleno desarrollo del proyecto y el chipset era modificado ligeramente y no disponíamos del código fuente del kernel de la nueva versión que más adelante comentaremos para que era necesario.

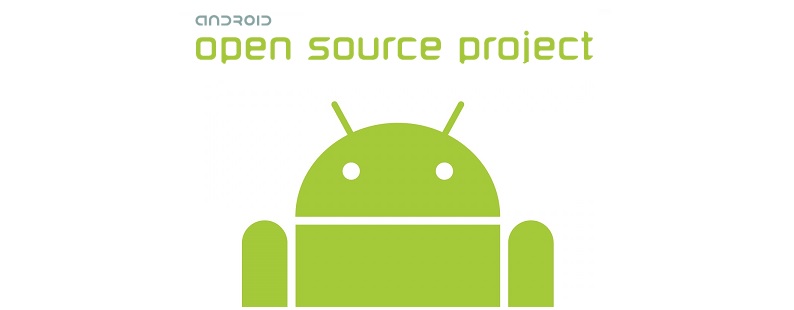
Mencionar que la Raspberry no solo ha servido para el campo de la enseñanza, sino también para que mucha gente pueda realizar pequeños proyectos como pueden ser emuladoras de consolas viejas o incluso pequeños rigs de minado. Actualmente cuenta con una comunidad muy grande que siempre intenta ser altamente creativa con nuevos proyectos y propuestas.

En nuestro caso la Raspberry Pi 3 B cuenta con las siguientes especificaciones:

* CPU: 1.2GHz 64-bit quad-core ARMv8 (Broadcom BCM2837)
* GPU: Broadcom VideoCore 4.
* RAM: 1 Gb compartido con la GPU.
* Bluetooth: Disponible, versión 4.1
* Wifi: Disponible a 2.4 Ghz, versión 802.11n.
* Puertos USB: 4 Puertos 2.0.
* Salida Video: HDMI.
* Salida Audio: HDMI y conector de 3.5 mm.
* Conexión Ethernet: Conexión RJ45 FastEthernet 10/100.
* Fuente de Alimentación: Mediante Micro USB con 5V o con la conexión GPIO.

## AOSP (Android Open Source Project)



Consiste en que un usuario desarrolle su propia versión de Android de manera que Google pone a disposición de todos los repositorios con el código fuente de las distintas versiones de Android que ha ido lanzando al mercado, y nosotros para nuestros pequeños proyectos tan solo debemos de compilar dicho sistema operativo siguiendo las instrucciones que Google proporciona en su página principal respecto al tema.

Sin embargo, además de lo previamente mencionado es necesario también saber para qué sistema estamos desarrollando el sistema operativo y, más concretamente, es necesario tener a nuestra disposición el código fuente del kernel de dicho sistema. En nuestro caso el código fuente del kernel de la Raspberry Pi 3 B pudimos encontrarlo en repositorio de Github de Igor Kalkov, Peter Yoon y Sahaj Sarup [a].

Otra caracterisctica clave que debemos tener en cuenta para que esto funcione es que debemos tener claro que el sistema operativo de Android está pensado para funcionar en una arquitectura de procesadores concreta, ARM. Dado que nuestra Raspberry funciona con dicha arquitectura no tuvimos ningún tipo de problema.

Una vez obtenidas ambas partes, kernel y sistema operativo, tan solo queda compilar ambas y obtener la imagen producida que usaremos para bootear nuestro sistema en Android, que en nuestro caso fue la versión 7.1.2. La gran funcionalidad que ofrece esto frente a la desventaja que supone el tener que descargarnos absolutamente todo el código el cual supone unos 150 GB en total, más la espera que conlleva la compilación de semejante cantidad de código es que nos permite establecer ciertas modificaciones tanto al kernel como al software que vendrá preinstalado en la imagen generada.

Una de las grande cosas que nos permitió modificar el tener un AOSP fue que pudimos establecer configuraciones por defecto al kernel de la Raspberry para activar el CPUSET, necesario en la demostración de la tesis del profesor Alejandro, siendo este elemento un componente clave, y activar de manera automática las conexiones e interfaces I2C en la Raspberry así como integrar el código fuente de dicho protocolo a la hora de compilar el sistema operativo para que este se encontrara ya disponible en la Raspberry al inicializarlo.

Otra de las grandes razones necesarias que influyo notoriamente en hacer un AOSP es que Android es un sistema basado en Linux, sin embargo, este está mucho más limitado respecto a lo que permite o no hacer al usuario no root, por ello era necesario modificar ciertas propiedades como es el sistema de autenticación a la hora de conectarse de forma remota mediante el protocolo ADB, permitiendo que el usuario generado por dicha conexión fuera superusuario. Esto es altamente importante debido a que las conexiones I2C empleadas en el sistema para comunicarnos con el Adaptador de Pistorms, se realizaran mediante lectura y escritura sobre ciertas zonas del sistema que solo son accesibles por el superusuario y por tanto para la ejecución de nuestros pequeños programas es totalmente necesario que esto fuera posible.

## ADB (Android Debug Bridge)

Este protocolo es un sistema de comunicación desarrollado por Google para permitir debugear un sistema operativo Android sin necesidad de tener que estar accediendo a los logs de las aplicaciones o del sistema para averiguar porque falla.

Sin embargo, con el paso del tiempo esta tecnología se ha ido desarrollando hasta tal punto que no sirve tan solo para debugear aplicaciones, sino que también nos sirve como funcionaria una simple función SSH a un ordenador, nos permite inicializarnos en un terminal del sistema Android de forma remota, y por tanto ejecutar scripts, programas o navegar por el sistema de ficheros de manera a como haríamos por un terminal de Linux normalmente. También nos permite instalar o desinstalar aplicaciones en java, o incluso montar y desmontar particiones concretas del sistema teniendo los permisos apropiados.

## Protocolo I2C

El protocolo I2C consiste en realizar una comunicación entre sistemas embebidos de manera simple y sencilla. Hoy en día podemos decir que se encuentra entre los más usados, entre los que destacan UART, SPI o CAN.

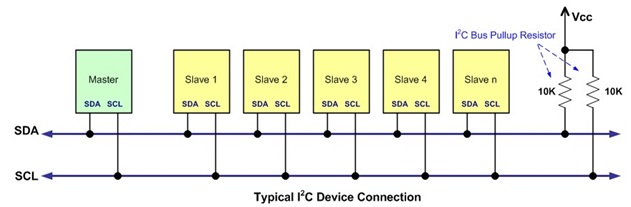
La idea sobre la que está basada no es ningún concepto nuevo ya que la vemos aplicada a numerosos y distintos tipos de algoritmos de control, el Maestro/Esclavo. En principio la idea es bastante sencilla, los distintos esclavos serán los múltiples sensores o dispositivos capaces de enviar o recibir información mientras que el maestro será generalmente la CPU encargada de determinar cuál de los esclavos es el permitido realizar la comunicación.

Por esta razón simultáneamente no puede haber 2 esclavos ocupando el bus I2C. Si deseáramos mandar la información del primer esclavo a otro segundo esclavo, esto habría que hacerlo en 2 partes, de manera que almacenáramos dicha información, como puede ser un buffer, y posteriormente cambiáramos el esclavo permitido a comunicarse para enviarle a este, dicha información. En un principio esto puede parecer algo tedioso, y no nos permite total paralelización en la comunicación, sin embargo, obtenemos un método de comunicación rápido y sencillo.

La comunicación en el bus está basada en 2 líneas, denominadas SDA (Serial Data) y SCL (Serial Clock). Como su propio nombre indica, SDA es la línea por la que es enviada y recibida la información en ambos sentidos, mientras que SCL es la línea que emplea el maestro para establecer el reloj para la transmisión de datos, controlada mediante unos bits iniciales en la comunicación y unos finales, dejando reflejado de esta forma cuando hay una comunicación activa o no evitando así cualquier tipo de conflicto. Tanto el maestro como los esclavos están conectados siempre a ambas líneas.

En nuestro caso hemos utilizado las conexiones GPIO de la Raspberry para realizar una conexión con el protocolo I2C con el Adaptador Pistorms y a través de este una conexión a los sensores y los motores.

Un ejemplo de cómo podría ser un bus I2C donde haya múltiples esclavos podría ser de la siguiente forma:



# Desarrollo del Sistema Operativo Android para la Raspberry

## Estado Final

## Proceso de Elaboración

## Problemas Encontrados

# Desarrollo de la Librería para Lego Mindstorms en I2C para Linux

## Estructura de la Librería

## Compilador Cruzado

# Modelo de Demostración de la Librería de Lego Mindstorms

## Proceso de Montaje

## Algoritmo

## Funcionalidad Final

## Problemas Encontrados

# Modelo de Demostración de la Tesis de Alejandro

## Proceso de Montaje

## Algoritmo

## Funcionalidad Final

## Datos Recogidos y Conclusión

# Bibliografía

[a]: Repositorio de Github de Igor Kalkov, Peter Yoon y Sahaj Sarup

<https://github.com/android-rpi>

[b]: Lego Mindstorms

<https://www.lego.com/es-es/mindstorms>

[c]: Raspberry Pi, pagina web oficial

https://www.raspberrypi.org/