Facultad de Ciencias

MANEJADORES DE SENSORES Y ACTUADORES LEGO MINDSTORMS PARA RASPBERRY PI Y MARTE OS

(Lego Mindstorms sensors and actuators drivers for Raspberry Pi and MaRTE OS)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Carlos Ayerbe González

Director: Héctor Pérez Tijero

Co-Director: Mario Aldea Rivas

Junio – 2017

Índice de contenido

Índice de figuras

Agradecimientos

En primer lugar, me gustaría citar a mi familia más cercana, mis padres, mi hermano y mi pareja quienes me han apoyado estos meses interesándose día a día, apoyándome, guiándome y hacer que nunca me rinda.

También quiero mencionar a mis amigos de siempre quienes, sin ellos, esta etapa no hubiese sido lo mismo.

A los nuevos compañeros y sobre todo amigos que hemos coincidido en esta carrera, son muchos los buenos momentos vividos.

Y por último, querría mostrar mi agradecimiento especial a dos profesores que me han ayudado y orientado en el desarrollo de este proyecto y hacer éste camino más fácil, Mario Aldea Rivas y Héctor Pérez Tijero.

Resumen

En los últimos años, con el progreso tecnológico de la sociedad, la robótica se ha convertido en una disciplina muy popular y con una gran implantación en el ámbito educativo. El auge de la robótica está presente tanto a nivel escolar como universitario. Así, la robótica constituye un objeto de estudio en sí misma dentro de una titulación universitaria y además puede ser utilizada para enseñar otras materias como programación, sistemas de tiempo real, sistemas empotrados, sistemas de control o inteligencia artificial de una forma más atractiva.

A la expansión mundial de la robótica ha contribuido en gran medida la aparición de "kits" de bajo coste que contienen los elementos básicos y completos para introducirse en el mundo de la robótica. Entre otros, hay que destacar los "kits" de robótica basado en el computador Arduino y los ofrecidos por la empresa Lego con el producto Mindstorms.

Junto con este auge de la tecnología y la robótica, aparecen computadores de tamaño reducido y de bajo coste como es la Raspberry Pi. Las características que presenta este computador junto con los dispositivos necesarios son más que suficientes para poder desarrollar un robot. Además existen plataformas que consiguen la conexión entre la Raspberry Pi y los "kits" de robótica. Por último, algunos desarrollos robóticos pueden presentar requisitos temporales por lo que resulta necesario utilizar un sistema operativo de tiempo real como MaRTE OS.

El objetivo de este trabajo es el desarrollo de un entorno que facilita el desarrollo de sistemas robóticos basados en Raspberry Pi, y la programación de sensores y actuadores de Lego Mindstorms. Para ello se utilizará un adaptador que hará la conexión entre el computador y los dispositivos.

El entorno desarrollado permite sacar partido en un mismo sistema de las numerosas extensiones existentes para Raspberry Pi y de la comodidad de montaje y variedad de sensores y actuadores proporcionados por Lego Mindstorms.

La facilidad de trabajar con Raspberry Pi permite utilizar un sistema operativo de tiempo real como es MaRTE OS, con lo cual facilita a los alumnos practicar con sistemas empotrados y de tiempo real.

Palabras clave: Raspberry Pi, PiStorms, Lego Mindstorms, MaRTE OS, librería BCM2835.

Abstract

In recent years, with the technological progress of society, robotics has become a very popular discipline and with a great implantation in the educational field. The rise of robotics is present at both school and university level. In addition, robotics is an object of study in itself within a university degree, and can be used to teach other programming subjects, real-time systems, embedded systems, control systems or artificial intelligence, in a more attractive way.

The worldwide expansion of robotics greatly responds greatky the emergence of low-cost "kits" containing the basic and complete elements to enter the world of robotics. It is necessary to emphasize the kits of robotics based on the Arduino computer and those offered by the company Lego with the product Mindstorms.

With this growth of technology and robotics, there are small, low-cost computers such as Raspberry Pi. The specifications presented by this computer with the necessary devices are enough to develop a robot. There are also platforms that get the connection between the Raspberry Pi and the robotic kits. Finally, some robotic developments may have temporary requirements so it is necessary to use a real-time operating system such as MaRTE OS.

The aim of this work is the development of an environment that facilitates the development of robotic systems based on Raspberry Pi and the sensors, actuators and assembly parts of Lego Mindstorms. We will use an adapter that will make the connection between the computer and the devices.

The developed environment allows you to take advantage of the many existing extensions for Raspberry Pi and the mounting convenience and variety of sensors and actuators provided by Lego Mindstorms.

The ease of working with Raspberry Pi is that allows you to use a real-time operating system such as MaRTE OS, which makes it easy for students to practice with built-in and real-time systems.

Keywords: Raspberry Pi, PiStorms, Lego Mindstorms, MARTE OS, BCM2835 bookstore.

1. INTRODUCCIÓN

1.1 Motivación

La evolución de la tecnología que estamos viviendo de cerca favorece la aparición de nuevas áreas de estudio que pueden ir integrándose en nuestra educación. Una de estas nuevas áreas es la robótica educativa. Esta nueva área ha crecido muy rápidamente en los últimos años en todo el mundo y está muy presente en nuestras vidas, ya que los robots están incorporándose en nuestro día a día, pasando tanto por la industria y trabajo hasta nuestras casas.

La idea de implementar la robótica como herramienta de educación no es nueva. Ya en 1983 el Laboratorio del Instituto Tecnológico de Massachusetts desarrolló el primer lenguaje de programación educativo llamado “logos”. La aparición de kits de robótica ha ayudado a su implantación, ya que éstos son muy manejables y versátiles y no exigen un gran conocimiento de programación o electrónica. Uno de los kits más importantes es el de Lego Mindstorms [1] que es el que utilizaremos en el presente trabajo.

Hoy en día la robótica se está implantando en algunos programas de las escuelas primarias y secundarias, así como en las universidades. También aparecen talleres extraescolares centrados en esta rama, con lo cual hace que el mundo de la robótica se esté expandiendo cada vez más. Esta expansión se debe a la atracción por parte de los jóvenes al ver sus proyectos funcionando. Muchas actividades educativas empiezan a depender de esta atracción por los robots [2].

Además, la robótica en la educación se ha venido practicando en los países más avanzados de todo el mundo como Finlandia, Reino Unido o Estados Unidos, donde a edades muy tempranas ya incluyen en su educación kits de robótica. Muchos de sus centros educativos ya tienen robots educativos y defienden que los robots y sus aplicaciones son un nuevo medio de aprendizaje y desarrollo de la creatividad, el pensamiento o el razonamiento, además de potenciar habilidades como la resolución creativa de problemas, la capacidad organizativa o la toma de decisiones [2].

Dentro del ámbito de la educación universitaria, la robótica constituye un objeto de estudio en sí misma, pero además puede ser utilizada para enseñar de una forma atractiva otras materias como programación, sistemas de tiempo real, sistemas empotrados, sistemas de control, inteligencia artificial, etc.

En la actualidad, uno de los ejemplos más relevantes es Lego Mindstorms. La característica que destaca de este sistema de desarrollo de robótica es que nos

podemos centrar más en la parte de programación sin tener que dedicar demasiado tiempo a la parte mecánica del robot, ya que el kit lo facilita con diversas piezas de montaje que encajan a la perfección. Indicar que ha habido una evolución importante en esta rama de la robótica dentro de la empresa de Lego, desde el bloque RCX, pasando por el kit NXT hasta la última versión que se ha llamado EV3.

Otro dispositivo que ha entrado con fuerza en el mundo de la tecnología es la Raspberry Pi. Esto se debe a que es un computador de bajo coste y tamaño reducido, con prestaciones suficientes y que permite realizar proyectos muy diversos. Además de existir comunidades y foros que cuentan con un gran número de participantes, en los cuales se pueden ver la gran cantidad de proyectos que hay y poder consultar soluciones de otros usuarios [3]. Existen diferentes versiones de la placa Raspberry Pi e incluso otros dispositivos con características similares como por ejemplo la placa de Arduino [4].

Junto a la aparición de la Raspberry Pi a su vez se han desarrollado placas o adaptadores que se conectan a ésta y permiten utilizarla para realizar proyectos de robótica. Estos adaptadores hacen de interfaz con los elementos de Lego Mindstorms, como son los sensores, actuadores e incluso suelen estar diseñados para la adaptación con las piezas de montaje del kit, lo que permite un desarrollo rápido de un robot. Suelen conectarse a la Raspberrry Pi a través de los pines de entrada/salida (GPIO) y utilizar algún bus de comunicaciones como el I2C. Dos de los adaptadores que se han investigado y estudiado para nuestro trabajo han sido el PiStorms [5] y BrickPi [6].

Algunos desarrollos robóticos tienen requisitos temporales por lo que necesitan utilizar un sistema operativo de tiempo real como MaRTE OS [7], que proviene de un proyecto desarrollado por el grupo de Ingeniería Software y Tiempo Real en la propia Universidad de Cantabria. Este sistema operativo soporta los lenguajes de programación C y Ada, siendo el lenguaje C el elegido para el desarrollo de nuestro proyecto.

1.2 Objetivos

El principal objetivo de este Trabajo de Fin de Grado es crear un entorno de desarrollo y programación de robots que se pueda utilizar en docencia de Grado y Máster universitarios.

El entorno está basado en la utilización del computador Raspberry Pi y de los sensores, actuadores y piezas de montaje de Lego Mindstorms. Para la conexión de los sensores y actuadores de Lego con el computador Raspberry Pi se utiliza el adaptador PiStorms.

El entorno desarrollado permite sacar partido en un mismo sistema de las numerosas extensiones existentes para Raspberry Pi y de la comodidad de montaje y variedad de sensores y actuadores proporcionados por Lego Mindstorms.

El sistema operativo de tiempo real utilizado es MaRTE OS, ya que permite acceder a los recursos del hardware y soporta aplicaciones desarrolladas en C entre otras características. Además, nos hemos basado en un proyecto existente, en el que se porta este sistema operativo a la placa Raspberry Pi.

Para la consecución del objetivo final ha sido necesario abordar tres objetivos intermedios:

- Facilitar al acceso a los pines del GPIO (*General Purpose Input/Output*) del Raspberry Pi desde MaRTE OS.
- Desarrollar una librería de uso de dispositivos I2C desde MaRTE OS.
- Desarrollar una librería que facilite al programador el uso de los sensores y actuadores Lego Mindstorms ocultando los detalles de la comunicación con el adaptador PiStorms.

1.3 Metodología

Para llegar al objetivo tratado en el anterior punto, seguimos la metodología conocida como el desarrollo iterativo incremental.

Es un modelo derivado del ciclo de vida en cascada. El modelo en cascada contiene un ciclo de vida clásico y secuencial hacia el desarrollo del software, y se inicia con la especificación de requisitos en nuestro caso del entorno que vamos a desarrollar, continúa con la planificación o diseño del mismo, le sigue el modelado y la construcción, y finalmente el despliegue con las pruebas y mantenimiento.

Lo que va a conseguir el desarrollo iterativo incremental es que al finalizar una iteración del modelo en cascada y llegar al objetivo indicado, se vuelve al primer punto de análisis para repetir los mismos pasos y mejorar con una nueva versión el objetivo anterior. Es decir, consiste en varios ciclos de vida del modelo en cascada, en el cual en cada iteración vamos incluyendo mejoras y nuevas funcionalidades a la aplicación.

A continuación, en la figura 1 se muestran las etapas del desarrollo iterativo incremental.

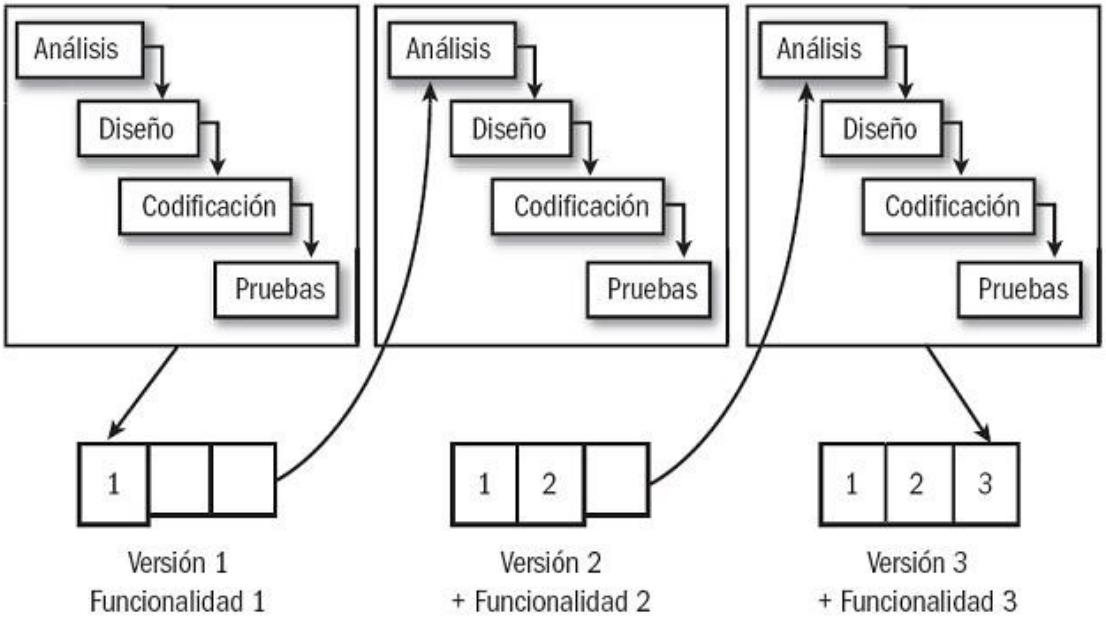


Figura 1: Esquema del modelo incremental iterativo

Como se puede observar en la figura 1, el ciclo de vida de una iteración en este modelo consta de: análisis del sistema, diseño del problema, la codificación o implementación de la solución y finalmente las pruebas a esa solución creada. Con todo ello se crea una iteración y por consiguiente la resolución del objetivo. En la parte incremental del modelo, se añadirá nuevas funciones y mejoras creando una nueva versión más compleja. Para ello se empezaría otra vez con el análisis y seguiríamos los mismos pasos seguidos en la iteración anterior pero ahora con el nuevo objetivo.

La primera iteración de nuestro trabajo constaba en poder controlar los pines del GPIO de la Raspberry PI desde MaRTE OS. Para ello se analizaron las librerías de uso para la GPIO. Seguido se creó un breve diseño de la solución y se implementó un programa para realizar las pruebas.

Una vez facilitado el uso de la GPIO, la segunda iteración tenía como objetivo el desarrollo de una librería de uso de dispositivos I2C desde MaRTE OS. Para ello se empezó con un análisis de las librerías que dan soporte para el uso del bus I2C. Una vez seleccionada la librería se pasó al diseño para adaptar y cambiar dicha librería a MaRTE OS, y seguidamente se implementó. Finalmente se realizaron las pruebas de comprobación de la librería.

Una vez conseguido el anterior objetivo y finalizada la iteración se establece un nuevo objetivo en el que se trata de desarrollar una librería que permita al programador la utilización de los sensores y actuadores de Lego Mindstorms.

De nuevo se realiza un análisis de requisitos para nuestra librería, seguidamente se realiza un diseño que será presentado en apartados posteriores en la memoria, se implementará en lenguaje de programación C y finalmente se realizan las pruebas tanto unitarias para cada sensor y motor, como pruebas generales con un demostrador que será presentado más adelante.

Cada iteración muestra una nueva versión con mejoras incluidas respecto a la iteración anterior, de ahí que sea incremental e iterativa.

2. TECNOLOGÍA Y HERRAMIENTAS UTILIZADAS

En este apartado se van a presentar y describir las principales características de los dispositivos y herramientas utilizadas.

2.1 Raspberry Pi

La Raspberry Pi se desarrolló con la idea de facilitar la enseñanza de la informática en las escuelas y en países en vías de desarrollo. Debido a ello y por la gran variedad de funciones que permite realizar, está teniendo una amplísima utilización a nivel mundial, no solo en las escuelas también en las universidades, empresas o proyectos de usuarios.

Una parte importante de la placa Rasbperry Pi es el Chipset. Su función es entablar la conexión correcta entra la placa madre y los diversos componentes esenciales de la placa, como lo son el procesador, las memorias RAM y ROM, la GPIO y el bus I2C entre otros. El modelo utilizado por la Raspberry Pi es el *System-on-a-chip Broadcom BCM2835*.

Las principales características del computador Raspberry Pi modelo B:

- Un Chipset Broadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 Mhz.
- Procesador gráfico (GPU) VideoCore IV.
- 512 MB de memoria RAM.
- Veintiséis pines GPIO (*General Purpose Input Output*).
- Puerto Ethernet 100 Mbps.
- Dos buses USB 2.0.
- Conector de alimentación microUSB.
- Salida analógica de vídeo RCA y otra salida analógica de audio estéreo por jack de 3.5mm.
- Salida digital de vídeo más audio HDMI.
- Lector de tarjetas SD.
- Dimensiones 85.60mm × 53.98mm

Figura 2: Imagen de la placa Raspberry Pi modelo B

Existen versiones más modernas y con mayores prestaciones como son la Raspberry Pi 2 o la Raspberry Pi 3, pero en nuestro caso hemos elegido la placa Raspberry Pi modelo B ya que presenta unas características más que suficientes para nuestro proyecto. Además, existe una adaptación del sistema operativo MaRTE OS [7].

2.2 Lego Mindstorms

La empresa LEGO creó una línea de juguetes centrada a la robótica educativa para niños y jóvenes. Esta empresa comercializa kits de robótica que incluyen piezas de montado para hacer cualquier prototipo de robot de la vida real, como puede ser un coche, un ascensor o una cinta de producción entre otros. Además, este kit educativo viene con un software que permite la programación de acciones de forma interactiva.

Son conocidas tres generaciones de Lego Mindstorms:

- La primera generación se conoce como Bloque RCX.
- La segunda generación es conocida como Bloque NXT.
- La tercera generación pertenece al Bloque EV3.

El kit de la tercera generación fue lanzado en septiembre de 2013, y sobre este último será sobre el que esté centrado nuestro trabajo.

En la siguiente imagen se presentan los principales elementos del kit EV3 [1]:

Figura 3: Kit de Lego Mindstorms generación EV3

En la figura 3 se puede apreciar el bloque central EV3 junto con los sensores y actuadores. Entre los sensores más destacados de la plataforma Lego, se encuentran: el sensor de ultrasonidos, que mide la distancia que hay hasta un objeto en centímetros o pulgadas, el sensor de color que es capaz de diferenciar hasta siete colores y además mide la intensidad de luz, el sensor giróscopo, que mide cómo de rápido gira y cuánto está girando el robot. Y el sensor de contacto, que reconoce si se está pulsando o no, y además cuántas veces ha sido pulsado.

Los actuadores o motores del Kit de Lego Mindstorms son: el Large Motor y el Medium Motor. Hay algunas diferencias entre ellos dos, por ejemplo, el Medium Motor tiene un peso de 39 gramos y una velocidad rotacional (rotaciones por minuto) de 260 rpm mientras que el Large Motor pesa 82 gramos y tiene una velocidad rotacional de 175 rpm [1].

En este trabajo, se ha dado soporte a estos sensores y actuadores de LEGO. También se han probado los actuadores del kit NXT y son estos últimos los que hemos utilizado para potenciar el demostrador que se presentará en el apartado correspondiente de la memoria.

2.3 Adaptadores de dispositivos Mindstorms a Raspberry Pi

Estas plataformas permiten utilizar los sensores y actuadores de los kits de Lego Mindstorms, nos aportan la misma funcionalidad que si tuviéramos

conectado el *brick* de Lego, además tienen un tamaño adecuado para poder desarrollar robots sin perder eficacia.

La existencia de diferentes adaptadores que sustituyen el Brick Inteligente de Lego Mindstorms nos hizo realizar un estudio y buscar información para luego hacer una comparación y elegir el que más nos convendría. Según íbamos leyendo y obteniendo datos de otros usuarios en la realización de diferentes proyectos, llegamos a la conclusión que había dos de ellos que destacaban del resto y de los cuáles podríamos tener más información. Estos dos adaptadores principales eran:

- PiStorms, distribuido por *mindsensors.com* [5].
- BrickPi, distribuido por *Dexter Industries* [6].

Para la elección del adaptador que vamos a utilizar, en este trabajo analizaremos las características, la documentación y el soporte y software disponible de cada uno de ellos. En la figura 4 aparece la sustitución del brick de Lego por el adaptador junto con la Raspberry Pi.

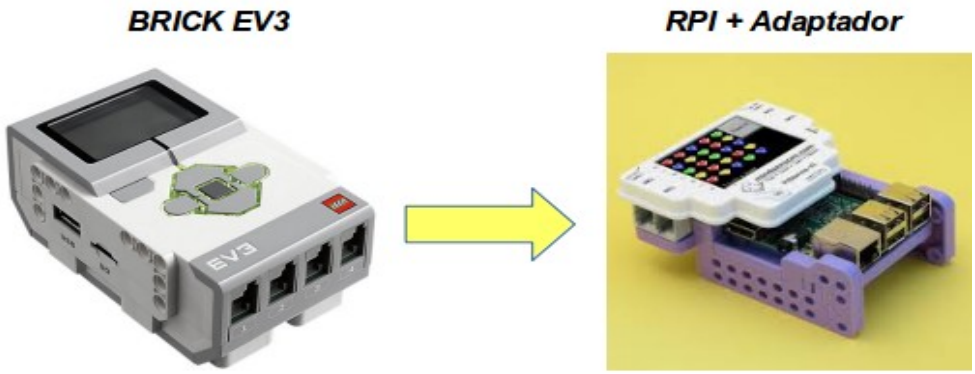


Figura 4: Sustitución del brick de Lego por RPI y adaptador.

2.4 Sistema Operativo ev3dev

El brick programable EV3 puede arrancar un sistema operativo alternativo desde una tarjeta miscroSD, como por ejemplo *ev3dev*, que es un sistema operativo basado en Linux Debian.

Este sistema operativo aporta gran flexibilidad a la hora de programar, ya que contiene drivers de bajo nivel para controlar los sensores y motores, y tan sólo escribiendo y leyendo ciertos archivos permite interaccionar con los dispositivos.

Una característica relevante del *ev3dev* es que está basado en Debian Linux,

por lo que existe una gran cantidad de paquetes de software libre disponibles para su instalación. Además, gracias al kernel de Linux, permite conectar dispositivos mediante USB o Bluetooth, como teclados, joysticks, cámaras o Wi-Fi dongles [8].

En nuestro trabajo hemos utilizado el *ev3dev* para tener una primera idea del funcionamiento y configuración del adaptador de Lego a Raspberry Pi, y así poder desarrollar posteriormente los drivers y librerías necesarios adaptados a nuestra plataforma.

2.5 Sistema Operativo MaRTE OS

MaRTE OS es un proyecto que se desarrollada por el grupo de Ingeniería Software y Tiempo Real en la Universidad de Cantabria. Señalar que es un sistema operativo de tiempo real para aplicaciones embebidas, que sigue el subconjunto de *Minimal Real-Time POSIX.13*. Proporciona un entorno para controlar y desarrollar aplicaciones *multi-thread* en tiempo real. Algunas características importantes de MaRTE OS [7].

- Soporta aplicaciones desarrolladas en Ada y C. Pudiendo mezclar ambos lenguajes.
- Sustenta las siguientes arquitecturas: x86 (32bits), Raspberry Pi, Linux.
- Soporte completo de tareas Ada (biblioteca de tiempo de ejecución GNAT adaptada a MARTE).
- Ofrece los servicios definidos en *POSIX.13*: pthreads(creación de hilos), mutexes, temporizadores.
- Todos los servicios tienen un tiempo acotado de respuesta.
- Espacio de dirección de memoria único compartido por la aplicación *multi-thread* y el sistema operativo MaRTE.
- Disponible bajo la Licencia Pública General GNU 2.
- Implementa el anexo de tiempo real Ada2012.
- Mayor parte de su código está escrito en Ada, C y ensamblador.

Hemos utilizado este sistema operativo para el desarrollo del trabajo, basándonos en un proyecto en el que se porta MaRTE OS a la arquitectura ARM que utiliza la placa Raspberry Pi.

En el sitio web del proyecto MaRTE OS [7] existe documentación para poder instalar el sistema operativo en nuestro entorno de trabajo, y configurarlo para

poder usarlo en la Raspberry PI.

2.6 Librerías de uso del BCM2835

Existen diversas librerías que están configuradas para el chipset BCM2835 que utiliza la Raspberry Pi y se utilizan para controlar las funciones del bus I2C. Esto nos llevó al análisis de las librerías más importantes que trabajan sobre el chipset BCM2835, algunas de ellas fueron:

- Librería WiringPi [9], es una librería programada en C y su objetivo es tener una única plataforma común y un conjunto de funciones para acceder a la GPIO de la Rasberry Pi. WiringPi utiliza su propio sistema de numeración de pines.
- Librería BCM2835 [10], es una librería escrita en C para Raspberry Pi. Proporciona acceso a la GPIO y otras funciones de entrada y salida en el chip Broadcom BCM 2835, permitiendo el acceso a los veintiséis pines de la Raspberry PI para que pueda controlar e interactuar con varios dispositivos externos.

La elección de la librería se basará en una comparación de ambas librerías en el siguiente apartado.

La librería que utilicemos nos permitirá manejar los pines la GPIO de la Raspberry Pi y además nos proporcionará funciones para comunicarnos con el adaptador a través del bus I2C.

En cualquier caso, será necesario consultar la documentación de BCM2835 ARM Peripherals Manual, que se puede acceder desde la documentación del sitio web de la propia Raspberry Pi [11].

2.7 Bus I2C

El bus I2C (del inglés, *Inter-Integrated Circuit*), es un estándar que permite la comunicación o intercambio de información entre dispositivos de baja velocidad como microcontroladores, convertidores, memorias, interfaces de entrada y salida, y otros periféricos similares en sistemas embebidos. Fue creado por la empresa Philips y ahora es utilizado por los principales fabricantes.

El I2C utiliza un sistema de bus de tipo maestro-esclavo. El maestro es quien realiza siempre la inicialización de la transferencia de datos y el esclavo es quien reacciona.

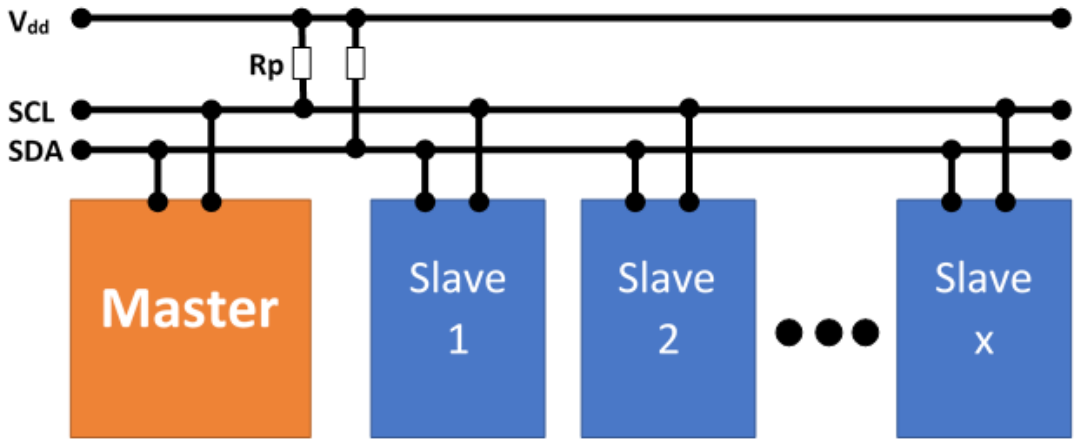


Figura 5: Esquema eléctrico de la tecnología I2C

En la figura 5 se muestra el esquema eléctrico del bus I2C. Este bus está formado únicamente por dos líneas de señal: una correspondiente al reloj (SCL, Serial Clock) y otra a los datos (SDA, Serial Data).

Como hemos indicado antes el dispositivo maestro es quien inicia la comunicación y quien establece la señal de reloj. Dependiendo del modo en que se utilice, se puede modificar esta señal siempre y cuando sea compatible con la interfaz del maestro.

La comunicación entre el maestro y esclavo se produce con una secuencia o señal de inicio en el bus. Esta secuencia junto con la secuencia de parada son especiales, ya que son los únicos casos que se permite que la SDA cambie cuando la línea SCL está alta. Cuando se transmiten datos la línea SDA tiene que permanecer estable, mientras que la línea SCL sigue alta. Estas secuencias señalan el inicio y el final de una transacción con los dispositivos esclavos.

Indicar que una unidad de datos consta de 8 bits, dependiendo del protocolo pueden ser leídos como una dirección o como un valor, y además tiene un bit de ACK, el bit de confirmación.

Cada dispositivo esclavo I2C tiene una dirección de 7 bits que va a ser única en el bus. Existen también dispositivos de 10 bits, aunque solo en casos específicos. Las direcciones de 7 bits permiten que se pueden poner hasta 128 dispositivos sobre un bus I2C. Añadir que siempre se envía un bit adicional, este bit extra, se utiliza para informar al dispositivo esclavo si el dispositivo maestro va a escribir o va a leer datos desde él. Si el bit (R/W) es cero, el maestro está escribiendo en el esclavo y si es uno, el maestro está leyendo del esclavo [11].

En nuestro trabajo la comunicación entre la Raspberry Pi y el adaptador se realiza a través del bus I2C.

2.8 Emulador QEMU

El objetivo principal de QEMU es emular un sistema operativo dentro de otro sin tener que hacer particiones del disco duro, utilizando para su ubicación cualquier directorio dentro de éste.

Este emulador hace de sistema informático completo, incluyendo procesador y periféricos. Además de emular diferentes sistemas operativos como pueden ser Linux o Microsoft Windows, también admite la emulación de plataformas hardware como x86, AMD, Sparc, ARM y alguna más. Por ello, este emulador puede simular la Raspberry Pi que tiene un procesador ARM [12].

En nuestro trabajo utilizaremos QEMU para ejecutar el programa compilado previamente con MaRTE OS, y así ver el resultado de la ejecución.

2.9 Doxygen

Doxygen es una herramienta que nos permite la generación de la documentación de nuestro trabajo realizado en el lenguaje de programación C. Esta herramienta es compatible con C, C++, Java, Python, PHP y algún lenguaje de programación más. Es muy utilizado, ya que hay mucha documentación en la web y permite organizar nuestra documentación de forma sencilla. Doxygen viene de documento(dox) y generador (gen), es decir un generador de documentación para código fuente [13].

En el anexo V aparecen unos extractos de la documentación generada por esta herramienta.

3. DESCRIPCIÓN DEL SISTEMA

3.1 Plataforma

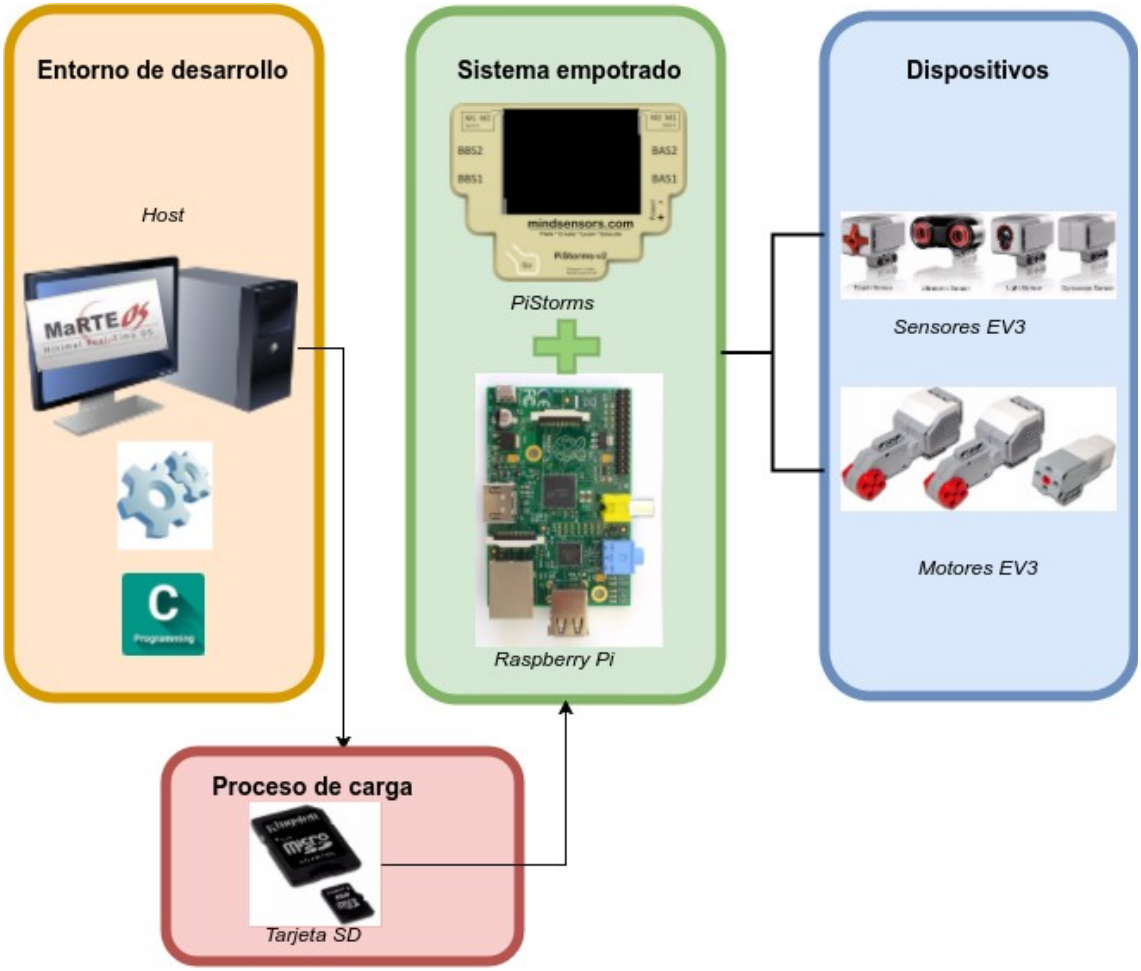


Figura 6: Esquema general del proceso del sistema

A continuación, describiremos y explicaremos como está compuesto el sistema en general, tratando los elementos y dispositivos que lo constituyen así como las conexiones entre ellos.

La figura 6 muestra el esquema general de la plataforma creada. Como se observa, esta plataforma consta de cuatro módulos importantes: un entorno de desarrollo, el proceso de carga, el sistema empotrado y los dispositivos. Los elementos constituyentes de estos módulos son los siguientes:

- *Host* con sistema operativo Linux y una instalación de MaRTE OS que genera aplicaciones para la placa Raspberry Pi.
- Tarjeta de memoria SD de 16 GB.

- Placa Raspberry Pi modelo B.
- Adaptador PiStorms compatible con Lego Mindstorms EV3 y NXT.
- Sensores y actuadores Lego Mindstorms EV3.

Tal y como se observa en la figura 6, los elementos principales de la plataforma están relacionados unos con otros. La primera relación que observamos se realiza entre nuestro entorno de desarrollo y el sistema empotrado a través del proceso de carga. Este proceso de carga se realiza mediante una tarjeta SD y permite el paso de aplicaciones generadas en el *host* al sistema empotrado.

La conexión entre la placa Raspberry Pi y el adaptador PiStorms se realiza a través de la GPIO, a través del cual se realizarán las llamadas para controlar el adaptador además de proporcionarle energía para poder utilizarlo.

La comunicación entre el sistema empotrado y los dispositivos de Lego Mindstorms se establece mediante cables RJ12 que conecta los sensores y actuadores con los puertos del adaptador PiStorms.

Estos son los elementos principales que se utilizan para crear la plataforma final junto con las conexiones necesarias para realizar la comunicación entre ellos. Todo esto formaría la parte hardware del sistema. Respecto al software, el elemento central es el sistema operativo MaRTE OS. Para ello, se ha utilizado un proyecto existente en el cual se porta dicho sistema operativo a la arquitectura ARM para Raspberry Pi, con el objetivo de poder ejecutar aplicaciones concurrentes en lenguaje C.

Dado que MaRTE OS está escrito principalmente en lenguaje Ada, se requiere un compilador compatible para arquitectura ARM. GNAT [14] proporciona un compilador cruzado compatible de 32 bits. Sin embargo, nuestro host ejecuta un Ubuntu de 64 bits por lo que se ha requerido instalar librerías adicionales para poder realizar la instalación de MaRTE OS correctamente. Por tanto, este compilador necesita librerías adicionales para poder hacer la instalación correctamente.

Con el sistema operativo MaRTE OS instalado y configurado con la arquitectura para la Raspberry PI ya tenemos el entorno listo para crear aplicaciones compatibles con la placa.

3.2 Entorno de desarrollo

El entorno de desarrollo se compone de un computador con un sistema operativo GNU/Linux sobre una arquitectura *x86*. Para poder crear código ejecutable para la placa Raspberry Pi desde este computador es necesario un

proceso de compilación cruzada, es decir, un compilador genera el ejecutable con las características específicas para la placa Raspberry Pi.

Como ya hemos indicado en el apartado anterior, mediante el compilador GNAT y la configuración de la arquitectura con la que MaRTE OS va a trabajar, en nuestro caso la específica para Raspberry Pi, se genera el ejecutable compatible con la placa Raspberry Pi.

Para ver el resultado de nuestra aplicación y comprobar que todo está correcto, existe un emulador QEMU que permite la emulación de plataformas hardware como la que contiene para la placa Raspberry Pi. El emulador oficial no tiene soporte para emular la placa de la Raspberry Pi que utilizamos en el proyecto, por eso utilizaremos un proyecto realizado por *Torlus* [15].

La parte de emulación nos sirve para comprobar el funcionamiento de programas sencillos con MaRTE OS. Sin embargo, en nuestro trabajo se utilizan dispositivos que se conectan a la GPIO y cuyo funcionamiento no puede comprobarse desde el emulador.

El proceso de carga del ejecutable en la Raspberry Pi se realiza a través de una tarjeta de memoria SD. El formato de la tarjeta SD está compuesto por un sistema de ficheros *FAT* 32. Hemos utilizado el instalador del sistema operativo *NOOBS* en la tarjeta SD.

Para ejecutar la aplicación desarrollada, es necesario sustituir la imagen del núcleo del sistema operativo de la tarjeta SD por nuestro ejecutable compilado anteriormente. Mediante este cambio y con la tarjeta SD insertada en la placa Raspberry Pi, se realiza la carga de nuestro programa y se ejecuta la aplicación.

3.3 Uso de dispositivos Mindstorms en Raspberry Pi

3.3.1 Comparativa

Como se ha mencionado anteriormente, existen diferentes adaptadores que sustituyen al brick de Lego y es compatible con la Raspbery Pi. A continuación, se presenta una comparativa entre dos de las plataformas que analizamos para este trabajo.

Las plataformas BrickPi y PiStorms tienen las mismas conexiones para los actuadores, ambos presentan un total de cuatro puertos. A diferencia de los puertos para conectar los sensores, el BrickPi posee cinco puertos mientras que en el PiStorms solo es posible conectar cuatro sensores. Además, los dos son compatibles con los motores y actuadores de Lego MindStorms tanto para el kit de NXT como para el de EV3. Asimismo, ambos se conectan con la

Raspberry Pi mediante los pines de GPIO y forman una estructura compacta. Igualmente, los dos necesitan una fuente de alimentación de 9 Voltios para poder utilizar sensores y motores correctamente. La comunicación que soportan ambos adaptadores es mediante el bus I2C. Es decir, ambos adaptadores presentan unas características y especificaciones similares y cumplen con su función.

Entre las diferencias de ambos adaptadores, podemos destacar que el adaptador PiStorms incluye una pantalla táctil, mientras que el BrickPi no lo contiene. Otra diferencia es que el PiStorms añade un botón en el adaptador, que puede ser de gran utilidad para encender o apagar un robot, mientras que el adaptador BrickPi no lo tiene.

En la tabla 1 del Anexo I se ofrece una comparación detallada y las características de estas plataformas compatibles con el kit de Lego Mindstorms [8].

Para tomar la decisión final tuvimos en cuenta dos detalles importantes. En primer lugar, nos dimos cuenta de que había mayor cantidad de proyectos con BrickPi pero el PiStorms tenía una guía avanzada para el desarrollador que explicaba muy bien qué registros utilizar. Esto resulta especialmente relevante para el desarrollo de las librerías de comunicación con el adaptador. El segundo detalle importante es la pantalla que incluye el PiStorms, ya que para trabajos futuros puede ser interesante poder trabajar con ella mediante el desarrollo de una interfaz. Así que nuestra decisión final fue escoger el adaptador de PiStorms.

3.3.2 Adaptador PiStorms

Como ya hemos indicado antes, el adaptador PiStorms está diseñado para conectarse directamente con la placa Raspberry Pi mediante los pines GPIO. Para suministrarle energía al bloque formado por la Raspberry PI más el adaptador, necesitamos, una batería externa recargable que nos genere el voltaje necesario o como un pack de 7 pilas AA recargables conectadas en serie. Esta última opción será la elegida ya que eran los medios que teníamos en el laboratorio y genera el voltaje suficiente para que funcione.

Para el uso del adaptador se necesita un voltaje de 6.6 Voltios, pero si se va a usar sensores y motores, es necesario utilizar 9 Voltios. En caso de sobrepasar los 10.5 Voltios la pantalla táctil podría parpadear.

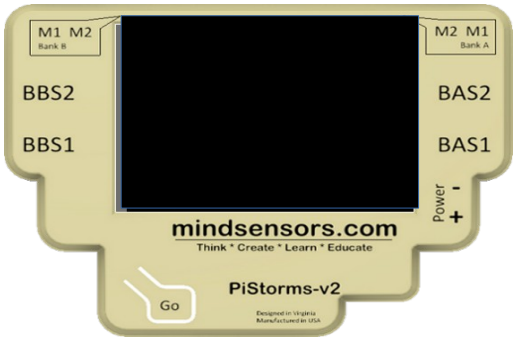


Figura 7: Adaptador PiStorms vista frontal

La figura 7 muestra los elementos más relevantes del adaptador [5]:

- Está compuesto por dos bancos (Bank-A y Bank-B), cada uno con puertos para conectar sensores y motores. Estos bancos tienen su propia dirección I2C. Cada banco contiene dos puertos para sensores y otros dos para motores (total cuatro motores y cuatro sensores), y son soportados tanto para los de Lego EV3 como para los de Lego NXT.
- Contiene un botón/interruptor. El botón debe ser presionado una vez en el encendido para establecer la conexión con la Raspberry Pi. También se puede apagar la Raspberry Pi apretando el botón durante aproximadamente diez segundos.
- PiStorms está equipado con una pantalla táctil LCD de 2,4 pulgadas.
- Tiene dos leds, uno en cada esquina superior, que se puede acceder a cada uno de ellos desde el respectivo banco, y así poder programarlo.
- El adaptador viene con un soporte que encaja perfectamente con la Raspberry Pi y es compatible para hacer una estructura con las piezas de Lego y poder integrarlo en cualquier robot.

En el Anexo II se puede ver cómo está organizado el adaptador con sus puertos y la estructura que tiene. Por últimos, PiStorms utiliza la tecnología I2C (que será desarrollada más adelante en la memoria) para leer y/o escribir información en sus registros.

4. DISEÑO DE LA LIBRERÍA

En este apartado se mostrará primero el diseño general del entorno creado y seguido se expondrá el diseño de la librería creada con la división de ficheros y las relaciones que tienen entre ellos.

4.1 Visión general

A continuación, vamos a mostrar la estructura que va a tener el entorno desarrollado para el sistema empotrado.

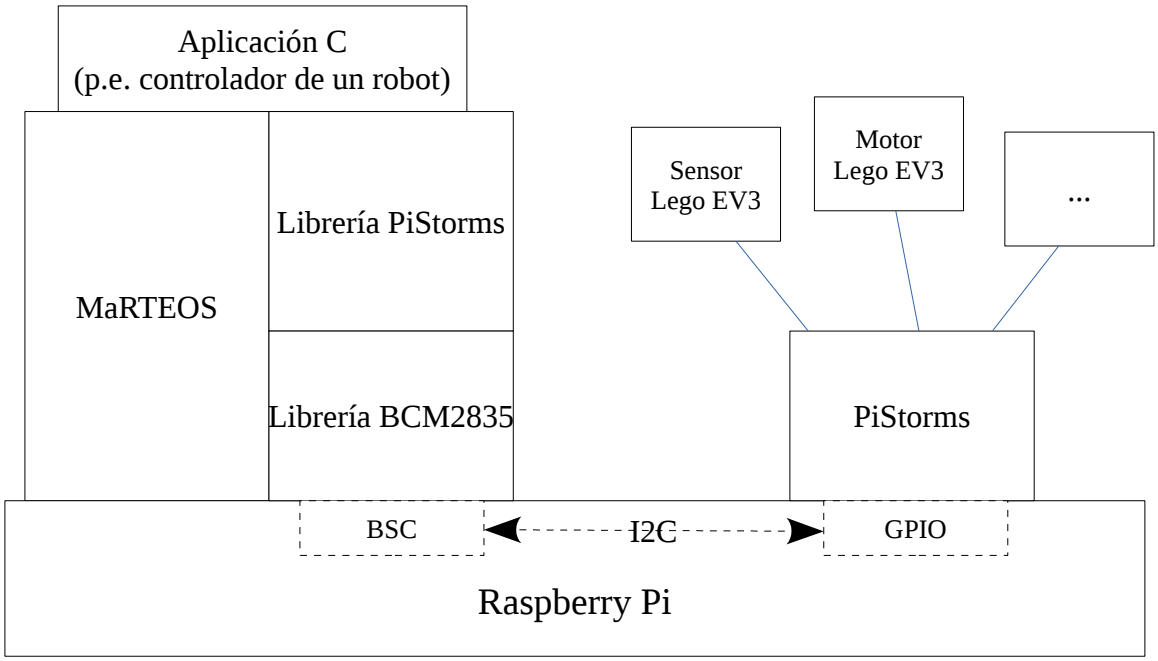


Figura 8: Diagrama de capas general para el sistema empotrado

En la figura 8 se muestra el diagrama de capas general. En la parte inferior del diagrama aparece la Raspberry Pi, sobre la cual se monta el resto de capas. Es decir, la Raspberry Pi va a ser un elemento base para nuestro trabajo. Además, existen dos partes bien diferencias que se comunican a través del bus I2C: el adaptador PiStorms y la aplicación junto con las librerías.

La comunicación I2C se realiza a través de los pines GPIO de la Raspberry Pi, en la cual se conectará al adaptador PiStorms (ver Figura 8). Una capa más arriba se encuentran los sensores y motores de Lego Mindstorms EV3, que se

conectarán al adaptador PiStorms mediante conectores tipo *RJ12*. Este tipo de conectores también son compatibles para los sensores y actuadores del kit Lego Mindstorms NXT, y se pueden utilizar las piezas para hacer la estructura final.

En la parte inferior izquierda de la figura 8 se encuentra el BSC (*Broadcom Serial Controller*), que implementa tres maestros independientes que tienen que estar en buses I2C separados, aunque solo trabajaremos con uno de ellos. A través del bus I2C se comunicará el adaptador PiStorms con la aplicación y las librerías creadas.

La nomenclatura de los pines y su función dependerá de la revisión del *GPIO Header* que contendrá nuestra Raspberry Pi. De acuerdo a la documentación, nuestra plataforma utiliza la revisión dos del modelo B. En el Anexo III mostramos la comparación de ambas revisiones de la GPIO Header (revisión 1 y revisión 2). La diferencia que hay entre estas versiones es que cada una utiliza un bus I2C diferente: en la revisión 1 se conecta a las líneas I2C SCL0 y SDA0, mientras que en la revisión 2 utiliza las líneas I2C SCL1 y SDA1.

En la parte central izquierda de la figura 8 se encuentra el sistema operativo MaRTE OS. Este sistema operativo lo tenemos instalado en nuestro *host* y configurado para generar aplicaciones compatibles con la placa Raspberry Pi.

Integrado en el sistema operativo se encuentran las librerías desarrolladas: por un lado, la librería referida al chip BCM2835, que será la capa de más bajo nivel y estará adaptado a MaRTE OS, y por otro lado, la librería que da soporte al adaptador PiStorms y los dispositivos (actuadores y sensores) de Lego Mindstorms. El diseño de ambas librerías se desarrollarán en el siguiente apartado.

Por último, en la parte superior de la figura 8 se encuentra la aplicación desarrollada en lenguaje. Esta aplicación hará referencia a la librerías PiStorms, que a su vez hace referencia a la librería BCM2835 que aporta las funciones para el control del bus I2C que permite utilizar la plataforma PiStorms junto con los sensores y actuadores.

4.2 Diseño de las librerías

En este apartado mostraremos el diseño y la estructura que hemos seguido para la creación de la llamada librería PiStorms. En el cual, daremos soporte a los sensores, motores y el propio adaptador que lo llamaremos *brick*. A continuación, se muestra la figura 9, en la cual se pueden apreciar los módulos en que se componen las librerías y la relación entre ellos.

La estructura del entorno desarrollado se compone de tres librerías principales: Librería BCM2835, Librería PiStorms y Aplicación. La estructura de cada librería y sus módulos son las siguientes:

Librería BCM2825:

- Se compone del módulo Bcm2835 y aporta las funciones para controlar la GPIO de la placa Raspberry PI y el bus I2C de la plataforma.

Librería PiStorms:

- Contiene el módulo pistorms que realiza operaciones como establecer el puerto y banco que se van a utilizar o iniciar el bus I2C. Hace referencia al módulo Bcm2835 de la librería BCM2835.
- Los módulos de los actuadores, sensores y del brick que aportan las funciones generales para manejar los actuadores, sensores y el brick PiStorms. Hacen referencia al módulo pistorms.
- Los módulos propios de cada sensor que dan soporte a cada uno de los sensores. Cada sensor tiene sus propias características. Hace referencia al módulo general de los sensores.

Aplicación:

- Esta librería se forma con el módulo *main* que contiene la aplicación final y la que aporta la lógica final del proyecto, haciendo referencia a los módulos necesarios de la librería PiStorms.

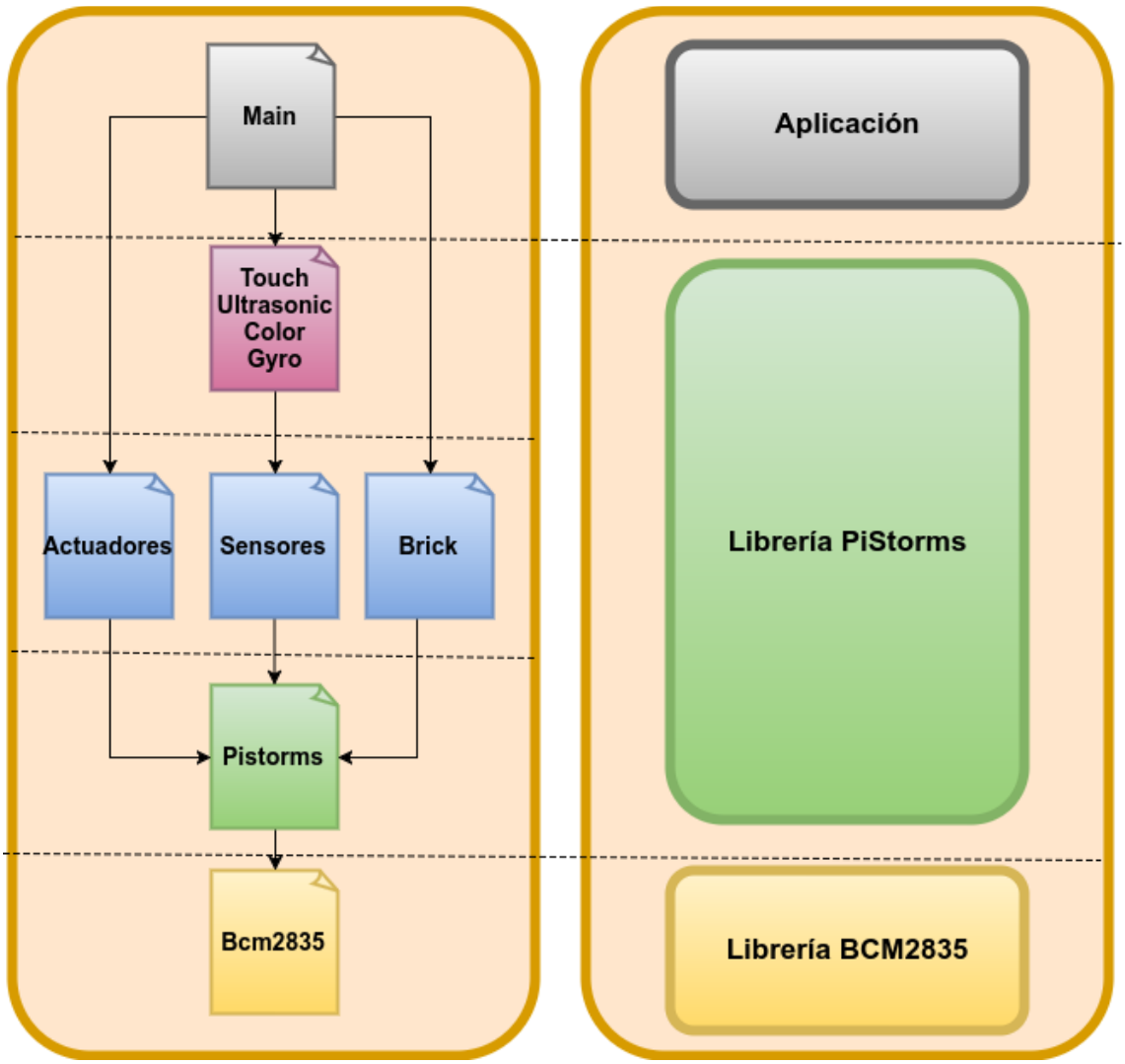


Figura 9: Diseño del entorno desarrollado

En la figura 9 aparece la relación entre los módulos del entorno desarrollado, junto con las librerías a las que pertenecen dichos módulos. En la librería BCM2835 se encuentra el módulo *bcm2835* la cual hemos adaptado para poder compilarlo con MaRTE OS. Este módulo contiene las funciones para utilizar el propio bus I2C mediante los pines de la GPIO de la Raspberry PI. La librería BCM2835 es la primera capa de nuestro entorno y se compone de un único módulo.

En una capa superior, aparece el módulo pistorms perteneciente a la librería PiStorms. Este módulo hace referencia al módulo B*cm2835* de la librería BCM2835, y contiene funciones básicas pero importantes, como inicializar la librería *bcm2835* y el bus I2C, a su vez también se puede terminar las operaciones de I2C.

Una función importante en nuestro trabajo es la que nos permite configurar el esclavo en el bus I2C y ésto lo hacemos mediante la función *set_active_bank(int connector_id)* en la cual comprobamos la dirección del banco actual y si es necesario cambiar de esclavo. Esta llamada es importante para los ficheros del siguiente nivel.

El siguiente nivel dentro de la misma capa, se encuentran los módulos de los sensores, actuadores y del propio brick PiStorms. El módulo del *Brick* y de los *actuadores* se pueden llamar directamente desde el programa *Main* como se muestra en la figura 9. Esto se debe a que el módulo del *Brick* ya contiene las funciones necesarias para poder utilizar el *brick* PiStorms: encender y apagar los leds, tocar y contar el número de veces que ha sido presionado el botón Go, saber si la pantalla táctil está siendo presionada y obtener el voltaje de la batería entre otras funciones. Esto se consigue mediante funciones creadas que hacen referencia al módulo pistorms dentro de la librería PiStorms.

El módulo referente a los motores se podría descomponer en dos ya que existen dos tipos de motores de Lego Mindstorms: Large Motor y Medium Motor. Pero éstos solo difieren en sus especificaciones, con lo cual se puede utilizar directamente desde este módulo. Es decir, las funciones que contiene dicho módulo como cambiar la velocidad, poner un objetivo de distancia o parar los motores van a funcionar para ambos motores. La única diferencia existente entre ellos es el tipo de frenado. Mientras el Large Motor tiene dos tipos de frenado (frenado brusco y frenado suave), el Medium Motor solo tiene la opción de frenado brusco. Si se utiliza la llamada para frenar suavemente el Medium Motor, éste frenará bruscamente.

Dentro del módulo sensores desarrollamos funciones que van a ser llamadas por los módulos específicos de cada sensor. Algunas de estas funciones nos proporciona la utilidad de configurar el tipo del puerto del adaptador PiStorms, obtener el modo actual que está configurado el sensor, poder cambiar el modo del sensor y leer los datos que el sensor está obteniendo. Este módulo se divide en cuatro submódulos como se ve en el diagrama. Estos submódulos corresponden a los sensores que hemos dado soporte. A diferencia de los motores, se han creado módulos específicos para cada sensor debido a que cada sensor tiene diferentes modos de actuar y leer datos. Por ejemplo, el giróscopo de Lego tiene dos modos de leer datos y además estos datos son de tipo entero, mientras que el sensor ultrasónico de Lego tiene tres modos diferentes y lee datos de tipo flotante.

Finalmente y como último nivel del diagrama, tenemos el módulo *Main* que pertenece a la capa de aplicación, y es el programa que realiza el programador. Este módulo está relacionado directamente con el módulo de los motores, el

módulo del brick y con el módulo de cada sensor de la librería PiStorms. Se incluirán los módulos de los sensores que se vayan a utilizar en el programa para poder utilizar las funciones de cada sensor.

Además, hemos desarrollado un módulo adicional llamado *internal*. Este módulo proporciona funciones que facilitan la depuración en nuestro entorno. Está basado en representaciones en pantalla de los códigos a la hora de realizar alguna escritura o lectura I2C. Abarca la capa de Librería PiStorms, desde el módulo pistorms hasta los módulos de los sensores. La librería BCM2835 tiene un sistema de depuración propio, con lo cual no hemos incluido el módulo *internal* en esta librería.

5. IMPLEMENTACIÓN DE LA SOLUCIÓN

En el siguiente apartado se explicará el desarrollo del entorno creado, complementando el diseño del apartado previo.

5.1 Adaptación de la librería BCM2835 a MaRTE OS

La necesidad de adaptar la librería BCM2835 ya ha sido comentada en apartados anteriores, ahora procedemos a tratar esos cambios realizados para hacerla compatible con MaRTE OS.

Los cambios han sido realizados sobre el fichero *bcm2835.c,* el archivo de cabecera *bcm2835.h* no ha sufrido ninguna modificación. Estos cambios se centran en eliminar toda la estructura de mapeo de dispositivos a ficheros que realizaba la librería original. Para el mapeo de los periféricos las funciones de la librería utilizan operaciones como: `open("/dev/gpiomem", O_RDWR | O_SYNC)`, está basado en un sistema operativo Linux con su sistema de ficheros propio, con lo cual todas estas partes no nos sirven y han sido eliminadas. Esta parte es eliminada debido a que el sistema operativo utilizado, MaRTE OS no trabaja con un sistema de ficheros, y para utilizar el mapeo en la memoria es necesario un sistema de ficheros. En nuestro caso, utilizamos las direcciones de los periféricos directamente sin realizar ningún mapeo previo en la memoria, con lo cual hemos quitado esta parte en todas las funciones que comenzaban realizando el mapeo.

No se van a utilizar todas las funciones que vienen en la librería, sino que vamos a utilizar todo lo referente al inicio de la GPIO con todas sus direcciones, que vienen perfectamente definidas e inicializadas, y haremos uso de la mayoría de las funciones que permiten utilizar el bus I2C. Algunas de las funciones I2C más relevantes se exponen a continuación:

- `bcm2835_i2c_setSlaveAddress(uint8_t addr)`. Nos permite indicar la dirección del esclavo para el bus I2C, y será el que tome el control de éste.
- `bcm2835_i2c_write(char* buf, uint32_t len)`. Se pasa como parámetro el buffer con el registro a escribir y los datos, junto con la longitud de dicho buffer.
- `bcm2835_i2c_read(char* buf, uint32_t len)`. Los parámetros que recibe son: el buffer donde se quiere recibir la información, y la longitud del buffer. Esta función se haría siempre después de un `bcm2835_i2c_write` ya que es así como se utiliza el bus I2C.

- `bcm2835_i2c_read_register_rs(char* regaddr, char* buf, uint32_t len)`. Esta función lee directamente la información de un registro. Acepta como parámetro el registro a leer, el buffer donde recibir la información, y la longitud de los datos a leer.

La comunicación a través de I2c sigue el siguiente proceso:

- Primero, establecer el esclavo I2C con el que se quiere realizar la comunicación. En nuestro caso tendremos dos esclavos uno por cada banco del PiStorms.
- Una vez establecido el esclavo, la llamada de `bcm2835_i2c_read_register_rs` permite leer la información de un registro. Esta función puede no ser compatible con algunos dispositivos, pero es equivalente a la siguiente combinación: primero escribiríamos el contenido del registro con la función `bcm2835_i2c_write` y seguido leeríamos ese valor con la función `bcm2835_i2c_read`.
- Y si quisiéramos escribir o modificar un registro se llamaría a la función `bcm2835_i2c_write`.

Existen funciones adicionales para el manejo del bus I2C y para suspenderse durante un intervalo de tiempo, pero no las utilizaremos. Para ello haremos una llamada a la función *nanosleep(const struct timespec *req, struct timespec *rem)*, en la cual especificaremos el tiempo en nanosegundos para generar una espera de tiempo. Esta función es necesaria por ejemplo entre un cambio de tipo del puerto de la plataforma PiStorms o a la hora de hacer un cambio de modo de funcionalidad de un sensor. Aunque para hacer más óptima la aplicación hemos intentado no introducir muchas esperas de tiempo, ya que esto puede hacer menos eficiente la aplicación.

La librería BCM2835 permite especificar la versión de la placa Raspberry Pi utilizada a través de una constante de configuración que en nuestro caso es la revisión 2 como se muestra en el Anexo III. La diferencia principal es la conexión de las líneas del bus I2C y los pines de la GPIO: mientras en la revisión 1 los pines 3 y 5 están configurados para conectarse a las líneas SDA0 y SCL0, en la revisión 2 estos pines se conectan a las líneas SDA1 y SCL1.

Una función importante y que permite poder utilizar la plataforma PiStorms es `int _set_active_bank(int connector_id)`. Esta función será llamada desde las funciones de los sensores y motores, ya que nos permite comparar la dirección del esclavo del bus I2C y no producir una sobrecarga de operaciones haciendo llamadas innecesarias. Es decir, si estamos trabajando con un sensor

del banco A del adaptador PiStorms, y queremos utilizar un motor del mismo banco no es necesario volver a configurar el esclavo. Sin embargo, si queremos utilizar un sensor o un motor del banco B habría que hacer un cambio de esclavo. Esta funcionalidad está proporcionada en dicha función.

El parámetro que recibe la función anterior se llama *connector_id* y es de tipo entero, esto es debido a que se ha implementado de manera que cada puerto se representa con un número: *BANK_A_PORT_1* (1)*, BANK_A_PORT_2 (*2)*, BANK_B_PORT_1* (3)*, BANK_B_PORT_2* (4)*.* Una vez que se sabe en qué puerto vamos a conectar el sensor o el motor, en función de que puerto sea se escribe o lee un registro u otro.

5.2 Implementación de la librería PiStorms

5.2.1 Visión general

La documentación de la plataforma PiStorms nos aporta la información suficiente para poder leer y modificar los registros necesarios utilizando las funciones de la librería. Con todo ello hemos podido desarrollar los drivers para los sensores y motores de Lego Mindstorms.

Se ha desarrollado un sistema de propagación de errores para detectar que tipo de error nos está surgiendo. Estos errores se han creado en el archivo *marte_pistorms.h* y se han llamado *PistormsCodes*. Estos códigos de error nos indicarán si hemos conectado bien o mal un sensor o motor por ejemplo, o si el ID leído es válido o no devolviendo un valor.

También hemos generado el ID de cada sensor leyendo previamente el registro que nos muestra la guía PiStorms. Se ha creado una función que nos retorna el ID del sensor y la comparamos con el ID correspondiente para detectar si se está leyendo correctamente. Por ejemplo, el ID del giróscopo es GYRO-RATE.

5.2.2 Implementación de los drivers de los sensores

Los drivers para los sensores se han desarrollado observando las funcionalidades que nos aportaba la guía de desarrollo de la plataforma PiStorms [16]. Algunas de estas funcionalidades que había que desarrollar son:

- Configurar el tipo del sensor para un puerto.
- Obtener el modo del sensor conectado en un puerto específico.
- Cambiar el modo del sensor en un puerto determinado.
- Leer los datos obtenidos por el sensor conectado en cierto puerto.

Estas funciones se encuentran en el módulo de los sensores y van a construir la base del entorno para poder utilizar los sensores. Cada sensor tendrá su propia librería ya que cada uno tiene funcionalidades diferentes. Además, la configuración del tipo de sensor es esencial para ajustar el puerto al que vamos a conectar el sensor. Existen diferentes tipos de sensor como aparecen en el Anexo IV.

En nuestra aplicación hemos dado soporte al *Gyro Sensor*, *Touch Sensor, Color Sensor* y *Ultrasonic Senso*r de Lego Mindstorms EV3. Estos sensores junto con el soporte proporcionado se describen brevemente a continuación.

Gyro Sensor EV3

Se trata de un sensor digital que detecta el movimiento de rotación en un solo eje. Como se puede observar en la figura 8, hay unas flechas en el propio sensor, si se gira el sensor en la dirección de las flechas, éste puede detectar la velocidad de rotación en grados por segundo. La velocidad máxima de giro que puede detectar el sensor es de 440 grados por segundo.

Funciones Principales	Descripción
`pistorms_sensor_gyro_configure`	Detecta si el sensor se ha conectado bien
`pistorms_gyro_set_mode`	Configura el modo del sensor (ANGLE o RATE)
`pistorms_gyro_read`	Obtiene los datos que lee el sensor

Tabla 1: Funciones principales del Gyro Sensor EV3

Figura 10: Gyro Sensor EV3

La tabla 1 contiene las funciones principales del Gyro Sensor, como por ejemplo configurar el modo de funcionamiento del sensor u obtener los datos que está midiendo.

También permite tener un seguimiento del ángulo de rotación total en grados. Tiene una precisión de +3 o -3 grados para un giro de 90 grados [17].

Touch Sensor EV3

También conocido como sensor de contacto, es un sensor analógico que puede detectar si se presiona el botón rojo o si se deja de presionar, además de saber cuántas veces ha sido pulsado. En la figura 11 aparece el sensor con el botón rojo.

A continuación se muestra la tabla 2 que contiene las funciones más

importantes para el Touch Sensor, como por ejemplo detectar si el sensor está correctamente configurado, detectar si el sensor ha sido presionado o contar cuántas veces ha sido presionado.

Funciones Principales	Descripción
`pistorms_sensor_touch_configure`	Detecta si el sensor se ha conectado bien
`pistorms_is_touched`	Detecta si el sensor ha sido presionado
`pistorms_num_touches`	Cuenta cuántas veces ha sido presionado
`pistorms_reset_touches`	Pone a cero el contador de veces tocado

Tabla 2: Funciones principales del Touch Sensor EV3

Figura 11: Touch Sensor EV3

En este sensor no hace falta hacer cambios de modo como en los otros, pero sí es necesario hacer un cambio de tipo de sensor para el puerto ya que es de tipo EV3 analógico, y por defecto el tipo de sensor para los puertos del PiStorms es de EV3 general. En la tabla 1 del Anexo IV se muestra el tipo de sensores que se pueden configurar en los puertos según la guía del desarrollador de PiStorms.

Ultrasonic Sensor EV3

El sensor ultrasónico es un sensor digital que puede medir la distancia a un objeto delante de él. La distancia máxima distancia que puede medir es 250 centímetros o 99 pulgadas. El sensor puede trabajar en centímetros o pulgadas. Si se trabaja con centímetros, la distancia detectable está entre 3 y 250 centímetros, con una precisión de +1 o -1 centímetros. Mientras que si el sensor está funcionando en pulgadas la distancia a detectar se encuentra entre 1 y 99 pulgadas, con una precisión de +0.394 pulgadas o -0.394 pulgadas. En la figura 12 se muestra el aspecto del sensor [17].

En la tabla 3, se recogen las principales funciones implementadas para este sensor, como obtener la distancia a un objeto o configurar el modo de funcionamiento del sensor.

Funciones Principales	Descripción
`pistorms_ultrasonic_set_mode`	Configura el modo del sensor (CENTIMETERS, INCHES o PRESENCE)
`pistorms_ultrasonic_read_distance`	Obtiene la distancia a un objeto en frente de él
`pistorms_ultrasonic_presence`	Detecta si existe otro sensor ultrasónico cerca

Tabla 3: Funciones principales del Ultrasonic Sensor EV3

Figura 12: Ultrasonic Sensor EV3

Además de estos dos modos de medir la distancia, existe un modo más que permite detectar otro sensor ultrasónico que esté funcionando cerca, este modo es conocido como presencia, en nuestra aplicación lo hemos llamado *Presence*.

Este sensor es muy útil para detectar objetos sin llegar a chocar con ellos, con lo que puede ser una buena opción para desarrollar un robot.

Color Sensor EV3

Es un sensor digital que puede detectar el color o la intensidad de la luz (ver figura 13). Este sensor se puede utilizar en tres modos diferentes: modo color, modo de intensidad de luz reflejada y modo de intensidad de luz ambiental.

En el modo de color o en nuestra aplicación *Measure Color*, el sensor de color reconoce siete colores: negro, azul, verde, amarillo, rojo, blanco y marrón.

En el modo de intensidad de luz reflejada, o en nuestra aplicación *Reflected Light* el sensor de color mide la intensidad de la luz reflejada a partir de una luz roja que emite el sensor. El sensor utiliza una escala de 0 (muy oscura) a 100 (muy clara).

En el modo de intensidad de luz ambiental o en nuestra aplicación *Ambient Light*, el sensor de color mide la intensidad de la luz que entra en el sensor desde su entorno, como la luz solar o el haz de una linterna. El sensor utiliza una escala de 0 (muy oscura) a 100 (muy clara) [17].

En la tabla 4, se muestran algunas de las principales funciones implementadas para el sensor de color, como medir la intensidad de luz o reconocer ciertos colores.

Funciones Principales	Descripción
`pistorms_color_set_mode`	Configura el modo del sensor (AMBIENT, REFLECTED o MEASURE
`pistorms_color_read_light`	Mide la intensidad de luz que entra en la cara del sensor
`pistorms_color_measure`	Reconoce hasta siete colores diferentes

Tabla 4: Funciones principales del Color Sensor EV3

Figura 13: Color Sensor EV3

Estos son los sensores que hemos dado soporte. Cada driver de los sensores es diferente ya que cada uno presenta diferentes características, por ejemplo, se ha visto como algunos tienen hasta tres modos de funcionar mientras otros tienen dos o funcionan con el modo por defecto.

También hay que indicar que cada sensor trabaja con sus propios tipos de datos, por ejemplo el Gyro Sensor lee datos de tipo *short*, mientras que el Ultrasonic Sensor lee datos de tipo *float*, si está en modo medición de distancia [18].

5.3.3 Implementación de los drivers de los motores

La implementación de los drivers de los motores ha sido realizada siguiendo los mismos pasos que para los sensores. En un principio se había pensado realizar un driver para el Large Motor y otro diferente para el Medium Motor, pero al realizar las pruebas de las funcionalidades de cada motor solo se encontró una diferencia. Esta diferencia es que el Medium Motor no tiene la opción de parar lentamente como lo hace el Large Motor, sino que siempre frena bruscamente. Con lo cual se decidió crear el módulo correspondiente al motor que contendrá todas las funciones para manejar los motores.

En la figura 14 aparece el Large Motor y en la figura 15 se muestra el Medium Motor, ambos pertenecientes al kit de EV3 [17]. Adicionalmente, se han probado los motores del kit NXT de Lego Mindstorms y funcionan perfectamente.

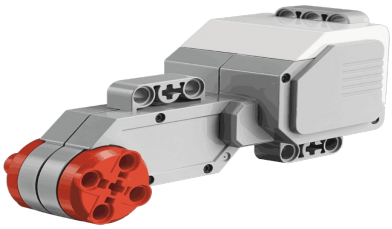


Figura 14: Large Motor EV3

Figura 15: Medium Motor EV3

Las principales funciones desarrolladas para los motores y siguiendo la guía del desarrollador de la plataforma PiStorms han sido:

- Obtener la posición actual del *encoder* del motor.
- Configurar una posición objetivo para el *encoder* del motor.
- Reiniciar todos los valores del *encoder* de los motores de un banco determinado.
- Añadir velocidad al motor.
- Indicar un tiempo específico en segundos en el cual el motor está operativo.
- Parar un motor específico lentamente o bruscamente.
- Parar ambos motores de un banco de forma sincronizada, lentamente o bruscamente.

SI hacemos una llamada de parar lentamente un motor de tipo Medium Motor, éste parará bruscamente ya que como se ha comentado anteriormente, este tipo de motor no contiene esta función.

Se ha creado una función interna llamada `set_sync_bank(int bank_id)` la cual sigue una lógica parecida a la función comentada en apartados anteriores `int _set_active_bank(int connector_id)`.

La función principal que permite correr nuestros motores es `int pistorms_motor_go(int connector_id, char go )`. Es necesario llamar esta función después de hacer la configuración previa de la velocidad, la posición del motor o el tiempo. En base a los parámetros que hemos configurado, el argumento a pasar puede ser: *SPEED_GO* para la velocidad, *TIME_GO* para el tiempo o *ENCODER_GO* para la posición del motor. Se pueden configurar dos parámetros a la vez. Por ejemplo, si queremos que el

motor funcione al 80% de su velocidad y durante 5 segundos, deberíamos hacer las llamadas a las configuraciones de ambos parámetros y después hacer la llamada a `pistorms_motor_go(int connector_id, char go )` introduciendo como segundo parámetro: *SPEED_GO* | *TIME_GO.* Con esta información la función escribirá el registro comando del motor con el valor necesario para que funcione correctamente.

Es importante saber que el parámetro del tiempo tiene preferencia sobre el de la posición del motor, con lo cual, si se han configurado previamente ambos parámetros, el motor solo responderá a la orden del tiempo establecido.

6. PRUEBAS Y ANÁLISIS DE LOS CONTROLADORES

6.1 Pruebas unitarias

Durante el desarrollo de la aplicación se han realizado pruebas según se iban implementando nuevas funcionalidades.

En primer lugar, se comprobó el manejo de la GPIO. Para ello se utilizó la librería de BCM2835 que permite el manejo de la GPIO. Se creó un programa llamado *blink.c* para comprobar si utilizábamos correctamente la GPIO con una aplicación compilada para Raspberry PI. Este programa consiste en encender un pin mediante la llamada `bcm2835_gpio_write(PIN, HIGH)` y después apagar dicho pin mediante la función `bcm2835_gpio_write(PIN, LOW)`, todo ello dentro de un bucle y generando tiempo entre las dos funciones. Para su comprobación conectamos un osciloscopio para medir la tensión de los pines utilizados.

Una vez que la librería de BCM2835 estaba adaptada correctamente, el siguiente paso era comprobar las funciones I2C mediante el adaptador PiStorms conectado a la GPIO. Para ello se creó un pequeño programa para obtener el *Vendor ID* del adaptador PiStorms, leyendo el registro específico para ello. En la tabla 2 del Anexo IV aparecen los registros de cada banco de PiStorms, incluido el *Vendor ID* mencionado.

Para realizar la lectura del registro se ha utilizado la función I2C específica para leer registros de la librería *bcm2835.* Previamente hemos configurado la dirección del esclavo I2C con la función: `bcm2835_i2c_setSlaveAddress(uint8_t addr)` con la dirección específica del banco a utilizar.

Una vez que conseguimos leer el resultado del *Vendor ID* seguimos con los demás registros para obtener el *Device ID* y el *Firmware Version*.

Comprobada la parte de las funciones I2C de la librería BCM2835 junto con el adaptador PiStorms, pasamos a hacer pruebas para los sensores. Se empezó comprobando el *Gyro Sensor EV3*, leyendo sus registros para obtener su id, el modo, los datos que estaba leyendo y escribiendo en el registro del modo para cambiar de modo del sensor. Todo esto se hace mediante las llamadas I2C comentadas anteriormente pero esta vez con los registros pertenecientes a los sensores EV3 como se muestra en a tabla 3 del Anexo IV.

Comprobaciones y pruebas similares se realizaron para comprobar el

Ultrasonic Sensor, Touch Sensor y *Color Sensor*, cada uno con sus especificaciones.

Se siguieron métodos de comprobación parecidos para los motores, ya que tienen un funcionamiento igual que los sensores, mediante las funciones I2C y los registros de la guía PiStorms se realizan las comprobaciones. Se probó la funcionalidad de la velocidad, la posición objetivo, los segundos, las paradas. Comprobamos que, si se pone un valor negativo en la velocidad, el motor gira en contra de la agujas del reloj mientras que si es positivos funciona a favor de las agujas del reloj. Se comprobó que la velocidad se mide en el porcentaje, es decir si se configura con un 100 quiere decir que el motor funcionará al 100% mientras que si ponemos un 80, éste funcionara al 80% de su velocidad. Para la posición objetivo del motor, se mide en grados, es decir si se pone 360 como parámetro, el motor girará una vuelta entera.

Además, se realizaron pruebas para el adaptador PiStorms, como por ejemplo encender y apagar los leds, tocar la pantalla o saber si se estaba presionando el botón Go. Al igual que los sensores y motores, el adaptador PiStorms tiene sus propios registros a los que se acceden mediante las funciones I2C.

6.2 Análisis de rendimiento

Se ha realizado un análisis de rendimiento para las principales funciones del entorno desarrollado. Con este análisis podíamos detectar algún desajuste en la programación de las funciones, ya que si detectábamos algún valor muy diferente al resto probablemente es que la programación no era la óptima.

Tener unos valores óptimos en tiempo, tanto para los sensores como los actuadores, permite realizar aplicaciones finales que sean eficaces y también facilita el desarrollo de algoritmos de control.

El análisis ha consistido en medir el tiempo que tardaban las funciones creadas mediante la función: `int clock_gettime(clockid_t clk_id, struct timespec *tp)`. Esta función está incluida en el fichero de Linux *time.h*. Recibe como primer parámetro un tipo de reloj, en nuestro caso hemos utilizado el *CLOCK_MONOTONIC*, y como segundo valor la dirección de una estructura *timespec* que se encuentra desarrollada en el fichero *time.h*.

A continuación se mostrarán unas tablas de resultados para las principales operaciones de nuestra aplicación. Estas tablas indican el tiempo medio utilizado para ejecutar completamente cada función. La media se calcula en base a cinco llamadas independientes.

MOTORES	Tiempo medio 5 operaciones (ms)
pistorms_motor_go	0,1984
pistorms_motor_get_pos	0,4484
pistorms_motor_set_pos	0,3644
pistorms_motor_reset_pos	0,1982
pistorms_motor_reset_all_parameters	0,1980
pistorms_motor_set_speed	0,1984
pistorms_motor_set_secs	0,1978
pistorms_motor_float	0,1964
pistorms_motor_float_sync	0,1964
pistorms_motor_brake	0,1984
pistorms_motor_brake_sync	0,1970

Tabla 5: Resultado de las pruebas de tiempo realizadas para los motores

SENSORES		Tiempo medio 5 operaciones (ms)
ULTRASONIC		
pistorms_sensor_ultrasonic_configure		11,5142
pistorms_ultrasonic_set_mode		10,3408
pistorms_ultrasonic_presence		0,4506
pistorms_ultrasonic_read_distance	cm	0,4518
	in	0,4524
GYRO		
pistorms_sensor_gyro_configure		11,5112
pistorms_gyro_set_mode		10,3418
pistorms_gyro_read		0,450400
TOUCH		
pistorms_sensor_configure_touch		11,5048
pistorms_is_touched		0,4484
pistorms_num_touches		0,4488
pistorms_reset_touches		0,1984

Tabla 6: Resultado de las pruebas de tiempo realizadas para los sensores

BRICK	Tiempo medio 5 operaciones (ms)
`pistorms_brick_led_On`	8,5778
`pistorms_brick_led_Off`	10,7194
`pistorms_brick_get_key_press_value`	0,1900
`pistorms_brick_reset_key_press_count`	0,0884
`pistorms_brick_touch_screen_X_asis`	0,1912
`pistorms_brick_touch_screen_Y_asis`	0,1924

Tabla 7: Resultado de las pruebas de tiempo realizadas para el brick PiStorms

Tras obtener estos resultados, podemos apreciar que donde más tiempo se utiliza es en la parte de los sensores, concretamente en la parte de obtener la configuración de los sensores. Ésto se debe a que esta función hace una llamada a `pistorms_port_set_type_sensor(int connector_id, int type)`, en la cual es necesario hacer una espera de tiempo para poder configurar el puerto correctamente y no se salte sin configurarlo.

Otra función que consume un tiempo superior a las demás, es la que realiza los cambios de modo de funcionamiento de los sensores. Esto se debe a la espera de tiempo que es necesario realizar para que el cambio de modo se complete correctamente y se puedan empezar a leer los datos en ese nuevo modo que va a funcionar el sensor.

En la parte del brick, las funciones que más tardan son las correspondientes a los leds del adaptador. Esto se debe a que debe configurar el banco con el que se va a trabajar y tiene que hacer una espera mínima de tiempo para que las operaciones internas de escritura I2C se completen.

7. DEMOSTRADOR FINAL

7.1 Estructura del robot

El entorno desarrollado nos permite crear un prototipo de robot que incluye los sensores y actuadores, las piezas del kit de Lego Mindstorms, la placa Raspberry Pi y el adaptador PiStorms.

En nuestro caso hemos creado la estructura de un coche, cuyo objetivo es que no choque con ningún objeto. Si detecta un objeto, frena, anda para atrás y gira continuando con su recorrido.

Para construir la estructura se han utilizado dos ruedas tipo oruga conectadas a dos motores que formará la base del coche. En la parte frontal hemos instalado un *Ultrasonic Sensor* y un *Touch Sensor*, para poder detectar objetos delante del coche.

La parte céntrica del coche está formada por la Raspberry Pi conectada a la plataforma PiStorms, el cual contiene una carcasa compatible con las piezas de Lego Mindstorms con lo cual nos facilitó hacer la estructura más sólida. El cargador de pilas se encuentra sujeto por la parte de abajo de la carcasa PiStorms.

Con las conexiones de los motores y los sensores a la plataforma PiStorms se completa la estructura del coche. En la figura 16 se muestra la estructura final del robot.

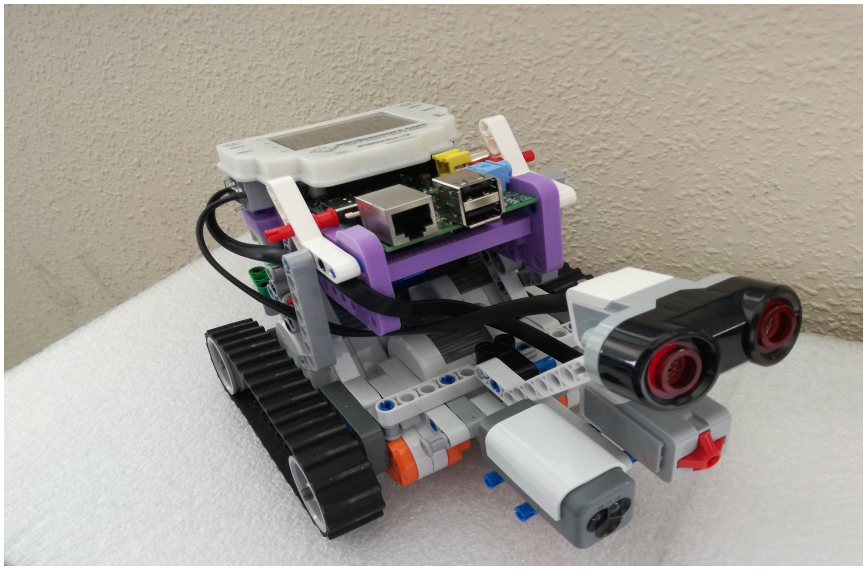


Figura 16: Demostrador del coche utilizando los sensores y actuadores de Lego

7.2 Descripción de la aplicación

El objetivo de la aplicación es dar al coche la capacidad de frenar cuando detecte un objeto delante de él y cambie de dirección para no colisionar con el objeto. Los objetos son detectados mediante el Ultrasonic Sensor y el Touch Sensor.

A continuación, se muestra el lazo principal del programa en pseudocódigo:

```
//Inicialización
Inicializar bcm2835 y bus I2C
Conectar Ultrasonic Sensor y Touch Sensor
MIENTRAS no se pulsa el botón GO
        Configurar velocidad de los motores para ir hacia delante
        Inicializar motores en modo velocidad.
        //Detectar un objeto
        SI distancia en centímetros < 6.0 ó Touch Sensor es tocado ENTONCES
                Frenado de MOTORES
                //El coche se mueve hacia atrás durante dos segundos
                Configurar velocidad de los motores para ir hacia atrás
                Configurar tiempo de 2 segundos a los motores
                Inicializar motores en modo velocidad y tiempo
                Dormir 2 segundos
                Frenado de motores
                //El coche comienza a girar
                Configurar velocidad de los motores para girar
                Configurar tiempo de 2 segundos a los motores
                Inicializar motores en modo velocidad y tiempo
                Dormir 2 segundos
        FIN SI
FIN MIENTRAS
 //Parada de motores y desconexión
```

Como se muestra en el pseudocódigo, la aplicación se ha diseñado para que los motores del coche funcionen a un porcentaje de su velocidad mientras no detecte ningún objeto. Para implementar esta funcionalidad se ha utilizado primero la instrucción `pistorms_motor_set_speed(MOTOR_1,50);` hacemos lo mismo con el MOTOR_2, seguido llamamos a la instrucción `pistorms_motor_go(MOTOR_1, SPEED_GO);` para que el motor se ponga en movimiento, también lo hacemos para el segundo motor.

Ahora crearemos la condición para saber si se debe cambiar la dirección del coche se basa en la detección de objetos a través del *Ultrasonic Sensor* y con el *Touch Sensor*. Hemos configurado que detecte a una distancia de 6 centímetros. Dentro de esta condición realizamos la parada sincronizada de los motores mediante la función `pistorms_motor_brake_sync(MOTORS_BANK_B)`. Seguido indicamos una velocidad a ambos motores y en dirección opuesta para volver para atrás. Se vuelve a llamar a la función de parada sincronizada y se realiza el giro del coche. El giro se forma configurando un motor a una velocidad positiva mientras que para el segundo motor se configura la velocidad negativa. Así se crea un giro más uniforme y continuo.

El algoritmo se ejecuta dentro de un bucle `while(! pistorms_brick_get_key_press_value())`, es decir hasta que no se presione el botón Go del adaptador PiStorms no se sale del bucle. Por lo tanto, el coche mantendrá una velocidad constante y en la misma dirección en caso de que nunca se encuentre un objeto. Si se pulsa el botón Go, se ha configurado para que se paren los motores y encender las luces rojas de los leds.

8. CONCLUSIONES Y TRABAJOS FUTUROS

El objetivo principal del proyecto era el de crear un entorno de desarrollo para dar soporte a los sensores y actuadores de Lego Mindstorms utilizando el sistema operativo MaRTE OS con una placa Raspberry Pi. Para llevar a cabo este trabajo, se realizó un estudio previo entre las diferentes plataformas que permiten conexiones con los dispositivos de Lego Mindstorms. Tras realizar una comparación entre ellos, se eligió el adaptador PiStorms.

Para utilizar dicho adaptador se estudiaron librerías que permitan el uso del bus I2C. Se analizó que librería era la más idónea para poder ser utilizada con MaRTE OS y finalmente se eligió la librería de BCM2835.

Una vez elegida la librería con la que íbamos a trabajar, se realizó la adaptación de ésta para el sistema operativo MaRTE OS, se realizaron los cambios necesarios y se consiguió dar funcionalidad a los pines GPIO de la Raspberry Pi junto con las funciones para controlar el bus I2C.

Se continuó con la realización de la librería Pistorms para formar la aplicación final. La aplicación se desarrolló siguiendo el diseño del apartado 4.2, en la cual se dan soporte a los sensores y actuadores además del propio *brick*, utilizando las funciones I2C de la librería BCM2835.

Finalmente creamos un demostrador en forma de coche para probar todo el sistema desarrollado y comprobar las diferentes funcionalidades de los dispositivos utilizados.

Respecto a proyectos futuros, se puede crear una interfaz para dar soporte a la pantalla del adaptador PiStorms, y poder usarla como consola de salida. Además, en nuestro proyecto se han utilizado los sensores y actuadores preferentemente de Lego pero el adaptador PiStorms soporta otro tipos de sensores y se podría crear nuevos soportes para nuevos sensores.

Existe una cámara que funciona con tecnología I2C y se utiliza en los kits de Lego. Podría ser interesante crear un entorno para dar soporte con MaRTE OS utilizando el adaptador PiStorms.

Además, se podrían crear prototipos nuevos de robots, como puede ser una cinta de montaje imitando a una cadena de una fábrica o un brazo robótico.

9. BIBLIOGRAFÍA

[1]: The LEGO group, 31313 MINDSTORMS EV3, 2017, https://www.lego.com/es-es/mindstorms/products/mindstorms-ev3-31313
[2]: , Los robots toman las aulas como nuevo recurso para la enseñanza, 2016, http://www.20minutos.es/noticia/2644370/0/robot-aula-educacion/nuevo-recurso/ensenanza/
[3]: Comunidad de usuarios, Foro oficial para Raspberry Pi. , 2017, https://www.raspberrypi.org/forums/
[4]: Arduino, Sitio oficial de Arduino, 2017, https://www.arduino.cc/
[5]: mindsensors.com, Sitio web adaptador PiStorms, 2005-2015, http://www.mindsensors.com/content/78-pistorms-lego-interface
[6]: Dexter Industries, Sitio web plataforma BrickPi, , https://www.dexterindustries.com/brickpi/
[7]: Universidad de Cantabria, Sitio web Sistema Operativo MaRTE OS, 2000-2017, http://marte.unican.es/
[8]: LEGO Group, Sistema operativo ev3dev, , http://www.ev3dev.org/
[9]: WiringPi, Información de librería Wiring Pi, 2017, http://wiringpi.com/
[10]: Broadcom Corporation, Periféricos ARm BCM2835, 2012
[11]: I2C info, Documentación I2C, 2017, http://i2c.info/
[12]: , Emulador QEMU para Raspberry PI, 2017, http://www.qemu.org/
[13]: Doxygen, Documentación de Doxygen, 2016, http://www.stack.nl/~dimitri/doxygen/
[14]: AdaCore, Sitio oficial de AdaCore GNAT, 2016, http://libre.adacore.com/tools/gnat-gpl-edition/
[15]: Torlus, QEMU para Raspberry Pi, , https://github.com/Torlus/qemu/tree/rpi
[16]: minsensors.com, Guía de desarrollo PiStorms, 2016
[17]: LEGO group, Guía de uso Lego Mindstorms EV3, 2017
[18]: ev3dev.org, Información de los sensores y motores EV3, 2014-2017, http://docs.ev3dev.org/projects/lego-linux-drivers/en/ev3dev-jessie/

ANEXOS:

ANEXO I:

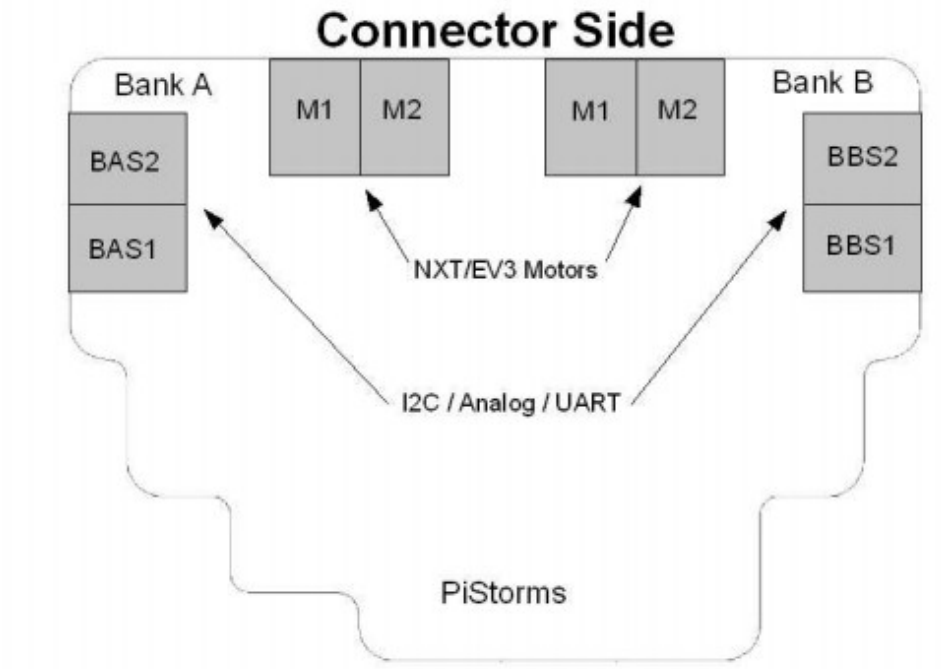
Tabla de comparación de plataformas.

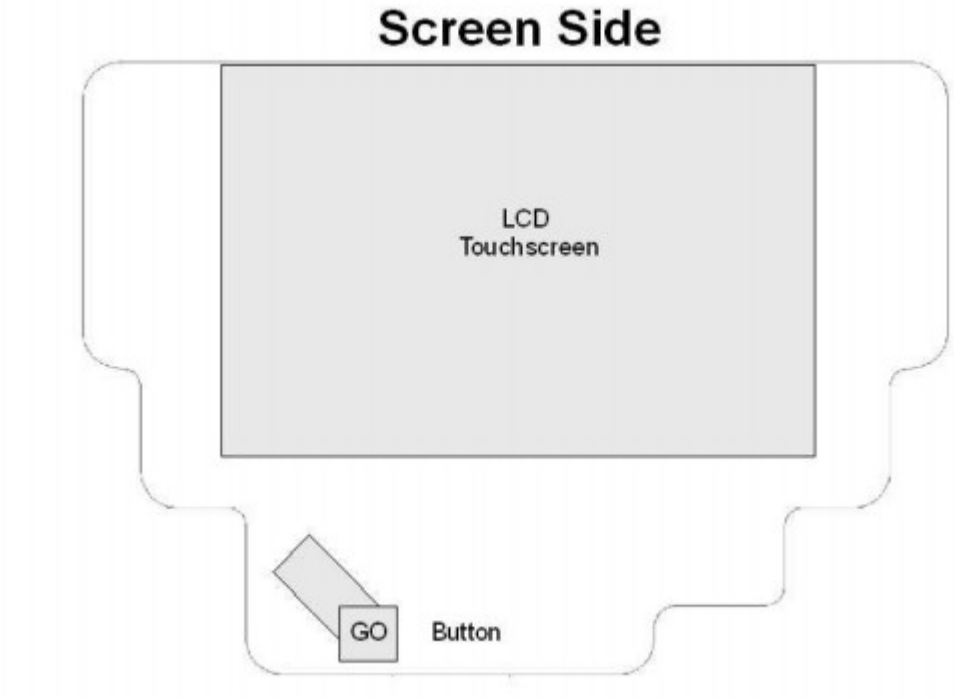
Fabricante		Lego	Dexter industries			mindsensors.com
Nombre del Modelo		EV3	BrickPi	BrickPi+	BrickPi3	PiStorms
Número del modelo		31313(Retail set) 45544(Education Set) 45500(EV3 Brick)	v1.7.3	v2.8	v3.2.1	PiStorms PiStorms-v2
CPU compatible		N/A	Raspberry Pi Zero Raspberry Pi Model A/A+/B/B+ Raspberry Pi 2 Model B Raspberry Pi 3 Model B			
Pantalla	**Resolution**	178x128 pixels	N/A			320x240 pixels
	Color Depth	1 bpp				16 bpp
	Backlight	No				Sí, siempre encendida
	Touchscreen	No				Sí
Botones	**Cantidad**	6	0			1
	Enter	Sí	No			Sí
	Up	Sí	No			No
	Down	Sí	No			No
	Left	Sí	No			No
	Right	Sí	No			No
	Back	Sí	No			No
LEDs		2-Rojo/Verde	2 - Azul		1-Amarillo(Ámbar)	2-Rojo/Verde/Azul
Altavoz		Sí	No			No
Puertos de	**Cantidad**	4	4 + 1 I2C	4	4 + 1 Grove	4

entrada					I2C	
	Detección automática	Sí	No			No
	Sensores EV3	Sí	Sí			Sí
	Sensores NXT	Sí	Sí			Sí
Puertos de Salida	**Cantidad**	4	4			4
	Detección automática	Sí	No			No
Indicador de batería	**Voltage**	Sí	No	Sí	Sí	Sí
	Corriente	Sí	No			No

ANEXO II:

Estructura del adaptador PiStorms.





ANEXO III:

Nomenclatura de la GPIO para Raspberry PI.

Raspberry Pi B
Rev 1 P1 GPIO Header

	Pin No.		
3.3V	1	2	5V
GPIO0	3	4	5V
GPIO1	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO21	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7

Raspberry Pi A/B
Rev 2 P1 GPIO Header

	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7

Key

Power +	UART
GND	SPI
I²C	GPIO

ANEXO IV:

Tabla1: Tipos de sensor compatibles con PiStorms.

Mode	Value	Defined Variable for Arduino Code
None	0	PS_SENSOR_TYPE_NONE
Switch	1	PS_SENSOR_TYPE_SWITCH
Analog	2	PS_SENSOR_TYPE_ANALOG
Light Reflected	3	PS_SENSOR_TYPE_LIGHT_ACTIVE
Light Ambient	4	PS_SENSOR_TYPE_LIGHT_INACTIVE
I2C	9	PS_SENSOR_TYPE_CUSTOM
Color Full	13	PS_SENSOR_TYPE_COLORFULL
Color Red	14	PS_SENSOR_TYPE_COLORRED
Color Green	15	PS_SENSOR_TYPE_COLORGREEN
Color Blue	16	PS_SENSOR_TYPE_COLORBLUE
Color None	17	PS_SENSOR_TYPE_COLORNONE
EV3 Switch	18	PS_SENSOR_TYPE_EV3_SWITCH
EV3	19	PS_SENSOR_TYPE_EV3

Tabla2: Conjunto de registros para cada banco de PiStorms.

Register	Read	Write
0x00-0x07	Firmware version - *Vxxxx*	-
0x08-0x0f	Vendor Id - *mindsnsrs*	-
0x10-0x17	Device ID - PiStorms	-

Tabla 3: Registros para los sensores EV3.

	Read	Write
0x70	Sensor 1 Ready	-
0x71-0x80	Sensor 1 ID (up to 16 characters)	-
0x81	Sensor 1 Mode	Sensor 1 Mode
0x82	Sensor 1 Length	-
0x83-0xA2	Sensor 1 Data (up to 32 characters)	-
0xA4	Sensor 2 Ready	-
0xA5-0xB4	Sensor 2 ID (up to 16 characters)	-
0xB5	Sensor 2 Mode	Sensor 2 Mode
0xB6	Sensor 2 Length	-
0xB7-0xD6	Sensor 2 Data (up to 32 characters)	-

ANEXO V

Extractos sacados de la documentación del sistema, generada por Doxygen.

marte_pistorms_sensor_color.c File Reference

Driver for control the Color of EV3 Sensor.

```
#include "marte_pistorms.h"
#include "marte_pistorms_sensors.h"
#include "marte_pistorms_sensor_color.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include "marte_pistorms_internal.h"
```

Functions

int **pistorms_sensor_color_configure** (int connector_id)
Detects if the Color Sensor is connect correctly.

int **pistorms_color_set_mode** (int connector_id, int mode)
Configure the mode of the Color Sensor.

int **pistorms_color_measure** (int connector_id)
Color sensor recognizes seven colors.

int **pistorms_color_read_light** (int connector_id, int mode)
Color sensor can measure the intensity of light that enters the small window on the face of the sensor.

Variables

char * **read_data**

Detailed Description

Driver for control the Color of EV3 Sensor.

Author:

Carlos Ayerbe González

Date:

8 Feb 2017

Version:

1.0

Definition in file **marte_pistorms_sensor_color.c**.

marte_pistorms_sensor_color.h File Reference

Driver for control the Color of EV3 Sensor.

Macros

#define **COLOR_SENSOR_ID** "COL-REFLECT"
#define **REFLECTED_LIGHT** 0
#define **AMBIENT_LIGHT** 1
#define **MEASURE_COLOR** 2
#define **REFLECTED** 0
#define **AMBIENT** 1

Functions

int **pistorms_sensor_color_configure** (int connector_id)
Detects if the Color Sensor is connect correctly.

int **pistorms_color_set_mode** (int connector_id, int mode)
Configure the mode of the Color Sensor.

int **pistorms_color_read_light** (int connector_id, int mode)
Color sensor can measure the intensity of light that enters the small window on the face of the sensor.

int **pistorms_color_measure** (int connector_id)
Color sensor recognizes seven colors.

Detailed Description

Driver for control the Color of EV3 Sensor.

Author:

Carlos Ayerbe González

Date:

28 Mar 2017

Version:

1.0

This is a C library for Raspberry Pi (RPi). It provides control over color sensor. This sensor is a digital sensor that can detect the color or intensity of light that enters the small window on the face of the sensor. This sensor can be used in three different modes: Color Mode, Reflected Light Intensity Mode, and Ambient Light Intensity Mode.

Definition in file **marte_pistorms_sensor_color.h**.

Module Documentation

I2C access

Functions
int **bcm2835_i2c_begin** (void)
void **bcm2835_i2c_end** (void)
void **bcm2835_i2c_setSlaveAddress** (uint8_t addr)
void **bcm2835_i2c_setClockDivider** (uint16_t divider)
void **bcm2835_i2c_set_baudrate** (uint32_t baudrate)
uint8_t **bcm2835_i2c_write** (const char *buf, uint32_t len)
uint8_t **bcm2835_i2c_read** (char *buf, uint32_t len)
uint8_t **bcm2835_i2c_read_register_rs** (char *regaddr, char *buf, uint32_t len)
uint8_t **bcm2835_i2c_write_read_rs** (char *cmds, uint32_t cmds_len, char *buf, uint32_t buf_len)

Detailed Description

These functions let you use I2C (The Broadcom Serial Control bus with the Philips I2C bus/interface version 2.1 January 2000.) to interface with an external I2C device.

Function Documentation

int bcm2835_i2c_begin (void)

Start I2C operations. Forces RPi I2C pins P1-03 (SDA) and P1-05 (SCL) to alternate function ALT0, which enables those pins for I2C interface. You should call **bcm2835_i2c_end()** when all I2C functions are complete to return the pins to their default functions

Returns:

1 if successful, 0 otherwise (perhaps because you are not running as root)

See Also:

bcm2835_i2c_end()

Definition at line 745 of file bcm2835.c.

void bcm2835_i2c_end (void)

End I2C operations. I2C pins P1-03 (SDA) and P1-05 (SCL) are returned to their default INPUT behaviour.

Definition at line 776 of file bcm2835.c.

uint8_t bcm2835_i2c_read (char * *buf*, uint32_t *len*)

Transfers any number of bytes from the currently selected I2C slave. (as previously set by

See Also:

bcm2835_i2c_setSlaveAddress)

Parameters:

in	*buf*	Buffer of bytes to receive.
in	*len*	Number of bytes in the buf buffer, and the number of bytes to received.

Returns:

reason see **bcm2835I2CReasonCodes**

Definition at line 900 of file bcm2835.c.

uint8_t bcm2835_i2c_read_register_rs (char * *regaddr*, char * *buf*, uint32_t *len*)

Allows reading from I2C slaves that require a repeated start (without any prior stop) to read after the required slave register has been set. For example, the popular MPL3115A2 pressure and temperature sensor. Note that your device must support or require this mode. If your device does not require this mode then the standard combined:

See Also:

bcm2835_i2c_write
bcm2835_i2c_read are a better choice. Will read from the slave previously set by **bcm2835_i2c_setSlaveAddress**

Parameters:

in	*regaddr*	Buffer containing the slave register you wish to read from.
in	*buf*	Buffer of bytes to receive.
in	*len*	Number of bytes in the buf buffer, and the number of bytes to received.

Returns:

reason see **bcm2835I2CReasonCodes**

Definition at line 975 of file bcm2835.c.