UNIVERSIDADE DA CORUÑA

Facultad de Informática
Departamento de Computación

Trabajo de fin de Grado
de Ingeniería Informática

Desarrollo de subsistema de interacción humano-máquina para el ROBOBO 2.0

Autor: Llamas Luaces, Luis Felipe
Director: Bellas Bouza, Francisco Javier

A Coruña, a 24 de agosto de 2016.

Información general

Título del proyecto:	**"Desarrollo de subsistema de interacción humano-máquina para el ROBOBO 2.0 "**
Clase de proyecto:	Proyecto de desarrollo en investigación
Nombre del alumno:	Llamas Luaces, Luis Felipe
Nombre del director:	Bellas Bouza, Francisco Javier
Miembros del tribunal:	
Miembros suplentes:	
Fecha de lectura:	
Calificación:	

Dr. Bellas Bouza, Francisco Javier

CERTIFICA

Que la memoria titulada **"Desarrollo de subsistema de interacción humano-máquina para el ROBOBO 2.0"** ha sido realizada por Luis Felipe Llamas Luaces con D.N.I. 48113017-F bajo la dirección del Dr. Francisco Javier Bellas Bouza. La presente constituye la documentación que, con mi autorización, entrega el mencionado alumno para optar a la titulación de Ingeniería en Informática.

A Coruña, a 24 de agosto de 2016.

Firmado:

Francisco Javier Bellas Bouza

Sigue rascando, hay millones de premios.

Agradecimientos

Resumen

ABSTRACT.tex

Palabras clave:

- √ Robótica
- √ Interacción humano robot
- √ Android

Índice general

Página

Índice de figuras

Figura **Página**

Capítulo 1

Introducción

Capítulo 2
Contextualización

En este capitulo se introducen los conceptos necesarios para la contextualización del trabajo. Se hablará sobre el proyecto D.R.E.A.M. [1] en el que se enmarca el proyecto ROBOBO, y sobre el trabajo previo en el robot.

2.1. Proyecto D.R.E.A.M.

Este trabajo fin de grado se enmarca dentro del proyecto europeo de investigación DREAM (Deferred Restructuring of Experience in Autonomous Machines) que se lleva a cabo en el Grupo Integrado de Ingeniería (GII). El objetivo global de dicho proyecto consiste en el desarrollo de una arquitectura cognitiva que permita a los dispositivos robóticos realizar tareas de aprendizaje y optimización de sus comportamientos. En particular, estas tareas se realizarán tanto durante el período de actividad como durante el período de inactividad de los robots (aprendizaje en fase de sueño) en analogía al funcionamiento del cerebro en lo seres humanos durante los períodos de sueño.

2.2. Plataforma ROBOBO 2.0

La primera versión de la plataforma fue desarrollada por el Grupo Integrado de Ingeniería de la UDC, el ROBOBO 1.0 (Figura 2.1) , sirvió como versión conceptual para comprobar la factibilidad del sistema plataforma + smartphone, y presentaba las siguientes características:

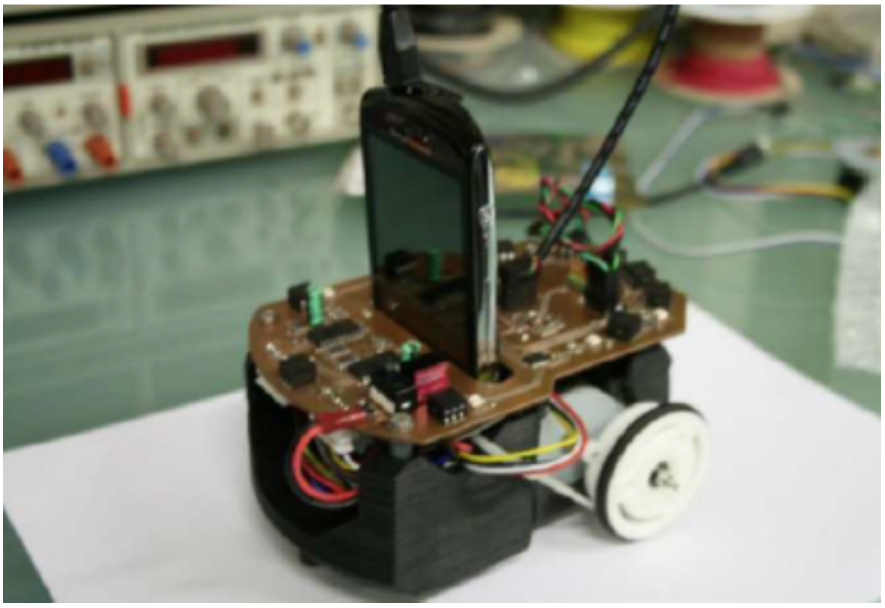

Figura 2.1: Robobo 1.0

- 9 LEDs (diodos emisores de luz) RGB para interactuar con el usuario, que cambiaban de color con la proximidad de objetos.
- 9 sensores IR de proximidad para proporcionar capacidad de movimiento autónomo: 2 en la parte frontal, 4 en los laterales y los últimos tres en la parte trasera; de manera que proporcionaba una visión general del entorno.
- 2 motores paso a paso (convierten impulsos eléctricos en desplazamientos angulares discretos, es decir, pueden avanzar un ángulo concreto en función de la señal recibida) que aplicaban movimiento a las ruedas, con un paso de 1/8, con gran precisión de giro.
- Compatibilidad con dispositivos Android.
- Capacidad de interacción con otros dispositivos ROS.
- Conexión USB para la comunicación con el teléfono inteligente.

La segunda versión, el ROBOBO 2.0 (figura 2.2) , fue desarrollada para solventar las carencias de la primera versión y añadir ciertas mejoras. Tras este rediseño, las características del robot pasaron a ser:

Figura 2.2: Robobo 2.0

- LEDs: pasan a ser 21, 9 de ellos formando una matriz de comunicación para dar una información avanzada mediante colores y formas.
- Motores de las ruedas: Pasan a ser motores de corriente continua, menos voluminosos y más eficientes energéticamente. Cada motor cuenta con un encoder magnético, que transforma el movimiento del motor en pulsos eléctricos, que pueden ser interpretados por la parte software del robot.
- Comunicación: Se elimina la comunicación USB a favor de una conexión Bluetooth, lo cual permite utilizar diferentes smartphones con distintas posiciones de los puertos.
- Plataforma universal para smartphones: Se incluye un adaptador universal para teléfonos moviles, montado sobre una plataforma motorizada que permite el movimiento del snartphone sobre el chasis del robot. Esta plataforma tiene dos grados de libertad, rotación e inclinación.
- Sensores: Modificadas tanto el tipo, la cantidad y la posición. Se colocarón en la siguiente disposición (figura 2.3):

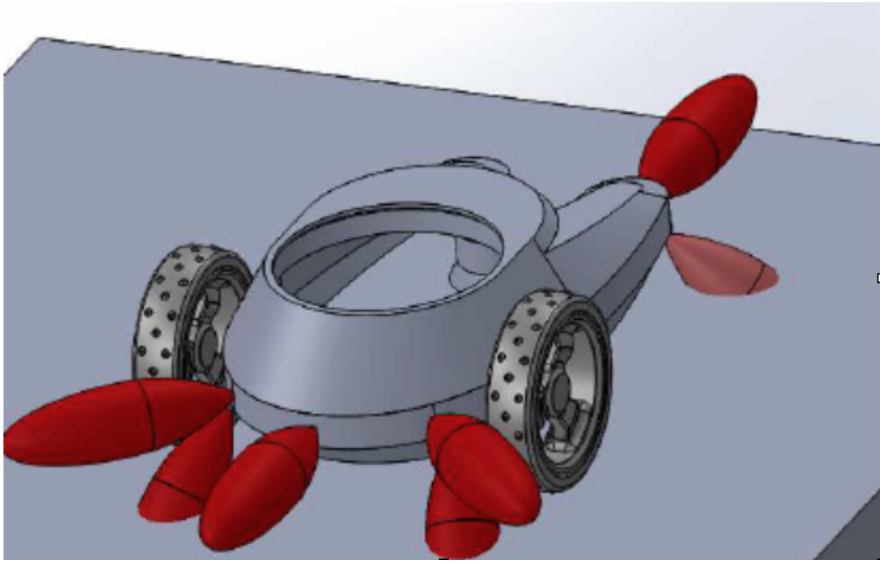

Figura 2.3: Disposición de los sensores en el ROBOBO 2.0

- 3 sensores para la detección de colisiones en la parte delantera; uno frontal y los otros dos cercanos a las ruedas, de tal forma que el ROBOBO no pudiera chocar, frontalmente con ninguna superficie.
- 2 sensores inclinados hacia abajo para la detección de caídas, en la parte delantera, colocados cerca de las ruedas, es decir, la zona con mayor peligro de caídas.
- 2 sensores de caídas y 2 de colisiones, de forma similar a los anteriores, en la zona anterior del robot.

- Unión sensores-microcontrolador: se han cambiado las resistencias pull-up anteriores por un multiplexor. Se consigue, así, un único elemento de muchas entradas y una única salida capaz de permitir la transmisión de una, y solo una, de las entradas hacia la salida.

- Microcontrolador: PIC32MX534F064H, de la familia de microcontroladores de 32 bits, económico, con alta capacidad de procesamiento, diversidad de interfaces de comunicación y multitud de puertos de entrada y salida; además de que su entorno de desarrollo y su compilador son gratuitos. Este microcontrolador se programa mediante el protocolo de comunicación I2C y logra el contacto entre sensores y microcontrolador mediante buses línea de reloj “SCL” y buses línea de datos “SDA”.

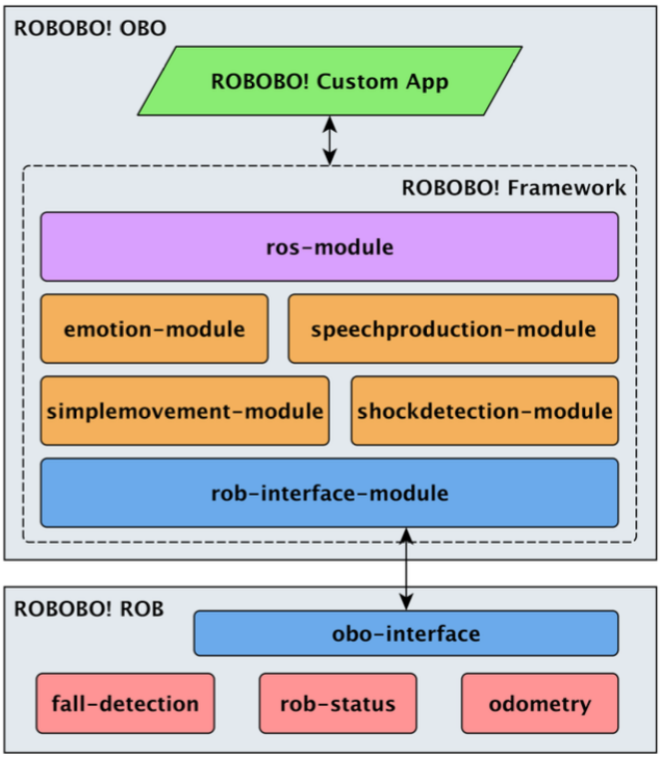

Figura 2.4: Estructura de ROBOBO! framework dentro del entorno ROBOBO

2.2.1. ROBOBO! Framework

El ROBOBO! Framework [2] es el framework de desarrollo usado para desarrollas aplicaciones para el ROBOBO. Está diseñado de manera modular, de forma que facilita la extensión del mismo con nuevos módulos. Está publicado con una licencia libre LGPL v3. Ofrece el módulo *rob-interface-module*, que permite la comunicación entre la plataforma robotizada (denominada ROB, figura 2.2) y el smartphone (denominado OBO en el sistema). También proporciona el módulo *ros-module* que sirve de interfaz entre el robot y ROS [3]

Capítulo 3

Interacción Humano Robot

En este capítulo se introduce el tema de la Interacción humano robot, y el diseño conceptual del sistema de interacción desarrollado para la plataforma ROBOBO.

3.1. Introducción

Desde finales de la década de los 90, la interacción entre humanos y robots ha cobrado importancia, creando un nuevo campo de investigación en la ciencia [4], la interacción humano-robot (HRI por sus siglas en inglés). El campo de la HRI busca entender, modelar y evaluar las diferentes modalidades de interacción entre las personas y los robots. La comunicación entre humanos y robots puede dividirse en dos categorías generales:

- Interacción remota
- Interacción próxima

La interacción remota suele ser referida como control supervisado o teleoperación, dependiendo de si el robot es autónomo con supervisión de un humano, que interviene en caso de necesidad, o si el robot es controlado por el humano directamente. Este tipo de interacción puede verse en robots de tipo industrial o en vehículos autónomos, cómo los llamados Drones del ejercito.

La interacción próxima es aquella en la que el robot interactúa directamente con el humano, llegando incluso a haber interacción física. Este tipo de interacción incluye elementos emotivos y sociales, y se puede encontrar en, por ejemplo, los

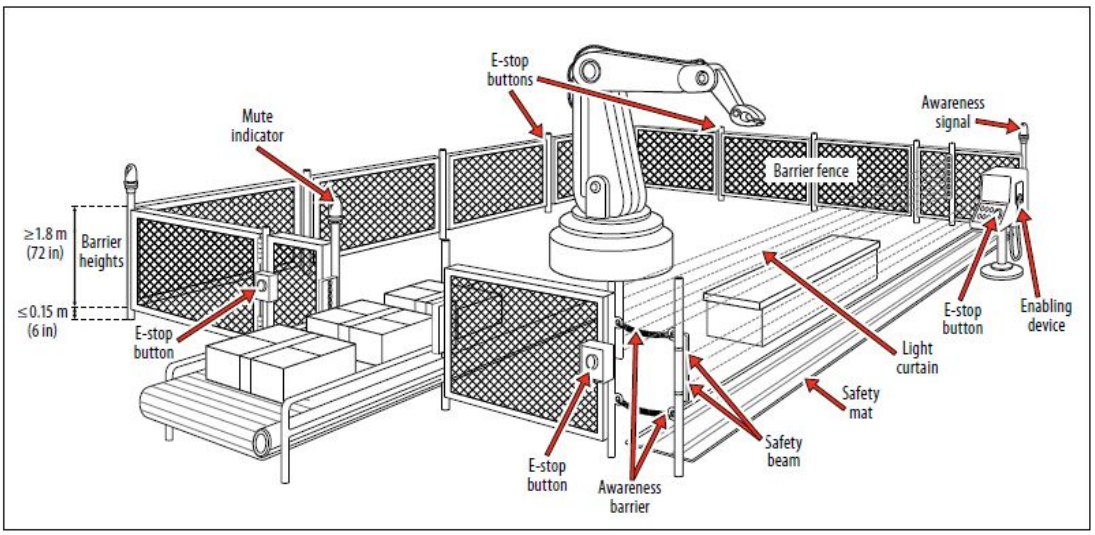

Figura 3.1: Esquema de una jaula de seguridad para un robot industrial

robots asistenciales o educativos. Este tipo de interacción es la que se tratará en este trabajo, en el cual se diseñarán diferentes sistemas de interacción para la plataforma ROBOBO.

3.1.1. HRI en la industria

Generalmente, en los procesos industriales, la interacción entre los robots y los operadores suele ser remota, los sistemas se programan para realizar una tarea y el humano solamente interviene en caso de necesidad, sin embargo pueden darse casos de interacción próxima con los robots. Uno de estos casos podría ser el aprendizaje de tareas mediante demostración, proceso mediante el cual, un operador humano realiza una tarea, por ejemplo moviendo manualmente el brazo de un robot, para que el controlador aprenda a realizar esa tarea. Este tipo de interacción permite que los robots aprendan comportamientos de alto nivel difícilmente programables.

Sin embargo a la hora de realizar tareas de forma cooperativa entre robots y humanos, la interacción cercana con robots industriales conlleva riesgos importantes, como pudo verse en el accidente del 2015 [5] en la planta de Volkswagen cerca de Kassel, Alemania, en el que un operario fué golpeado por un brazo industrial durante su instalación, resultando en la muerte del técnico. Para evitar esta clase de accidentes, se está buscando la consciencia del entorno en los manipula-

dores robóticos, para poder adaptar sus reacciones al contexto actual. Este tipo de consciencia no solo disminuye los riesgos de operación, sino que también disminuye espaciales, ya que los robots no requerirían de jaulas de seguridad (figura 3.1) , también la productividad se vería afectada positivamente, ya que tareas imposibles de realizar para un robot y para un humano individualmente, pueden llevarse a cabo mediante la llamada robótica cooperativa. En *Cooperative Tasks between Humans and Robots in Industrial Environments* [6] presentan un sistema de robótica cooperativa en el que un operador y un robot colaboran de manera cercana para llevar a cabo diferentes tareas, de manera que el robot realiza las tareas repetitivas y peligrosas, mientras que el humano lleva a cabo las tareas que requieren de cierta precisión o inteligencia con la que no cuenta el robot. En este sistema el operador lleva un traje de posicionamiento, que permite al robot conocer su posición, pudiendo así adaptar sus movimientos de manera que el humano no corra riesgos.

3.1.2. HRI en robots asistenciales

Uno de los campos en los que la interacción entre humanos y robots cobra mucha importancia es en el nicho de los robots asistenciales. Los robots asistenciales, también llamados de servicio, son definidos por la federación internacional de robótica cómo *Robots que operan de forma total o semiautónoma para realizar servicios útiles para el bienestar de humanos y equipamiento, excluyendo las operaciones de manufactura* [7]. En esta definición se diferencia entre dos tipos de robots asistenciales:

- Robots personales
- Robots profesionales

Los robots personales son aquellos que se utilizan para labores no comerciales, generalmente por personas sin perfil técnico. Por ejemplo, sillas de ruedas eléctricas, robots de asistencia de movilidad, o aspiradoras automáticas.

Los robots profesionales son aquellos utilizados para realizar tareas de asistencia en un entorno comercial, generalmente manejados y supervisados por un personal especializado. Por ejemplo robots de limpieza automatizados para zonas públicas, robots de mensajería en oficinas u hospitales, robots anti-incendios,robots quirúrgicos y de rehabilitación en hospitales o los robots terapéuticos.

Figura 3.2: Robot terapeutico Paro

En los robots terapéuticos se pueden encontrar múltiples formas de interacción, por ejemplo, el robot Paro [8] (figura 3.2) es un robot terapéutico con forma de bebé foca, utilizado con éxito en terapias contra la Demencia, que busca una interacción emocional con el paciente, para ello cuenta con cinco tipos de sensores diferentes, táctiles, auditivos, de temperatura, de luz y posturales. Los pacientes realizan una interacción con el robot cómo la que tendrían con un animal, y el robot responde acorde a los estímulos que recibe. Esto, en conjunto con la forma física del robot, más semejante a un animal de peluche que a una máquina, permite al paciente desarrollar emociones. El robot NAO [9](figura 3.3) es otro de los robots que se están empleando con éxito en tareas asistenciales, tanto de manera terapéutica, como robot de relaciones publicas o para tareas de educación. El robot cuenta con una amplia variedad de sensores y actuadores que le permiten interactuar de diversas formas con el usuario:

- Cámaras: Permiten reconocimiento de caras y procesado de imagen.
- Sensores táctiles y de presión: Permiten una interacción física con el robot.
- Altavoces: Permiten al robot producir diversos sonidos y hablar.
- Micrófonos: Permiten reconocer habla y ubicar el origen de los sonidos espacialmente.
- Sensores de distancia: Permiten detectar la distancia a los objetos.

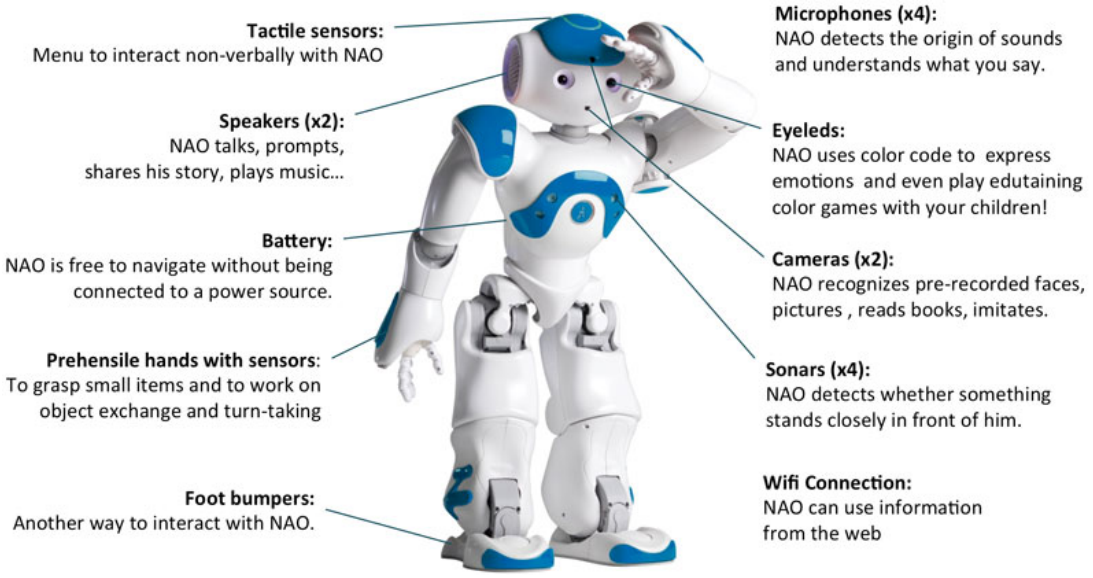

Figura 3.3: Esquema de características del robot NAO

- 25 Grados de libertad: Permiten al robot interactuar fisicamente con su entorno y realizar comunicación no verbal.
- Unidad de medición inercial: Permite detectar aceleraciones y giros.

En el caso de los robots médicos, los quirúrgicos especificamente, la interacción no suele pasar de la teleoperación del robot, es decir, el robot se controla como si fuera una extensión del cirujano. El robot más conocido en este campo es el sistema quirúrgico Da Vinci (figura 3.4), utilizado en cirugía de precisión. En este robot el cirujano controla los diferentes brazos del aparato desde una consola que cuenta con controles con feedback háptico, es decir, el operador no mueve unicamente el brazo, sino que siente lo que hay al final del mismo, la presión ejercida y la resistencia al movimiento, dando al cirujano el tacto necesario para realizar las diferentes tareas que se realizan en una operación, como coser o cortar, de forma natural y con un alto grado de precisión.

3.1.3. HRI en robots de entretenimiento

Se entiende por robots de entretenimiento aquellos cuya finalidad no es más que divertir al usuario. Dentro de esta categoría podríamos incluir a los robots mascota, que generalmente realizan una interacción de alto nivel con el usua-

Figura 3.4: Sistema quirúrgico daVinci

rio. Por ejemplo, uno de los robots más relevantes en el ámbito de los robots mascota es el desarrollado por Sony, Aibo [10] (figura 3.5) , cuya finalidad era comportarse como un perro. En este robot podemos encontrar múltiples tipos de interacción, interacción física en forma de movimientos perrunos, reconocimiento facial a través de cámaras, a través de caricias utilizando los sensores táctiles, y en las últimas versiones del robot mediante una matriz de leds situada en la cara del aparato, que permite poner diferentes expresiones en función del "humor"del robot. Mediante todas estas capacidades motoras y sensoriales, se puede interactuar con una unidad Aibo de manera semejante a la que se tendría con un perro real. Otro ejemplo de mascota robótica es el Bandai SmartPet (figura 3.6) , un robot pensado para utilizar junto un smartphone IPhone, que es colocado en la cabeza del robot y provee múltiples formas de interacción, reconocimiento de gestos mediante la cámara frontal, reconocimiento de sonido utilizando el micrófono y reconocimiento de gestos táctiles en la pantalla.

Figura 3.5: Robot mascota Aibo de Sony

Figura 3.6: Robot mascota Smartpet de Bandai

Figura 3.7: Beebot

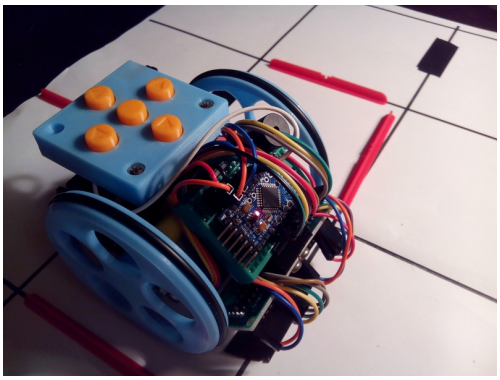

Figura 3.8: Escornabot

3.1.4. HRI en robots educativos

En los últimos años el uso de robots con fines educativos, si bien ya eran usados de manera educativa en enseñanza superior, ha tomado impulso en la educación primaria y secundaria, apareciendo múltiples robots enfocados a este tipo de mercado. La interacción con esta clase de robots puede darse de diferentes formas según el público objetivo, dependiendo principalmente del rango de edades del mismo.

Enfocados a la educación infantil nos podemos encontrar con robots cómo el BeeBot(figura 3.7) o su alternativa libre, el Escornabot(figura 3.8),la finalidad de estos robots es la introducción a los niños a la programación, en forma de pensamiento secuencial, y a la resolución de problemas. En esta clase de robots la interacción está limitada a la introducción de comandos en la botonera de la parte superior del robot, que serán traducidos a movimientos del robot posteriormente.

En la educación primaria y secundaria los robots utilizados ya adquieren una mayor complejidad y suelen emplearse para una introducción real a la programación, generalmente utilizando lenguajes gráficos de muy alto nivel como Scratch [11]. Uno de los robots más relevantes en este ámbito es el kit Mindstorms de Lego (figura 3.9) , que no solo permite programar el robot, sino también construirlo. El Lego Mindstorms cuenta con diferentes sensores y actuadores que ofrecen diversas maneras de interacción con el robot:

- Motores: Permiten el movimiento del robot de manera relativamente precisa
- Sensores de distancia: Permiten medir distancias y actuar en consecuencia a los datos medidos

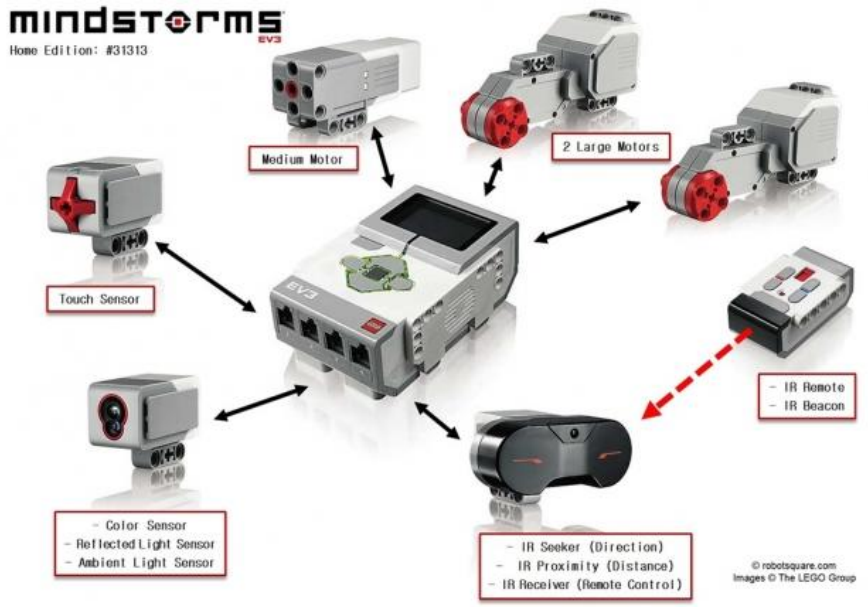

Figura 3.9: Kit básico Lego Mindstorms EV3

- Sensores de luz ambiente: Miden la intensidad de la luz del entorno
- Sensores de color: Permiten detectar los colores básicos
- Unidad inercial: Permite medir giros en el plano horizontal

En la llamada educación especial la robótica también esta siendo empleada con éxito, por ejemplo el robot NAO (figura 3.3), del que se habló en la sección anterior, se utiliza para educar a niños con trastornos de espectro autista. El éxito de este tipo de educación viene dada debido a que la interacción con el robot, al ser programada, resulta predecible para el alumno, creando un entorno estable y pautado que resulta óptimo en este tipo de trastornos. La interacción en este caso se da en forma de movimientos preprogramados y mediante los leds de los ojos del robot, que cambian de color segun las emociones que intenta expresar el robot, lo cual ayuda al alumno a ejercitar uno de las mayores dificultades dadas por el autismo, la dificultad de establecer relaciones empáticas.

Ejemplos de diferentes robots y modos de interacción... Industriales, guías de museo, asistenciales, educativos

3.2. Interacción con robots educativos

Ejemplos específicos de robots educativos

3.3. Solución para el ROBOBO

La plataforma ROBOBO, cuenta con características que permiten realizar diversas maneras de interacción, el hecho de estar basado en un smartphone, pone todas las capacidades del mismo a disposición a la hora de interactuar con los usuarios. Partiendo del hardware con el que se cuenta, base motorizada *ROB* y smartphone *OBO* , se han definido, en una analogía a los sentidos, una serie de paquetes para dotar al ROBOBO de diversas funcionalidades de interacción.

El primero de estos paquetes sería el paquete de habla. Siendo el habla, probablemente, la principal forma de interacción que se da entre los humanos, parece lógico que, teniendo la capacidad de proceso de un smartphone moderno, dotar al robot ya no solo de la capacidad de producir habla, sino de entender texto hablado es un concepto interesante. Este paquete daría al robot la posibilidad de comunicación bidireccional con los humanos, además de darle al robot cierta personalidad más «orgánica» , haciéndolo más atractivo, por ejemplo, a los niños.

El segundo, de forma análoga al sentido del tacto, teniendo una «piel artificial» cómo es la pantalla táctil del teléfono, dar la capacidad al robot de sentir toques y caricias parece una idea interesante a la hora de llevar una interacción cercana con el robot.

El tercero, siguiendo la analogía con los sentidos, sería el sentido del oído. La capacidad de producir y reconocer sonidos abre puertas a tipos de interacción interesantes. Por ejemplo, la capacidad de reaccionar ante sonidos fuertes como palmadas o la capacidad de reconocer notas musicales podrían permitir al robot comunicarse mediante el uso de la música, y estando el ROBOBO enfocado directamente a la educación, esta clase de interacción podría emplearse para la enseñanza musical en la educación infantil y primaria.

El cuarto paquete, teniendo hoy en día todos los smartphones como mínimo una cámara, busca dotar al ROBOBO de sentido de la vista. Con este sentido, el robot podría detectar la presencia de gente y a que distancia se encuentran y de esta manera adaptar su comportamiento a su entorno, por ejemplo, alejándose

si nota que alguien esta muy cerca o siguiendo con la «cara» a la gente a su alrededor. La capacidad de discernir diferentes colores también es una habilidad interesante de cara a la educación de niños muy pequeños.

Por último, teniendo la capacidad de conectarse a internet del smartphone, parece interesante dotar al robot de la capacidad de comunicarse a través de la red. El correo electrónico es un sistema de comunicación muy extendido y casi obligatorio hoy en día, así que, dada la extensión del sistema, y la cantidad de usuarios con los que cuenta, parece la opción correcta para este tipo de comunicación.

Capítulo 4

Desarrollo

En este capitulo se desarrollarán los detalles de la implementación de los diferentes módulos de interacción que se han diseñado para el ROBOBO así como la metodología de trabajo seguida.

4.1. Arquitectura Global

El sistema de interacción humana del sistema ROBOBO fue desarrollado de una forma modular, de manera que las diferentes funcionalidades puedan ser utilizadas de manera separada.

La gestión de los módulos la realiza el Framework Robobo, que proporciona una interfaz común que debe ser implementada

Cada «sentido» de los elaborados conceptualmente en la sección 3.3 se ha implementado en forma de módulo Android, y cada funcionalidad especifica ha sido integrada en módulos que implementan IModule, interfaz común para los módulos del ROBOBO , para poder ser manejados por el ROBOBO! Framework [2].

- Paquete Speech: Gestiona las operaciones que involucran habla.
  - Módulo SpeechProduction: Permite utilizar la síntesis de voz.
  - Módulo SpeechRecognition: Permite el reconocimiento de voz, por palabras clave o por gramáticas.
- Paquete Touch: Gestiona el sentido del tacto del robot.

    - Módulo Touch: Permite la detec ción de gestos en la pantalla táctil del móvil.

- Paquete Sound: Gestiona el sentido del oído del robot.
    - Módulo SoundDispatcher: Módulo auxiliar para los módulos que emplean la librería TarsosDSP [12]
    - Módulo ClapDetectionModule: Módulo que permite la detección de sonidos percusivos.
    - Módulo PitchDetection: Módulo que permite la estimación de la frecuencia de un sonido.
    - Módulo NoteDetection: Módulo que detecta notas musicales a partir de frecuencias.
- Paquete Vision: Gestiona el sentido de la vista del robot.
    - Módulo BasicCamera: Provee de un stream constante de frames capturados desde la cámara frontal del smartphone.
    - Módulo FaceDetection: Permite detectar caras en los frames capturados por la cámara.
    - Módulo ColorDetection: Permite detectar colores sobre fondos blancos en los frames.
- Paquete Messaging: Gestiona los métodos de mensajería del robot
    - Módulo Messaging: Permite enviar correos electrónicos, pudiendo adjuntar imágenes.

Cada paquete se distribuye en forma de librería Android.

4.1.1. Estructura de un módulo

Todos los módulos comparten una estructura de paquetes similar, un paquete con el nombre del módulo , que contiene la interfaz del módulo (de la forma *INombreModulo*) que extiende la clase *IModule*, una clase abstracta (*ANombreModulo*) que implementa la interfaz anterior y gestiona labores comunes como la suscripción a los Listeners, y los paquetes de las diferentes implementaciones

especificas. Opcionalmente, este paquete también podrá contener la interfaz de los listeners, y clases de utilidad como enumerados.

Las implementaciones de los módulos tienen el nombre *NombreImplementacionNombreModulo*, y deben extender la clase abstracta citada anteriormente.

4.2. Metodología

La metodología seguida fue un proceso iterativo en el cual en cada iteración se fueron añadiendo nuevos módulos de interacción y realizando correcciones en los módulos ya implementados.

4.2.1. Primera iteración

En esta primera iteración el trabajo de desarrollo se centró en los paquetes de producción de habla (Módulo *Speech*) y el de tacto (Módulo *Touch*. Se implementaron las funcionalidades básicas de cada módulo:

- *SpeechProductionModule*: Se implementaron las funcionalidades de síntesis de voz, selección de voces y de localización.
- *SpeechRecognitionModule*: Se implementó el reconocimiento básico por palabras clave.
- *TouchModule*: Se implementaron el reconocimiento de toques largos(touch) y cortos(tap), así como el reconocimiento de caricias(caress) y deslizamientos rápidos(fling).

4.2.2. Segunda iteración

En la segunda iteración se desarrolló el paquete de interacción por sonidos y se realizaron correcciones tanto en el módulo de reconocimiento de voz cómo en el táctil.

Nuevas implementaciones en esta iteración:

- *SoundDispatcherModule*: Implementado para dar soporte al resto de módulos que usan la librería TarsosDSP [12].

- *ClapDetectionModule*: Implementado el módulo de detección de sonidos percusivos.
- *PitchDetectionModule*: Implementada el módulo de detección de frecuencias en hertzios.
- *NoteDetectionModule*: Implementada el módulo de detección de notas musicales. Implementada clase para representar las notas.

Actualizaciones de módulos de la anterior iteración:

- *SpeechRecognitionModule*: Implementada la búsqueda por gramáticas
- *TouchModule*: Mejorada la información producida por el evento *onFling*.

4.2.3. Tercera iteración

En esta iteración se desarrollaron los paquetes de visión y de mensajería. También se sacaron de dentro de los módulos los diferentes parámetros, para poder ser configurados desde el exterior mediante ficheros *properties*.

Nuevas implementaciones en esta iteración:

- *BasicCameraModule*: Implementado el notificador de frames.
- *FaceDetectionModule*: Implementado el modulo de detección de caras.
- *ColorDetectionModule*: Implementado el modulo de detección de colores.
- *MessagingModule*: Implementado el módulo de mensajería por gMail.

Actualizaciones de módulos:

- *SoundDispatcherModule*: Parametrización mediante properties.
- *ClapDetectionModule*: Parametrización mediante properties.
- *PitchDetectionModule*: Parametrización mediante properties.

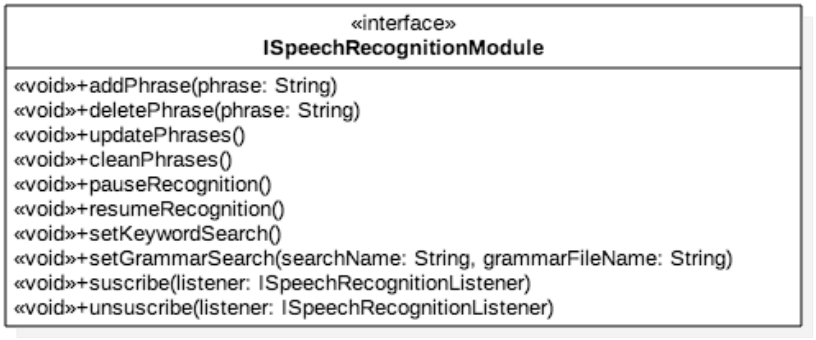

Figura 4.1: Módulo SpeechRecognition

4.3. Librerías de interacción

A continuación se hablará sobre los detalles de implementación de las diferentes librerías de interacción desarrolladas.

4.3.1. Paquete Speech

Este subsistema gestiona todas las labores que tienen que ver con la generación o detención del habla. En su interior encapsula dos paquetes diferentes, Production y Recognition, que contienen respectivamente los módulos de producción y reconocimiento del habla.

4.3.1.1. Módulo recognition

El módulo recognition permite al usuario reconocer texto hablado de forma fácil. Provee las Interfaces *ISpeechRecognitionModule* y *ISpeechRecognitionListener*, así cómo la clase abstracta *ASpeechRecognitionModule*.

ISpeechRecognitionModule ofrece la declaración de métodos de configuración del reconocedor de voz y de suscripción de listeners.

Especificamente permite:

- Añadir y eliminar palabras reconocibles.
- Actualizar el listado interno de frases
- Limpiar el listado interno

- Pausar y reanudar el reconocimiento
- Añadir búsquedas por gramática
- Utilizar búsqueda por palabras

ISpeechRecognitionListener es la interfaz que deben implementar las clases para ser notificados de los eventos que produce el módulo de reconocimiento.

Especificamente notifica cuando:

- El reconocedor ha sido inicializado
- Una frase ha sido detectada

ASpeechRecognitionModule es la clase abstracta que gestiona las tareas comunes a todas las implementaciones del modulo, a saber:

- Mantener un listado de los listeners
- Suscribir y desuscribir listeners
- Notificar a los listeners de los diferentes eventos

Pocketsphinx

Se ha decidido realizar la implementación del módulo utilizando la librería CMU Sphinx [13], desarrollada por la Universidad de Carnegie Mellon. Específicamente se utiliza PocketSphinx, una versión de CMU Sphinx ligera, diseñada para su uso en sistemas embebidos.

PocketSphinx permite varios tipos de búsqueda diferentes, pero al contrario que las versiones más completas de la librería, sólo permite tener una búsqueda activa al mismo tiempo. Entre los modos de búsqueda que ofrece se encuentran la búsqueda por Keywords y la búsqueda por Gramáticas, que son las que son ofrecidas por la interfaz del módulo. Adicionalmente PS también permite el uso de búsquedas por fonemas, pero no se ha contemplado su uso.

El módulo se llama *PocketSphinxSpeechRecognitionModule*, extiende *ASpeechRecognitionModule* e implementa la interfaz *RecognitionListener* que provee PocketSphinx.

La búsqueda por keywords permite al usuario definir, en tiempo de ejecución, una serie de palabras que podrán ser reconocidas, estas palabras deben estar contenidas en el diccionario del modelo fonético.

La búsqueda por gramática permite definir archivos de gramáticas en el formato JSFG 1.0 [14], lo cual permite detectar construcciones más complejas de palabras. Para funcionar correctamente deben colocarse en el directorio assets/sync/

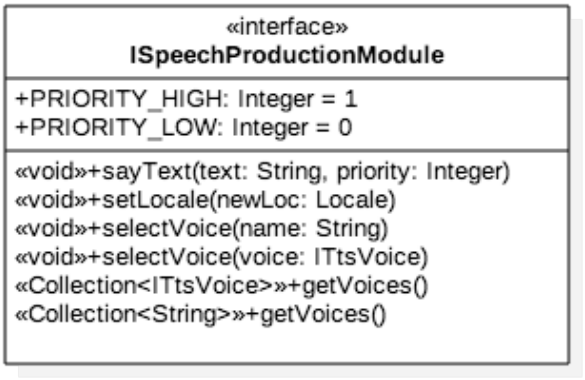

Figura 4.2: Modulo SpeechProduction

de la aplicación tanto el modelo fonético del idioma, como los archivos de gramáticas que se vayan a utilizar, todo archivo dentro de este directorio debe contar con un fichero de mismo nombre y extensión .md5 conteniendo un hash del original en su interior. Los archivos deben ser declarados dentro del fichero assets.lst.

Dado que el inicio del reconocedor de voz se realiza de manera asíncrona, es necesario notificar a la aplicación principal de cuando el módulo esta preparado, de lo contrario, llamar a una operación del mismo podría ser inseguro. Le arranque del módulo se notifica a través del método onModuleStart() que debe ser implementado por cualquier clase que vaya a utilizar el módulo.

4.3.1.2. Módulo Production

El módulo production permite al usuario producir voz a partir de texto de manera simple. Provee la interfaces *ISpeechProductionModule* y *ITtsVoice*, así como la excepción *VoiceNotFoundException*. La primera contiene las declaraciones de los métodos del módulo en sí, la segunda provee una forma de representar las diferentes voces a utilizar por el módulo.

Las funcionalidades que provee la interfaz son:

- Pronunciar un texto
- Cambiar la localización
- Seleccionar voces

- Recuperar las voces disponibles

Implementación específica:

La implementación de la interfaz se ha realizado con las librerías del propio sistema Android (**android.speech.tts.TextToSpeech**)

Figura 4.3: Módulo Touch

4.3.2. Paquete Touch

El subsistema Touch permite interacción con el robot en forma de gestos táctiles sobre la pantalla del teléfono. En su interior encontramos un único modulo, el módulo touch.

4.3.2.1. Módulo Touch

El módulo contiene las interfaces *ITouchModule* e *ITouchListener*, la clase abstracta *ATouchModule* y por la clase enumerado *TouchGestureDirection*.

ITouchListener es la interfaz que debe implementar cualquier clase que quiera ser informada de los gestos táctiles que ocurran en la pantalla del dispositivo. Las acciones que se detectan son:

- Tap: Toque simple y rápido en la pantalla
- Touch: Toque mantenido en la pantalla
- Caress: Deslizamiento sobre la pantalla
- Fling: Al terminar un deslizamiento de manera rápida

ITouchModule es la interfaz que debe cualquier implementación del módulo. El único método remarcable de la misma es *FeedTouchEvent*, en el cual se le pasan los eventos táctiles al módulo desde la aplicación principal, que debe sobrecargar el método *OnTouchEvent* de la clase *Activity*, esto es debido a que el módulo no puede extender la clase *Activity*.

Detección táctil de Android

Para la implementación del módulo se escogió usar la clase *GestureDetector* propia de Android, que provee de varios listeners para diferentes eventos táctiles. Se han simplificado dichos eventos, para producir los definidos en *ITouchListener*

4.3.3. Paquete Sound

Este subsistema contiene los módulos que realizan diversas tareas de reconocimiento de audio. Han sido implementadas detección de tonos, notas musicales y palmadas.

4.3.3.1. Módulo Sound Dispatcher

SoundDispatcher es un módulo de utilidad, necesario para el uso de los módulos implementados a partir de la librería TarsosDSP [12]. Este módulo es el que se encarga de capturar el sonido del micrófono y segmentarlo en partes para su posterior procesado. Los distintos módulos se registran con sus procesadores en este módulo. El paquete contiene en su interior *ISoundDispatcherModule*, la interfaz del módulo, que permite

- Añadir procesadores de audio
- Eliminar procesadores de audio
- Iniciar el procesado de audio
- Parar el procesado de audio

ASoundDispatcherModule implementa *ISoundDispatcherModule*, pese a estar vacía se conserva para mantener la estructura común de los módulos.

TarsosDSP
La implementación se realiza en el módulo *TarsosDSPSoundDispatcherModule*, que extiende la clase *ASoundDispatcherModule*. En su interior se crea un objeto *AudioDispatcher*, que se encarga del procesado del audio, y al que pueden añadirse distintos procesadores de audio.

4.3.3.2. Módulo Pitch Detection

PitchDetection permite detectar la frecuencia en Hertzios de sonidos, es utilizado, por ejemplo en el módulo de detección de notas musicales. En su interior encontramos la interfaz *IPitchDetectionModule* y la clase abstracta *APitchDetectionModule*, que gestionan la suscripción de listeners a los eventos. También

se encuentra *IPitchListener*, que debe ser implementada por cualquier clase que desee recibir notificaciones del módulo de detección de tonos.

TarsosDSP

La implementación del módulo se ha realizado en la clase *TarsosDSPPitchDetectionModule*. Esta implementación depende del módulo SoundDispatcher, ya que en su inicialización se registra a si mismo en el susodicho modulo. Esta implementación utiliza el algoritmo YIN [15] para la detección de frecuencias. La implementación del algoritmo viene dada por la propia librería TarsosDSP. El módulo genera un evento con la frecuencia cuando un sonido es detectado, y al terminar la detección devuelve un -1.

4.3.3.3. Módulo Note Detection

El módulo NoteDetection busca proporcionar una manera simple de detectar notas musicales. Depende directamente del módulo PitchDetection, del cual obtiene la frecuencia que traduce en notas musicales. En su interior encontramos la interfaz del módulo *INoteDetectionModule*, la clase abstracta *ANoteDetectionModule*, que además de la gestión de listeners provee un método de conversión de frecuencias a notas, la interfaz *INoteListener* que recibe notificaciones al:

- Comenzar una nota
- Detectar una nota
- Terminar una nota

Toda clase que desee recibir los eventos producidos por el módulo debe implementar esta interfaz. Por último en el paquete encontramos una clase *Note* que se utiliza como representación de las notas musicales. Este enumerado contiene la información del índice de la conversión de notas, la nota musical en formato anglosajón, y la octava a la que pertenece la nota.

TarsosDSP

TarsosDSPNoteDetectionModule extiende *ANoteDetectionModule* e implementa *IPitchListener*. Simplemente convierte los tonos detectados a notas musicales, filtra las notas desafinadas y controla los inicios y finalizaciones de las notas, generando eventos de los cuales serán notificados los listeners.

4.3.3.4. Módulo Clap Detection

El módulo ClapDetection permite detectar ruidos percusivos tales como palmadas. Igual que el PitchDetection, depende directamente del módulo SoundDispatcher para su funcionamiento. En su interior se encuentra la interfaz del módulo, *IClapDetectionModule*, la clase abstracta *AClapDetectionModule* y la interfaz del listener, *IClapListener*. Las clases que implementen este listener recibirán una notificación cada vez que se detecte una palmada.

TarsosDSP

La implementación del módulo, contenida en *TarsosDSPClapDetectionModule*, emplea el objeto provisto por la librería TarsosDSP *PercussionOnsetDetector*, que es registrado en el *DispatcherModule* activo en ese momento. Este objeto permite la detección de sonidos percusivos, como palmadas de una forma sencilla.

4.3.3.5. Módulo NoteGenerator

El módulo NoteGenerator provee una forma de sintetizar tonos musicales y hacer sonar secuencias de notas. En su interior se encuentra la interfaz del módulo *INoteGeneratorModule* que define las funcionalidades del módulo, la clase abstracta *ANoteGeneratorModule* que gestiona la suscripción a los listeners. La interfaz del listener se encuentra en *INotePlayListener* y su finalidad es notificar cuando una nota o una secuencia de notas termina de reproducirse. La clase *Note* es la representación de las notas musicales en el sistema, contiene la nota, el índice y la octava.

Android

La implementación especifica del módulo se encuentra en AndroidNoteGeneratorModule y se ha realizado mediante la clase *AudioTrack*. El objeto *AudioTrack* se crea Sample a Sample utilizando la ecuación de una onda senoidal de la frecuencia de la nota requerida. La reproducción de secuencias se implementa mediante el uso de las clases *Timer* y *TimerTask*

4.3.3.6. Módulo EmotionSound

El módulo EmotionSound busca proporcionar una manera de reproducir una serie de sonidos prefijados para expresar diferentes emociones. La interfaz del módulo esta definida en *IEmotionSoundModule*, y en ella están definidos, en forma de constantes, los sonidos utilizables. Los sonidos se encuentran en el directorio raw de la carpeta res del paquete Sound.

Android

La implementación de este módulo se ha realizado mediante la clase *MediaPlayer* provista por la API de Android, que permite reproducir diferentes recursos audiovisuales. El módulo está definido en la clase *AndroidEmotionSoundModule*.

4.3.4. Paquete Vision

Este subsistema contiene los diferentes módulos de captura y procesado de imagen

4.3.4.1. Módulo Basic Camera

Este es el módulo básico de cámara, del cual hacen uso el resto de módulos de procesado de imagen. Este modulo produce un stream constante de imágenes, que no deben ser necesariamente mostradas, capturadas de la cámara frontal del dispositivo y notifica a los listeners suscritos. La interfaz del módulo se encuentra en *ICameraModule*, la clase abstracta que gestiona los listeners en *ACameraModule* y la interfaz del listener en el que se notifica la captura de las imágenes en *ICameraListener*. Además, el modulo proporciona una clase Frame, que representa a las imágenes capturadas con sus características.

Android Camera2
Para realizar la implementación del módulo se ha empleado la clase *Camera2* que proporciona Android. Esta implementación permite obtener el stream mencionado anteriormente sin mostrar las imágenes en la pantalla, pero produce una velocidad de captura de imágenes baja, de alrededor de dos cuadros por segundo.

4.3.4.2. Módulo Face Detection

El módulo FaceDetection permite detectar caras en los frames producidos por el módulo BasicCamera. La interfaz del módulo es *IFaceDetectionModule* y su clase abstracta *AFaceDetectionModule*. El listener que debe ser implementado por la clase que utilice el módulo es *IFaceListener*, que notifica de las coordenadas del centro de la cara y de la distancia entre ojos cuando una cara es detectada. Solo se contempla la detección de una cara al mismo tiempo.

Android FaceDetector
Para implementar este módulo se ha empleado la clase *FaceDetector* provista por el paquete Media de Android, el módulo se llama *AndroidFaceDetectionModule*.

4.3.4.3. Módulo ColorDetector

Este módulo provee la funcionalidad de detección de colores sobre fondos con alto contraste. La interfaz del módulo se puede encontrar en *IColorDetectionModule*, la clase abstracta *AColorDetectionModule* gestiona los listeners en los que será notificada la detección de los colores. La interfaz de dicho listener se encuentra en la clase *IColorListener* y debe ser implementada por toda clase que desee recibir las notificaciones de los colores.

4.3.4.3.1. OpenCV Para realizar la implementación de este módulo se ha empleado la librería OpenCV [**?**] una de las más usadas en visión por computador. Este módulo, contenido en *OpenCVColorDetectionModule* realiza los siguientes pasos para la detección de colores:

- Detección de bordes mediante el algoritmo de Canny, obteniendo el contorno con mayor area.
- Creación de una máscara binaria con el area detectada
- Conversión de la imagen original al espacio de color HSV
- Media de color en los pixeles no nulos en el canal H de la imagen resultante de un AND entre la máscara y ma imagen convertida
- Clasificación del color detectado

4.3.5. Paquete Messaging

El subsistema Messaging aglomera las diferentes opciones de comunicación por mensajería de texto.

4.3.5.1. Módulo email

El módulo Email permite al usuario la comunicación mediante correo electrónico, pudiendo mandar tanto texto como imágenes, por ejemplo, las capturadas con el módulo *BasicCamera*.

GmailBackground

Para implementar el módulo, se ha empleado la librería GmailBackGround [16], que permite el envío de mensajes de correo electrónico con una cuenta de Gmail. El módulo se encuentra en la clase *GmailBackgroundMessagingModule*.

Capítulo 5
Evaluación

5.1. Comparativa

5.2. Problemas conocidos

Capítulo 6

Conclusiones

6.1. Trabajo Futuro

Apéndice A

Diagramas

Apéndice B

Instalación

Apéndice C

Manual de uso

Bibliografía

[1] "D.r.e.a.m. project," http://www.robotsthatdream.eu/, accessed: 2016-15-6.

[2] "Robobo framework," https://bitbucket.org/mytechia/robobo-framework, accessed: 2016-07-13.

[3] "Robot operating system (ros)," http://www.ros.org/, accessed: 2016-07-14.

[4] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Foundations and trends in human-computer interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[5] "Worker at volkswagen plant killed in robot accident," http://www.ft.com/cms/s/0/0c8034a6-200f-11e5-aa5a-398b2169cf79.html, accessed: 2016-07-19.

[6] J. A. Corrales Ramón, G. J. García Gómez, F. Torres Medina, and V. Perdereau, "Cooperative tasks between humans and robots in industrial environments," 2012.

[7] "Definition of service robots," http://www.ifr.org/service-robots/ , accessed: 2016-07-19.

[8] "Paro robots," http://www.parorobots.com/index.asp, accessed: 2016-07-19.

[9] "Nao robot, aldebaran robotics," https://www.ald.softbankrobotics.com/en/cool-robots/nao/find-out-more-about-nao, accessed: 2016-07-19.

[10] "Sony aibo — the history of the robotic dog," http://www.sony-aibo.com/, accessed: 2016-07-21.

[11] "Scratch language project main page," https://scratch.mit.edu/, accessed: 2016-07-20.

[12] J. Six, O. Cornelis, and M. Leman, "TarsosDSP, a Real-Time Audio Processing Framework in Java," in *Proceedings of the 53rd AES Conference (AES 53rd)*, 2014.

[13] "Cmu sphinx open source speech recognition kit," http://cmusphinx.sourceforge.net/, accessed: 2016-07-6.

[14] "Jspeech grammar format," https://www.w3.org/TR/jsgf/, accessed: 2016-07-6.

[15] A. de Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *J Acoust Soc Am*, vol. 111, pp. 1917–1930, 2002. [Online]. Available: http://audition.ens.fr/adc/pdf/2002_JASA_YIN.pdf

[16] "Gmail background library," https://github.com/yesidlazaro/GmailBackground, accessed: 2016-08-4.

[17] "Open source computer vision library," https://github.com/opencv, 2015.