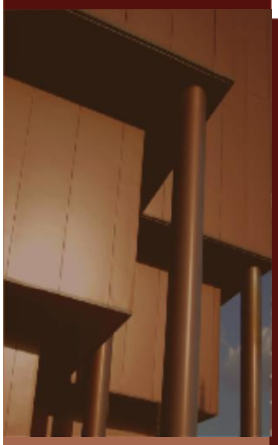
Proyecto Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

Integración de robot social en sistema domótico

Autor: Ana Rocío Racero Valcárcel

Tutor: Jose María Maestre Torreblanca



Dep. Ing. De sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Proyecto Fin de Grado
Grado en Ingeniería en Tecnologías Industriales

Integración de robot social en sistema domótico

Autor:
Ana Rocío Racero Valcárcel

Tutor:
José María Maestre Torreblanca
Profesor titular

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Proyecto Fin de Grado: Integración de robot social en sistema domótico

Autor: Ana Rocío Racero Valcárcel

Tutor: Jose María Maestre Torreblanca

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis amigas,

A mi tutor,

Al equipo Aisoy Robotics

Agradecimientos

Quisiera dedicar estas líneas a todas aquellas personas que me han ayudado a llegar hasta aquí.

En mi primer lugar a mis padres, Ana y Jose, por vuestro apoyo y ayuda incondicional que me habéis proporcionado y mi a hermano, Grego, por los buenos consejos que siempre me has dado.

A mis amigas por estar ahí cuando lo he necesitado, y los buenos momentos que me han hecho pasar. Y María por todos tantos días juntas en la esi, que gracias a eso y muchos más somos lo que somos.

A tí, Iñigo por tu cariño y soportarme los malos momentos.

Al equipo Aisoy Robotics por las facilidades proporcionadas frente a las adversidades que han ido surgiendo con el robot.

A mi tutor del presente proyecto, Pepe Maestre, por su paciencia, tiempo y dedicación que ha tenido, además gracias por el trato cercano que me has dado ya que me ha hecho afrontar el trabajo con el mejor ánimo.

Ana Rocío Racero Valcárcel

Sevilla, 2016

Resumen

El objetivo de este proyecto es dar a conocer un nuevo robot, Aisoy, y principalmente y lo más atractivo, su integración en domótica. No obstante para llegar ahí se ha hecho un previo análisis del robot con diversas demos y sus capacidades, así como un aprendizaje básico de su lenguaje de programación.

Para ello este proyecto se ha divido en tres capítulos.

El primer capítulo es una introducción al robot, la explicación de su hardware, su software y la programación en un lenguaje accesible para todos los públicos, destinado principalmente a niños, Scratch.

El segundo capítulo trata del lenguaje de programación utilizado para llevar a cabo el proyecto, Python. Tiene varios subcapítulos para un aprendizaje desde cero de dicho lenguaje, además de una introducción en algunas librerías que han sido las utilizadas para la programación.

El último capítulo ocupa la integración en domótica, con una breve explicación de la domótica, y para su aplicación se ha usado la plataforma If This Then That, la cual también se explica detalladamente y con diversos ejemplos. Para la aplicación de la domótica se ha adquirido un kit de bombillas LED Belkin inteligentes, capaces de ser conectadas a la plataforma IFTTT.

Abstract

The objective of this project is to introduce a new robot, Aisoy, and especially and most attractive, their integration in home automation. However to get there it has made a preliminary analysis of the robot with various demos and capabilities, as well as learning a basic programming language.

To do this project has been divided into three chapters.

The first chapter is an introduction to the robot, the explanation of its hardware, software and programming language accessible to all audiences, mainly aimed at children, Scratch.

The second chapter deals with the programming language used to carry out the project, Python. It has several subchapters for learning that language, plus an introduction in some libraries that are used for programming.

The last chapter deals with the integration in home automation, with a brief explanation of home automation, and application platform has been used If This Then That, which is also explained in detail and with several examples. For the application of home automation it has been acquired a LED intelligent kit bulb, capable of being connected to the platform IFTTT Belkin.

Índice

ÍNDICE DE FIGURAS

1 AISOY

1.1 Introducción

Hace 5 años que se empezó a desarrollar el proyecto del robot Aisoy. Fueron un conjunto de ingenieros matemáticos y consultores los que decidieron dar el paso. Esta empresa tiene sede en España y todo su capital humano y material procede de distintos lugares de España.

Todo ello se concentró en una empresa denominada Aisoy Robotics S.L fundada en diciembre de 2008 presentando dicho robot como una nueva especie de robots diseñados para cambiar y revolucionar los hábitos de ocio de las personas.

Cuando comenzaron no existía la gran plataforma actual de Raspberry por lo que comenzaron con Beagleboard. Beagleboard es una placa de bajo consumo fabricada por Texas Instrument que satisfacía las necesidades previstas para el robot social. Más tarde con la aparición de Raspberry Pi y el gran auge que tuvo decidieron dar el paso y cambiar a dicha plataforma modificando todo el hardware del robot para que fuera como un conjunto de accesorios que se conecta a cualquier dispositivo ya fuere Raspberry pi o un ordenador con Ubuntu.

Los robots están fabricados a imagen y semejanza nuestra y adquieren su propia personalidad a base de experiencias y de las interacciones que tienen con las personas que los cuidan. Su grado de sensibilidad es muy parecido al de los humanos ya que posee un corazón artificial que le permite sentir hasta 14 estados de ánimos diferentes. Esta capacidad facilita la creación de un vínculo emocional entre los usuarios y el robot.

Otra característica importante es su capacidad de reacción ante estímulos externos gracias a los sensores de tacto ubicados en su cuerpo y botones de control para conocer información interna.

Aunque posee más de 2000 palabras en su vocabulario es posible enseñarle más palabras. Comparte aficiones con los seres humanos tales como la música ya que puede reproducir un álbum o canción.

Todo el mundo puede programar el robot con Scratch, que se explicará más adelante. Pero si se quiere avanzar más, se puede usar el SDK en Python o C++. Este proyecto se ha centrado en Python.

1.2 Apariencia física

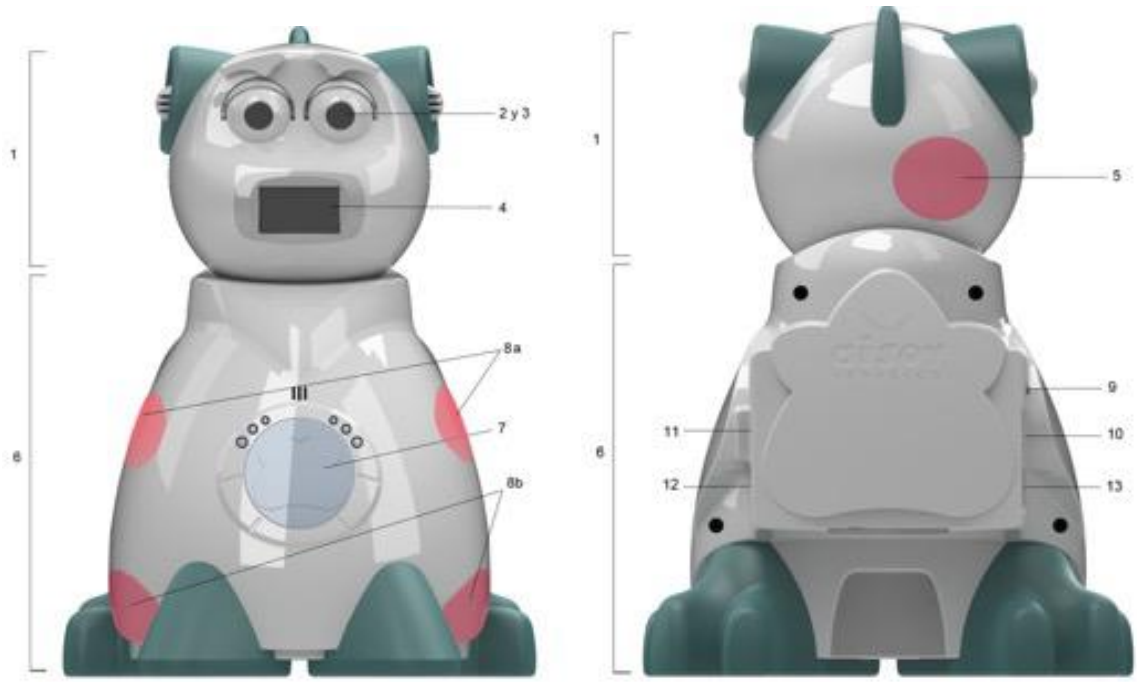


Figura 1 : Apariencia física

1 – Cabeza

2 – Cámara

3 – Micrófono

4 – Boca

5 – Sensor de tacto trasero

6 – Cuerpo

7 – Círculo de emociones

8 – Sensores de tacto laterales: 8a: belly right y back left y 8b belly left y back left

9 – Botón On/Off

10 – Tarjeta SD

11 – Tarjeta WiFi

12 – Entrada Ethernet + entrada USB

13 – Conector de corriente

1.3 Hardware

En el interior de Aisoy se incluye un microcontrolador, una CPU con 1 GB de RAM, para gestionar todos los sensores y actuadores, pero para ello necesita un ordenador al que se le denomina cerebro, una Raspberry PI 2 Model B suficiente potente como para cubrir cualquier necesidad. [1]

La Raspberry Pi es un ordenador de tamaño de una tarjeta de crédito que es posible conectar a una pantalla y un teclado. Se trata de un ordenador de placa simple (SBC, Single Board Computer) de bajo coste creada

en Reino Unido.

Fundación Raspberry PI fue creada con la finalidad mejorar la enseñanza del uso de ordenadores en las escuelas. Sin embargo esta pequeña placa es cada día más usada en cualquier ámbito de la vida social o educativa por su gran potencia y efectividad en tan poca cosa.

También posee un botón externo para realizar las funciones de on/off y reset, un led RGB, dos leds de estados, un conector HDMI, 3 conectores USB, un conector a Ethernet y expansión I2C y GPI para conectar otros dispositivos.

Para el funcionamiento del robot es necesario un transformador 12V /5 A que junto con la batería recargable da autonomía al robot. La batería tiene una autonomía de hasta dos horas en condiciones normales.

Las especificaciones técnicas de peso y dimensiones son las siguientes:

- Alto: 223 mm
- Ancho: 163 mm
- Largo: 165 mm
- Peso: 1Kg aprox.

Aisoy tiene diversos sensores y actuadores que se detallan a continuación.

1.3.1 Sensores

Cámara: Se encuentra situada en el ojo izquierdo como se muestra en la figura 2. Se trata de una cámara USB de 3 Mpx, con la cual se puede realizar detección facial y reconocimiento de objetos.

Micrófono: Situado en la misma placa de la cámara. Aisoy puede captar sonidos ya sean voces o música.

Acelerómetro: Sensor destinado al conocimiento de la posición e inclinación en 3D.

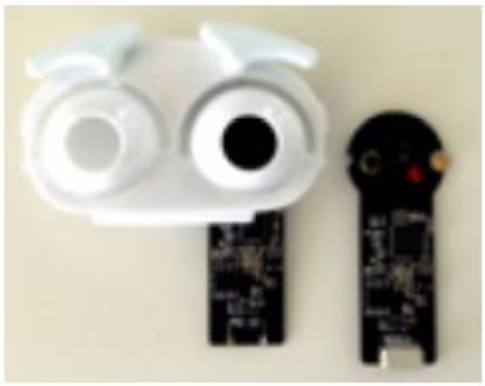


Figura 2: Cámara Aisoy

Sensores táctiles: Posee 5 sensores repartidos por su cuerpo para que el robot pueda saber cuándo le tocan o abrazan. (Véase figura 3).

Consumo de los servos: Aisoy puede determinar con este sensor si alguno de sus motores está bloqueado o haciendo demasiada fuerza.

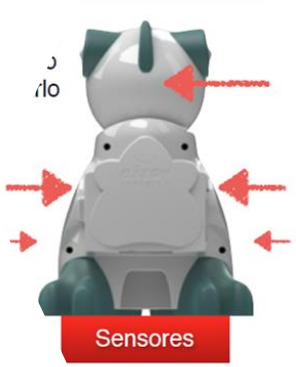


Figura 3: Posición de los sensores

1.3.2 Actuadores

Servomotores: Hace uso de 4 servomotores básicos; rotación de la cabeza, apertura y cierre de párpados y el movimiento de cejas hacia arriba y hacia abajo. (Véase figura 4). Y el Botmobile, opcional, que hace uso de 2 motores para moverse hacia adelante, hacia atrás y rotar.

Figura 4: Servomotores Aisoy

LED RGB: Mediante la combinación de tres led de color rojo, verde y azul, éste es capaz de mostrar cualquier combinación de colores. Se utiliza sobre todo para proporcionar información sobre el estado emocional. (Véase figura 5).

Figura 5: Corazón Aisoy

Altavoz: Integra un pequeño altavoz para reproducir cualquier tipo de sonido. Esta señal se amplifica con un amplificador de sonido por lo que se puede escuchar en la salida de audio de la Raspberry Pi. Junto con el micrófono forman dos de los elementos más importantes que le permiten mantener una conversación, haciendo uso de distintos software: para el reconocedor de voz utiliza ASR, Automatic, Speech Recognition y para sintetizar la voz, TTS, Text to Speech.

Pantalla: En la boca de Aisoy podemos ver una pantalla de resolución 128x64 pixeles como se muestra en la figura 6. En ella el robot muestra sonrisas, enfado, o cualquier texto o imagen creada por el usuario, así como también es dónde se muestra la dirección IP, el nivel de batería, la versión del sistema operativo o la red wifi a la que se encuentra conectado.

Figura 6: Boca Aisoy

1.4 Airos en Aisoy

Airos es el cerebro del robot, controla todo lo relacionado con el robot así como sucesos y toma de decisiones. Airos incorpora librerías de programación para el desarrollo de aplicaciones en lenguaje C++ y Python. El robot dispone de una aplicación Airos Manager para la gestión de los robots, permite cambiar su nombre además de conectarlo a la red wifi o actualizarlo. Con esta aplicación podemos ver información básica del robot como es la dirección de IP, dirección MAC o número de serie. Un requisito indispensable para la ejecución de la aplicación es tener instalados los servicios Bonjour en el ordenador.

Es posible conectar el robot mediante un cable de Ethernet. Simplemente conectando el cable desde el robot al PC o desde el robot al router y tener una conexión directa con la red.

Está construido sobre las características de la distribución Raspbian y ROS de Willow Garage.

Raspbian es un sistema operativo libre basado en Debian optimizado para el hardware de Raspberry Pi. Raspbian es algo más que un sistema operativo ya que ofrece más de 35.000 paquetes y software precompilado que hace más fácil la instalación en la Raspberry. Es el denominado sistema operativo no oficial de Debian Wheezy armhf para el procesador de Raspberry.

Mientras que ROS es también un sistema operativo pero tiene como objetivo simplificar la tarea de crear el comportamiento del robot, que es compleja y robusta, a través de una amplia variedad de plataformas robóticas. (Véase Anexo A).

Por tanto ambas cualidades de los dos sistemas hacen que Airos sea diferente. Entre los principales se encuentra la conectividad abierta, la facilidad de la programación de un robot emocional, la aplicación gratuita para gestionar tus robots y la opción de conectar accesorios como el Botmobile. [1]

Airos es capaz de almacenar los sucesos capturados por los sensores que posee evaluando lo que repercute cada uno de ellos sobre el robot. Con todos esos datos capturados es posible obtener lo que se denomina respuesta emocional y estado de ánimo del robot. En el siguiente gráfico (figura 7) se explica más esquemáticamente lo expuesto anteriormente.

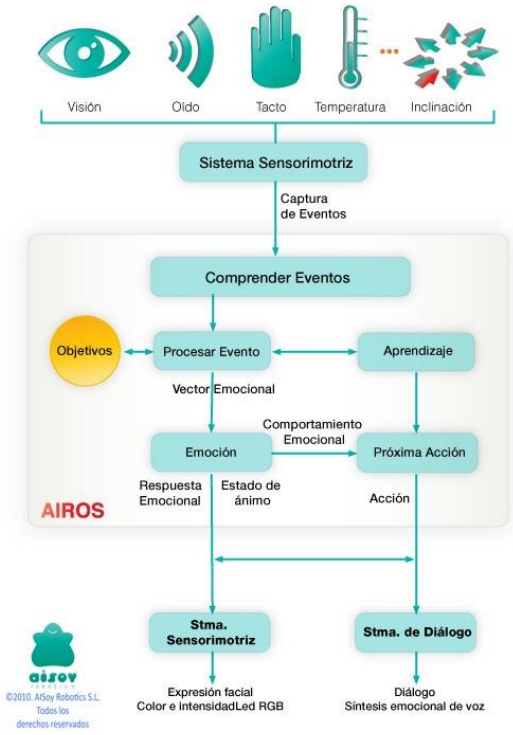


Figura 7: Esquema de respuesta emocional [2]

Es posible saber el estado de ánimo del robot debido a que puede ser reflejado en la expresión, tono de voz o en el corazón al cambiar de color. [2]

Aisoy1 hace uso de distintos servicios software que le permiten comportarse como un robot social. Las capacidades más importantes son:

• Reconocedor de voz (ASR, Automatic Speech Recognition): Se trata de una tecnología que permite a un ordenador identificar las palabras que una persona habla en un micrófono o teléfono y convertirlo en texto escrito. Aunque todavía no se encuentra en el punto en el que se pueda comprender toda el habla, es utilizado por un gran número de aplicaciones y servicios. Gracias a este servicio, el robot es capaz de entender lo que le dices.

• Sintetizador de voz (TTS, Text-to-Speech): Tts es un conversor de texto escrito a audio, se basan en crear una onda de audio correspondiente a cada fonema. Permite a Aisoy1 hablar de forma casi casi humana.

• Reconocimiento de caras, QR y colores: Gracias a la cámara incorporada y a su software mejorado, Aisoy1 es capaz de distinguir estos 3 elementos. Pueden ser usados en programas y juegos.

• Expresividad: La cara de Aisoy1 ha sido diseñada para ser muy expresiva. Esta cualidad es muy útil trabajando con emociones.

1.5 Programación

Aisoy se puede programar en dos niveles. En un nivel básico con la aplicación Scratch del MIT o un nivel avanzado en Python o C++. A continuación se hará una breve descripción de Scratch mientras que Python va a ser desarrollado en el apartado siguiente por ser objeto principal del presente proyecto.

1.5.1 Scratch

Scratch salió a la luz gracias al equipo de lifelong Kindergarten en el instituto de tecnología de Massachussetts apoyado financieramente por instituciones como Fundación Nacial de Ciencia, Microsoft o el Laboratorio de Medios del MIT. Su principal objetivo era crear un lenguaje de programación accesible para todos los públicos. [3]

Para la programación en Scratch hay dos modalidades: Scratch X u Offline. Ambas coinciden en la mayoría de los bloques de programación.

Scratch offline es una versión de escritorio. Se accede desde el navegador a un editor. En él se muestra la pantalla de Scratch en la cual se puede programar mediante la utilización de bloques. (Véase figura 8). Pero para ello es necesario incluir una extensión de Aisoy que se cargará mediante una plantilla proporcionada por Aisoy cuando se conecta el robot mediante la aplicación Airos Manager. Tal plantilla se ubicará en la misma carpeta donde se encuentra el ejecutable de Airos Manager y que tendrá la extensión .json.

Después de ello se conecta el PC al robot con un bloque específico de conexión que se encuentra en la extensión de bloques de Aisoy y posteriormente ya se puede hacer uso de todos los bloques de la extensión para poder realizar nuestro propio programa. [1]

Scratch X sin embargo es una versión online. Para su ejecución es necesario primero conectar el navegador

al robot para ello se ejecuta en la barra de dirección del navegador: https://IP_DE_SU_ROBOT:9090. Posteriormente será posible acceder a https://www.aisoy.es/pages/scratchx desde el navegador. [1]

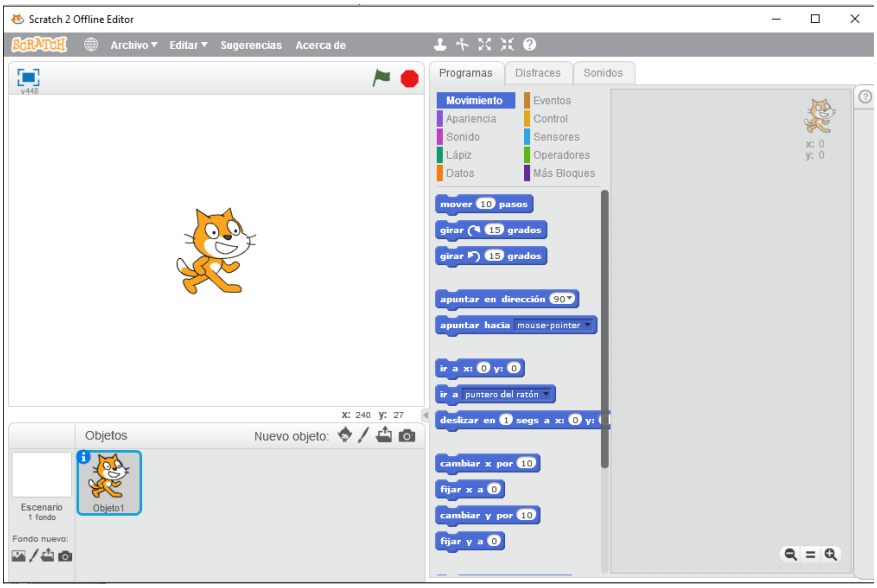


Figura 8: Ventana principal de Scratch

Algunos bloques básicos:

Conectar el robot a Scratch.

Figura 9: Bloque Scratch 1

Desconectar el robot.

Figura 10: Bloque Scratch 2

Lenguaje y tipo de voz (hombre o mujer).

Figura 11: Bloque Scratch 3

Diccionario reconocible por el robot (list). Esta lista deberá ser una lista propia de Scratch en el que previamente se ha de introducir las palabras que queramos que el robot reconozca por voz.

Figura 12: Bloque Scratch 4

Estado del robot. Cabe destacar el bloque block (no-block), se utiliza para determinación si un bloque es bloqueante o no. Si es bloqueante no pasará a la siguiente acción hasta que haya terminado mientras que si no es bloqueante podrá ejecutar la siguiente acción.

Figura 13: Bloque Scratch 5

Decir algo moviendo o sin mover la boca del robot.

Figura 14: Bloque Scratch 6

Figura 15: Bloque Scratch 7

Mover algunas partes del robot:

Mover el cabeza en unos determinados segundo en horizontal o vertical.

Figura 16: Bloque Scratch 8

Figura 17: Bloque Scratch 9

Bloques similares existen para el movimiento de las cejas y los ojos.

Figura 18: Bloque Scratch 10

Figura 19: Bloque Scratch 11

Cambiar el color de su corazón. Combinando las secuencias Rojo, Azul y Verde en el rango (0-255)

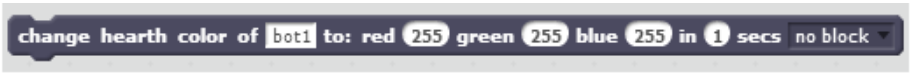


Figura 20: Bloque Scratch 12

Indicar el estado de la pantalla LED. Dispone de 70 caracteres. 0 significa apagado y x encendido.

Figura 21: Bloque Scratch 13

Bloques a ejecutar cuando el robot sea tocado, cuando esté en una posición determinada o cuando esté escuchando una frase determinada.

Figura 22: Bloque Scratch 14

Figura 23: Bloque Scratch 15

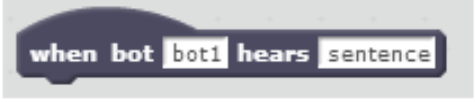


Figura 24: Bloque Scratch 16

Acciones para el Botmobile:

Mover hacia adelante durante unos determinados segundos.

Figura 25: Bloque Scratch 17

Mover hacia adelante indefinidamente.

Figura 26: Bloque Scratch 18

Parar el Botmobile.

Figura 27: Bloque Scratch 19

1.5.1.1 Ejemplo

Se ha creado un programa en Scratch a modo de demo para ilustrar algunas capacidades del robot.

En esta demo el robot pide al usuario recordar una serie de palabras que éste mismo menciona. Y posteriormente el usuario debe decir alguna, si el robot la reconoce, el robot continúa jugando. Luego el robot menciona una palabra en inglés y el usuario debe repetirla, cuando la reconoce, el usuario tiene la opción de seguir jugando o no. Para elle el robot le indica que le toque el lado derecho si no quiere seguir jugando.

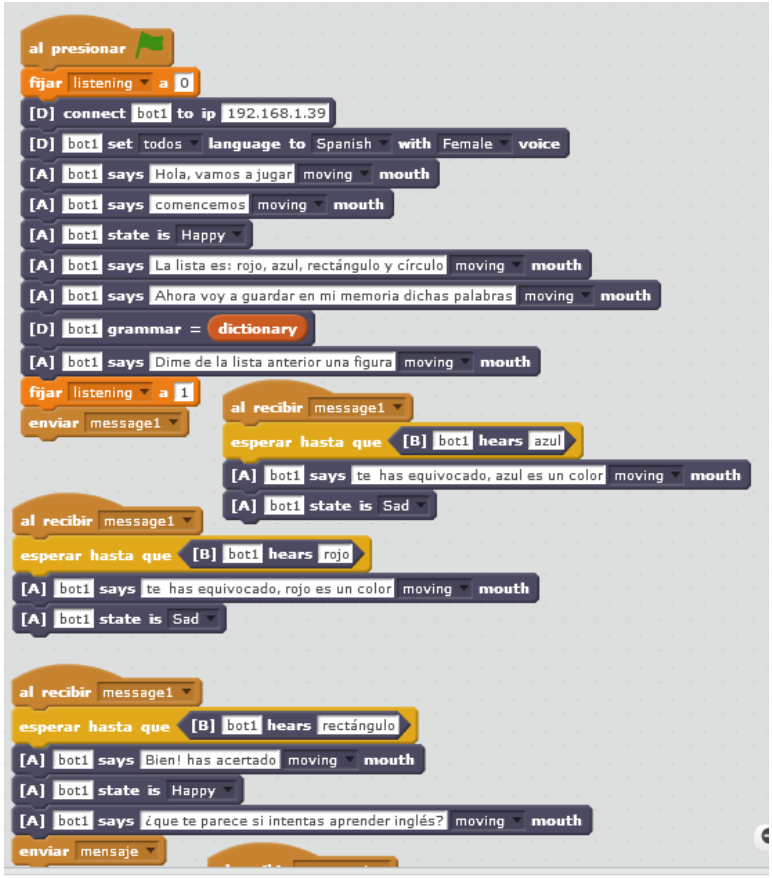


Figura 28: Ejemplo1 Scratch Parte1

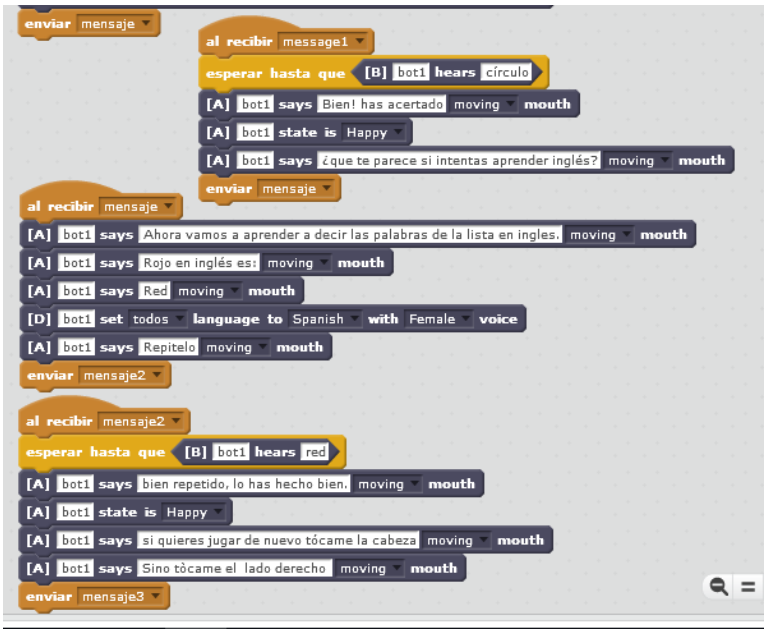


Figura 29: Ejemplo 1 Scratch Parte 2

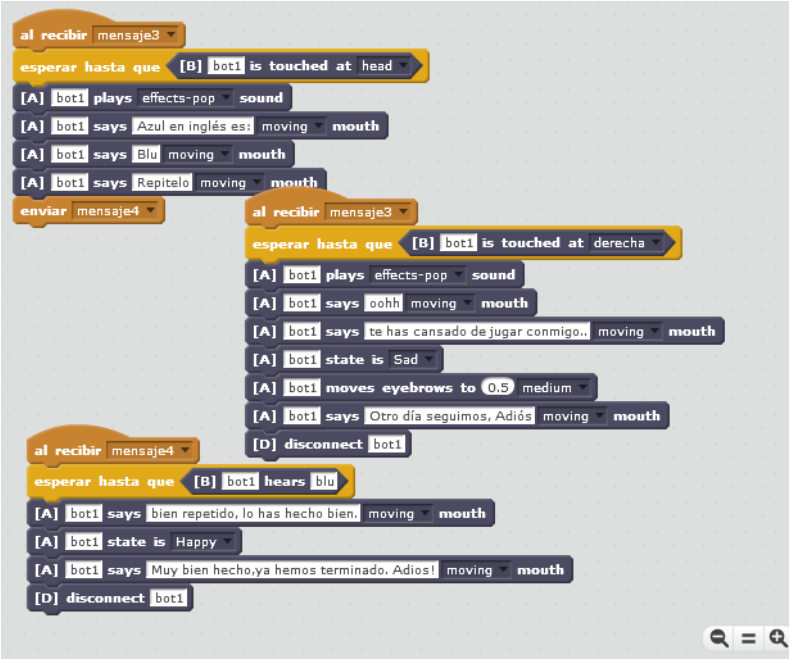


Figura 30: Ejemplo 1 Scratch Parte 3

2 AISOY Y PYTHON

2.1 Introducción

Python es un lenguaje de programación potente y fácil de aprender. Posee eficientes estructuras de datos de alto nivel y un enfoque sencillo pero efectivo de la programación orientada a objetos. La sintaxis elegante, los tipos de datos dinámicos y el hecho de ser un lenguaje interpretado hacen de Python un lenguaje ideal para la creación de scripts y el desarrollo rápido de aplicaciones en todo tipo de áreas y plataformas. [4]

Posee una licencia de código abierto, denominada Python Software Foundation License, la cual es compatible con la Licencia Pública General (GPL) del proyecto GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores. [5]

2.1.1 Historia

Python se remonta a finales de los ochenta, comenzando a implementarse en diciembre 1989. Por este tiempo, un investigador holandés llamado Guido van Rossum, que se encontraba trabajando en el centro de investigación CWI (Centrum Wiskunde & Informatica) de Ámsterdam, se le asignó un proyecto que consistía en el desarrollo de un sistema operativo distribuido llamado Amoeba. Por aquel tiempo, el CWI utilizaba un lenguaje de programación llamado ABC. En lugar de emplear este lenguaje para el proyecto Amoeba, Guido decide crear uno nuevo que pueda superar las limitaciones y problemas con los que se había encontrado al trabajar en otros proyectos con ABC. Así pues, fue ésta la principal motivación que dio lugar al nacimiento de Python. [6]

En 1991 salió a la luz la primera versión de este lenguaje, pero no es hasta tres años después cuando decide publicarse la versión 1.0. Inicialmente el CWI decidió liberar el intérprete del lenguaje bajo una licencia open source propia, pero en septiembre de 2000, coincidiendo con la publicación de la versión 1.6, se toma la decisión de cambiar la licencia por una que sea compatible con la licencia GPL (GNU General Public Licence). Esta nueva licencia se denominará Python Software Foundation Licence y se diferencia de la GPL al ser una licencia no copyleft. Este hecho implica que es posible modificar el código fuente y desarrollar código derivado sin la necesidad de hacerlo open source.

Hasta el momento solo se han liberado tres versiones principales, teniendo cada una de ellas diversas actualizaciones. En lo que respecta a la versión 2, la última en ser liberada fue la 2.7, en julio de 2010. Actualmente, la versión cuenta con la actualización 3.3, liberada en septiembre de 2012. Ambas versiones, la de 2 y 3, son mantenidas por separado. Esto implica, que tanto la 2.7 como la 3.3 se consideran estables pero, lógicamente, correspondientes a diferentes versiones. Muchos se preguntarán porque mantener ambas versiones y no seguir una evolución lógica. La respuesta a esta pregunta es fácil de responder: Entre ambas versiones existen diferencias que las hacen incompatibles. [6]

2.1.2 Versiones

Las versiones de Python se identifican por tres números X.Y.Z, en la que:

- X corresponde a las grandes versiones de Python (1, 2 y 3), incompatibles entre sí:

Los principales cambios introducidos en Python 2 fueron las cadenas Unicode, las comprensiones de listas, las asignaciones aumentadas, los nuevos métodos de cadenas y el recolector de basura para referencias cíclicas.

Mientras que los cambios introducidos en Python 3 fueron la separación entre cadenas Unicode y datos binarios, la función print (), cambios en la sintaxis, tipos de datos, comparadores, etc.

Por el momento, no hay planes de crear otra versión incompatible.

- Y corresponde a versiones importantes en las que se introducen novedades en el lenguaje pero manteniendo la compatibilidad (salvo excepciones).

Desde hace unos años, las versiones X.Y se publican aproximadamente cada año y medio y se mantienen durante cinco años, excepto la versión 2.7, que se mantendrá durante diez años, hasta 2020.

- Z corresponde a versiones menores que se publican durante el período de mantenimiento, en las que sólo se corrigen errores y fallos de seguridad.

Normalmente, se publica una última versión X.Y.Z justo antes de que una versión X.Y deje de mantenerse. Algunas empresas comerciales ofrecen el mantenimiento de versiones antiguas una vez acabado el mantenimiento oficial.

La figura 31 muestra la fecha de publicación de las versiones importantes de Python, en cada una de las tres grandes versiones, Python 1, Python 2 y Python 3. Las versiones indicadas con punto rojo se consideran obsoletas, las versiones indicadas con punto negro siguen publicando actualizaciones.

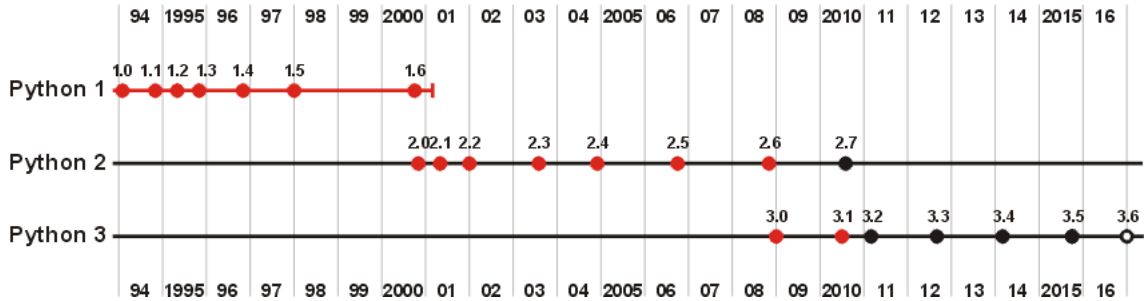


Figura 31: Versiones de Python

La transición de Python 2 a Python 3 ha resultado mucho más costosa de lo esperado, debido a que Python 3 introdujo muchos cambios en el lenguaje y obliga a reescribir prácticamente todos los programas. [6]

2.1.3 Introducción a la programación en Python

Para hacer uso de Python es tan fácil como acceder a la página web oficial y descargar la versión que se desee. Una vez descargado instalar siguiendo los pasos que aparecen.

Es un entorno de desarrollo, o más conocido por sus siglas en inglés IDE (Integraed Develoment Environmet), es un entorno de programación que ha sido empaquetado como una aplicación y suelen consistir en un editor de texto, un compilador, un depurador y una interfaz gráfica de usuario (GUI).Una de las ventajas de los IDE es que mientras el código es editado puede ser ejecutada en el IDE, y además te avisa de los posibles errores de sintaxis.

A continuación se van a detallar algunos de los aspectos más importantes de Python.

En Python cada archivo.py se denomina script, y al conjunto de ellos módulo. Estos módulos a su vez pueden formar paquetes. Un paquete es una carpeta que contiene varios archivos .py pero para q éste funcione debe contener un modulo.py que se llama __init__.py que puede estar vacío.

- Para importar módulos enteros se usa:

```
import  modulo
```

- Si queremos importar un módulo de un paquete:

```
import paquete.modulo1
```

- Para acceder a elementos del módulo se usan los siguientes comandos, dependiendo si el módulo está dentro de un paquete o no.

```
modulo.CONSTANTE_1
paquete.modulo1.CONSTANTE_1
```

- También es posible importar todos los elementos de un módulo:

```
from paquete.modulo1 import *
```

- Para definir funciones se procede de la siguiente forma:

```
def nombredelafuncion(valor1, valor2):
          return valor= valor1*valor2
```

La función anterior multiplica dos valores. Dichos valores son los correspondientes a los argumentos de entrada de la función.

- En las funciones también existen las llamadas recursivas:

```
def jugar(intento=1):
    respuesta = raw_input("¿Cómo se escribe cielo en inglés? ")
    if respuesta != "sky":
        if intento < 3:
            print "\nFallaste! Inténtalo de nuevo"
            intento += 1
            jugar(intento) # Llamada recursiva
        else:
            print "No lo has acertado, la respuesta es: sky!"
    else:
        print "\nBien has acertado!"
jugar()
```

Existe la posibilidad de buscar caracteres en un determinado texto.

- Para contar la cantidad de apariciones en una subcadena se realiza lo siguiente:

```
cadena = "bienvenido a mi aplicación".capitalice()
>>> print cadena.count("a")
3
```

- Para buscar una subcadena dentro de una cadena:

```
cadena = "bienvenido a mi aplicación".capitalize()
>>> print cadena.find("mi")
```

- Para saber si una cadena finaliza con una subcadena determinada:

```
endswith("subcadena" [, posicion_inicio, posicion_fin])
```

Devuelve true o false.

```
cadena = "bienvenido a mi aplicación".capitalize()
>>> print cadena.endswith("aplicación")
True
>>> print cadena.endswith("Bienvenido")
False
>>> print cadena.endswith("Bienvenido", 0, 10)
True
```

- Dar formato a una cadena sustituyendo dinámicamente:

```
cadena = "Importe bruto: ${0} + IVA: ${1} = Importe neto: {2}"
>>> print cadena.format(100, 21, 121)
Importe bruto: $100 + IVA: $21 = Importe neto: 121
```

2.2 Librerías Python: Open CV

OpenCV fue iniciado por Gary Bradsky en 1999, pero fue en 2000 cuando salió a la luz la primera versión. A Gary se unió Vadim Pisarevsky para gestionar el software, y en 2005 ganaron el DARPA Grand Challenge. Más tarde continuó su desarrollo con el apoyo de Willow Garage, con Gary Bradsky y Vadim Pisarevsky al frente del proyecto. Las siglas de OpenCV provienen del término anglosajón Open Source Computer Vision Library. Por lo tanto, OpenCV es una librería de tratamiento de imágenes, destinada principalmente a aplicaciones de visión por computador en tiempo real. [7]

Actualmente OpenCV es compatible con una amplia variedad de lenguajes de programación como C ++ , Python , Java , etc. , y está disponible en diferentes plataformas , incluyendo Windows , Linux , OS X , Android , iOS , etc.

Esta librería es multiplataforma, existiendo versiones para GNU/Linux, Mac OS y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos, calibración de cámaras, visión estérea y visión robótica.

Hoy, OpenCV está relacionado con una gran cantidad de algoritmos de la Visión por ordenador y se encuentra en continua expansión.

En comparación con otros lenguajes como C / C ++, Python es más lento. Sin embargo, Python posee otra característica importante como es que puede ser extendido fácilmente a otros lenguajes como C o C++. Esta característica nos ayuda a escribir los códigos de computación intensiva en C / C ++ y Python creando un conjunto de modo que podemos utilizar estas envolturas como módulos de Python. Esto proporciona una gran ventaja: nuestro código será tan rápido como el código original de C / C ++. Así es como funciona OpenCV en Python, es una envoltura alrededor de Python con base original de aplicación en C ++. OpenCV es un software de código abierto, para que cualquiera pueda hacer su contribución. Es por ello que se encuentra en continuo cambio al poder ser ampliado con diversos módulos creados por otros programadores.

En cuanto al tratamiento de imágenes con OpenCV, éstas se podrían representar como una función bidimensional de intensidad luminosa o nivel de gris, donde cada variable indicaría las coordenadas espaciales bidimensionales y el valor en cada punto sería igual a la intensidad luminosa de la imagen en ese punto.

Cada imagen está compuesta por una matriz de datos, es decir un conjunto ordenado de elementos, donde cada elemento se corresponde con un pixel y donde al brillo de cada píxel se le conoce como nivel de gris del mismo. De forma genérica, los valores de nivel de gris varían entre 0 y 255, donde los extremos son el blanco y el negro. Todos los demás valores representan distintas intensidades de gris. Por norma general, 0 es el color negro y 255 el blanco.

En cuanto al tipo de imágenes en OpenCV, existen 3 tipos básicos de imágenes y entre ellas difieren por la forma en la que son interpretados los elementos de la matriz de datos:

- Imagen binaria

Cada píxel asume uno de los valores discretos. En esencia estos dos valores se corresponden con blanco y negro. Una imagen binaria se almacena como una matriz bidimensional de 0 y 1. Cada píxel se corresponde con 1 bit.

Una imagen binaria puede considerarse como un tipo especial de imagen de intensidad, que contiene sólo el blanco y el negro. Una imagen binaria puede ser almacenada en una matriz double o uint8. Una matriz double es una matriz que almacena sus datos en 32 bit y sus valores son números en punto flotante tomándose su rango [0 ó 1] por convención, mientras que en una matriz uint8 se encuentran almacenados

en 8 bits, por lo que pueden almacenar 2^8=256 datos y su rango es [0 255]. Sin embargo, una matriz uint8 es preferible, ya que utiliza mucha menos memoria.

En la figura 32 se muestra un ejemplo de imagen binaria.

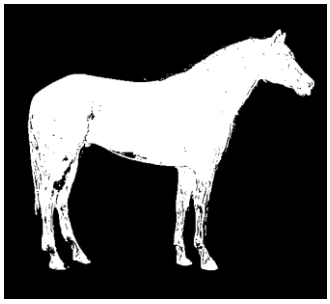


Figura 32: Imagen Binaria

- Imagen de intensidad

También llamada imagen en escala de grises. Consiste en una matriz de datos cuyos valores representan intensidades dentro de un mismo rango. La matriz puede ser doublé o uint8. Los elementos de la matriz representan diferentes intensidades o niveles de gris, donde el 0 representa el negro y el 255 el blanco. Cada píxel se corresponde con 1 byte, de ahí los 255 niveles permitidos.

En la figura 33 se muestra un ejemplo de una imagen en escala de grises.

Figura 33: Imagen Escala de Grises

- Imagen de color

También llamada imagen en formato RGB(Red, Green, Blue). Que en OpenCV para Python es BGR(Blue, Green, Red), tiene las mismas funciones pero el orden de los colores está cambiado. Representa cada color del píxel como un conjunto de tres valores, que representan las intensidades de rojo, verde y azul que componen el color. Cada píxel se codifica con 3 bytes (uno por color), siendo el número total de colores posibles del orden de 16,7 millones ($2^8 \cdot 2^8 \cdot 2^8$).

En la figura 34 se muestra un ejemplo de imagen a color.

Figura 34: Imagen a color

Para poder llevar a cabo el tratamiento de imágenes es necesario la digitalización de las mismas, se trata del paso del mundo continuo al mundo discreto. Para ello hay dos procesos, el muestreo y la cuantificación.

El muestreo es consecuencia de la discretización de las coordenadas espaciales en el sensor de luz. La imagen digital tiene un número infinito de puntos y cada uno de ellos representa una zona no infinitesimal de las escena.

Figura 35: Muestreo de una imagen

La resolución de cada una de las imágenes de la figura 35 es 512 x 512, 64 x64 y 32 x32 respectivamente. Se puede apreciar que cuanto menos pixeles sean discretizados menos calidad de la imagen.

La cuantificación es consecuencia de la discretización de la intensidad lumínica captada por el sensor, en nuestro caso, la cámara.

Figura 36: Cuantificación de una imagen

Los valores medio de la intensidad lumínica de las imágenes de la figura 36 son 256, 8 y 4 respectivamente.

2.2.1 Instalación de Open CV

Como se ha mencionado anteriormente es necesaria la instalación de previa de Numpy y opcionalmente la instalación de Matplotlib para el tratamiento de imágenes con OpenCV.

Matplotlib es una biblioteca de trazado para Python que le proporciona una amplia variedad en métodos de trazado. Numpy es también una extensión de Python que le agrega mayor soporte para vectores y matrices constituyendo una biblioteca de funciones matemáticas de alto nivel.

Cuando se hayan descargado ambos paquetes, se procede a su instalación. Dichos paquetes deben ser instalados en la carpeta de Python. Posteriormente es necesario descargar OpenCV si el software no lo trae ya instalado, se trata de un archivo comprimido .zip. Descomprimir e ir a la carpeta con la siguiente ruta OpenCV / build / Python / 2.7. Copiar el archivo cv2.pyd y pegar en la siguiente ruta: C: / python27 / lib / site- packeges.

Para acceder a esa ruta se ha descargado un software denominado FileZilla, que nos permite acceder a la información de la Raspberry mediante su dirección de IP, nombre de usuario y contraseña.

Aisoy ya trae diversas librerías instaladas, para saberlo se puede hacer uso del programa anterior y analizar el contenido. En este caso OpenCV ya viene viene instalada, al igual que numpy. Pese a ello, se ha explicado la instalación de esta librería para el ejecutable de Python.

Una vez realizados los pasos anteriores ya tendremos instalados Open CV. Para su uso en los programas de Python es necesario importar las dos librerías con el comando import.

```
import cv2
```

2.2.2 Funciones básicas de Open CV

2.2.2.1 Leer una imagen

Para leer una imagen se usa el comando cv2.imread (). Puede tener hasta dos argumentos de entrada. El primero obligatorio de trata de la imagen que deseamos leer y el segundo, opcional, especifica la manera que se quiere que sea leído. Existen tres posibilidades del citado segundo argumento:

- cv2.IMREAD_COLOR: carga una imagen en color.
- cv2.IMREAD_GRAYSCALE: Cargas imagen en escala de grises
- cv2.IMREAD_UNCHANGED: Carga la imagen como inclusión de canal alfa.

Estos argumentos se pueden sustituir por los números 1,0 y -1 respectivamente. Un ejemplo de su utilización se muestra a continuación:

```
img = cv2.imread('imagen1.jpg',0)
```

2.2.2.2 Mostrar una imagen

Para mostrar una imagen se usa la función cv2.imshow (), como argumento de entrada tiene en primer lugar el nombre que deseamos que lleve la imagen y en segundo lugar el nombre que hemos asignado a la imagen al leerla anteriormente.

```
cv2.imshow('image',img)
```

2.2.2.3 Guardar una imagen

La función usada es cv2.imwrite(). Al igual que las funciones anteriores posee dos argumentos de entrada, el primero para especificar el nombre que le queremos dar al archivo seguido del formato, y el segundo el nombre del archivo que deseamos guardar.

```
cv2.imwrite('imagen1gray.png',img)
```

Cuando se hace uso de estas funciones es recomendable usar también las siguientes funciones que se detallan para configurar las ventanas.

- cv2.waitKey () es una función de teclado. Su argumento de entrada es el tiempo en milisegundos. Devuelve un número que tiene correspondencia con una posición en el teclado. Por ejemplo el número 27 equivale a la tecla Esc, mientras que si queremos especificar la tecla lo haremos con el comando ord ('nombre de la tecla'). Esta función espera cualquier evento de teclado. La función espera los milisegundos especificados para cualquier evento de teclado. Si pulsa cualquier tecla en ese periodo, el programa continúa. Si llega a 0, se espera indefinidamente una pulsación de tecla. Pero lo más común es configurar para detectar pulsaciones de teclado. Esto sirve para que cuando el programa termine, el usuario pueda quitar la pantalla cuando desee, pulsando una determinada tecla la ventana desaparecerá. A continuación se muestra un ejemplo:

```
k = cv2.waitKey(0)
if k == 27:
cv2.destroyAllWindows()
elif k == ord('s'):
cv2.destroyAllWindows()
```

- cv2.destroyAllWindows ( ) destruye simplemente todas las ventanas que se hayan creado. Si desea destruir cualquier ventana específica, utilice la función de cv2.destroyWindow ( ) donde se pasa el nombre de la ventana exacta como argumento.

Con la biblioteca Matapolib descrita anteriormente es posible también llevar a cabo las funciones anteriores.

El color de una imagen cargada por OpenCV está en modo de BGR, como se ha comentado anteriormente. Pero Matplotlib muestra en el modo RGB, por lo que imágenes en color leídas por OpenCV CV no se muestran correctamente en Matplotlib. También existe otro modelo de color usado por open CV, el denominado HSV (Hue, Saturation, Value,) que se traduce como matiz, saturación y valor.

OpenCV proporciona una solución a este problema de modelos, la función cv2.cvtColor. Ésta función permite cambiar la imagen a un modelo u otro. A continuación se muestran ejemplos:

- De RGB a BGR:

```
cvtColor (img, img, COLOR_RGB2BGR)
```

- De BGR a HSV:

```
cvtColor(img, img, COLOR_BGR2HSV)
```

2.2.2.4 Dibujar sobre la imagen

Para dibujar sobre la imagen, Open CV también nos proporciona una serie de funciones que se configuran de manera similar. A continuación se exponen las básicas y sus configuraciones.

- `cv2.line (Imagen, coord. punto inicial, coord. punto final, color en modo BGR, tipo de línea)`
- `cv2.circle (imagen, coord. punto central, radio, color en modo BGR, tipo de línea)`
- `cv2.rectangle (imagen, coord. punto superior izq., coord. punto inferior derecho, color en modo BGR, tipo de línea)`
- `cv2.putText (imagen, texto, coord.. de la posición, tipo de fuente, tamaño de fuente, color )`

2.2.3 Clasificador de Haar

El clasificado de Haar es una herramienta de la librería Open CV. Fue el primer framework de detección de objetos propuesto por Paul Viola y Michael Jones en 2001 que permitía el análisis de imágenes en tiempo real, haciendo uso de una función matemática propuesta por Alfred Haar en 1909. [8]

Los clasificadores Haar están basados en diferentes regiones rectangulares ubicadas en imágenes en escala de grises como se puede ver en la figura 37. Como dicha imagen resultante se encuentra definida por los rectángulos anteriores se puede estimar como suma de pixeles, en base a un conjunto de clasificadores en cascada. Cada clasificador proporciona la información correspondiente de cada región, determinando si es lo buscada o no. A diferencia de otros algoritmos, éste solo invierte capacidad de procesamiento a las subregiones que posiblemente representen un rostro.

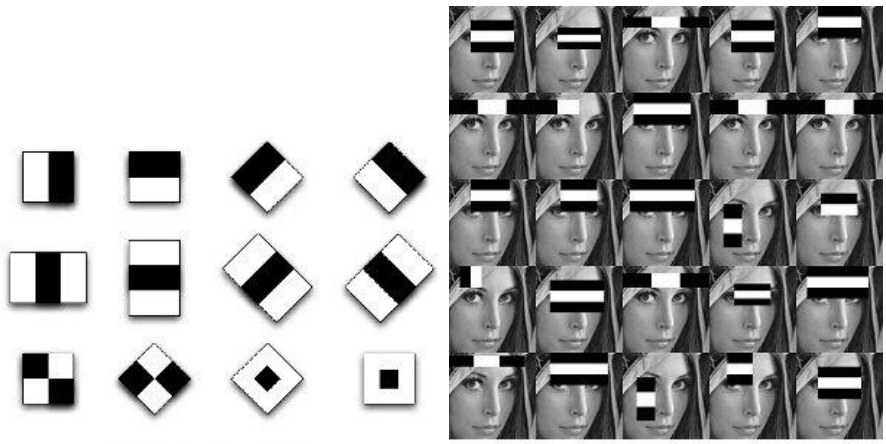


Figura 37: Método de clasificar de Haar

Para que funcione este tipo de clasificadores, es necesario entrenar con diversas imágenes. Para el procesamiento del clasificador es necesario dos tipos de muestras: positivas y negativas.

Las muestras positivas son imágenes con el objeto recortado que deseamos identificar. Las muestras negativas son los posibles entornos del objeto que se quiere identificar. Se estima que es necesario de 1000 a 10000 muestras negativas por cada muestra positiva.

Pero OpenCV nos facilita todo lo anterior con un archivo .xml que contiene toda esta información. Este archivo se lee con la función cv2.CascadeClasiffier (nombredelarchivo). Con ésta función se carga el archivo .xml lo que permite a la siguiente función identificar el rostro.

El siguiente paso es usar la función cv2. detectMultiScale (filtro, scaleFactor, minNeighbors, minSize, flags) la cual se encarga de detectar regiones a partir del archivo cascade que hemos cargado. Tiene varios argumentos de entradas, aunque no todos son necesarios.

El primer parámetro es el filtro a usar, en este caso es el filtro de escala de gris. El parámetro scaleFactor compensa las diferentes dimensiones de los posibles rostros, un valor intermedio es 1.2.

El parámetro minNeighbors le indica al detector cuantas veces tiene que pasar sobre la imagen para reducir el número de falsos positivos en la imagen, esto se debe a que el analizador hace un recorrido de la imagen en distintas posiciones y direcciones, así que de manera horizontal podemos deducir que algo se parece a un rostro, pero en la siguiente pasada aparecería otro objeto que no lo es.

Por último el parámetro minSize para indicar el tamaño mínimo del objeto, de otra forma será ignorado.

2.2.4 Template matching

Template matchig es otro herramienta de OpenCV con dos funciones básicas: cv2.matchTemplate y cv2.MinMaxLoc.

Antes de empezar es necesario recordar la utilización de imágenes en OpenCV como matrices. Éstas a su vez están formadas por varias capas de matrices de cada color. En OpenCV se usa el modelo BGR, es decir, tres colores, entre un rango 0-255.

Este método consiste en la comparación de plantillas para buscar y encontrar la ubicación de una plantilla en una imagen más grande. Para ello se desliza la imagen de la plantilla sobre la imagen de entrada y va comparando las regiones. Existen varios métodos de comparación implantados por Open CV. La función cv2.matchTemplate devuelve una imagen en escala de gris, donde cada pixel indica cuanto parecido tiene él y su vecindad con la plantilla. Esta función tiene como argumento de entrada la imagen real, la plantilla y el método en el que se quiere que se realice la búsqueda.

Si la imagen de entrada es de un tamaño (ancho (W) x alto (H)) y la imagen de la plantilla es de tamaño (ancho (w) x alto (h)), la imagen de salida tendrá un tamaño de (W-w + 1, H-h + 1). Una vez que ha obtenido el resultado, se puede utilizar la función cv2.minMaxLoc () para averiguar dónde está el valor máximo / mínimo. Dicho valor, es el valor de máxima coincidencia de la plantilla y la imagen, y éste se toma como la esquina superior izquierda del rectángulo y luego tomar w, h como la anchura y altura del rectángulo. Ese rectángulo es su región de plantilla. Es por esto que es necesario que la plantilla este realizada mediante una foto a la misma distancia aproximadamente que la foto de entrada. Dependiendo el método usado en la función cv2.matchTemplate, la esquina superior del rectángulo será máxima o mínima.

- Los métodos para que dicha esquina superior izquierda sea el máximo son:
  - ✓ cv2.TM_CCOEFF
  - ✓ cv2.TM_CCOEFF_NORMED
  - ✓ cv2.TM_CCORR
  - ✓ cv2.TM_CCORR_NORMED
- Mientras que los utilizados para que sea el mínimo de la matriz son:
  - ✓ cv2.TM_SQDIFF,

- ✓ cv2.TM_SQDIFF_NORMED

Con esta herramienta y un algoritmo adecuado es posible el seguimiento de objetos en la pantalla.

Es posible que al trabajar en escala de gris cualquier objeto con forma parecida sea también reconocido por el algoritmo. Además si el fondo de la imagen es blanco mejora la calidad de respuesta.

2.3 Programación de Aisoy

Para programar el robot Aisoy1, es necesario instalar un intérprete de órdenes. En este caso se ha hecho uso de la aplicación putty. AirosSDK contiene el intérprete de Python 2.7.3 instalado accesible desde la aplicación citada anteriormente. Con los módulos que contiene este sistema de fábrica es posible programar el robot desde Python para realizar algunas funciones. Sin embargo para usar funciones más avanzadas puede ser necesario instalar más librerías/paquetes o módulos.

Si queremos acceder a la información interna del sistema operativo, es posible. Como se explicó anteriormente me he descargado el programa FileZilla, que te permite a partir de una dirección de ip un usuario y una contraseña conectarte remotamente a otro puerto. Así con este programa puedo acceder a la Raspberry y ver todo lo que tiene instalado, así como cargar archivos en él. Para editar los archivos .py fuera de la aplicación putty es necesario volcarlos al local y posteriormente al remoto, entonces es cuando se hace uso de dicho programa FileZilla.

Crear un arhivo.py desde dicho intérprete es fácil, según se explica en el manual de usuario. Pero editar esos archivos puede complicarse un poco por el limitado manejo del cursor en él, por lo que se puede hacer uso del editor que se comentó anteriormente cuando te descargas el archivo .zip de Python.

En este proyecto también se ha hecho uso de otro editor de archivos .py denominado Sublime 3. Este editor me ha facilitado visualmente realizar programas, ya que posee gran variedad de colores asi como poder ir a una línea u otra del script con el cursor del ratón.

En este apartado se van a describir las herramientas internas de Aisoy. Es posible acceder a él mediante ssh y programar en Python con una serie de objetos:

- **Accel: Constructor**

Funciones de Accel:

Position = accel.get_position()

Devuelve información sobre la posición del robot: standup, facedown, left, right, fordward, backward.

Rpy = accel.get_rpy()

Devuelve información sobre la posición real del robot en tres valores (roll, pitch and yaw) en un vector.

- **Asr: constructor**

Funciones de Asr

Asr_name = asr.get_asr_name()

Devuelve el nombre de asr.

Vec_lang = asr.get_available_languages()

Devuelve un vector con los tipos de language: [en, es, fr]

Gramar = asr.get_grammar()

Devuelve el tipo de gramática actual.

Language = asr.get_language()

Devuelve el lenguaje actual.

Is_available = asr.is_language_available("en")

Devuelve un booleano indicando si el lenguaje que se pone como argumento de entrada está disponible.

Recognized = asr.listen()

Devuelve la palabra o frase que haya sido reconocida. Es posible introducir un número como argumento de entradas que será el tiempo que esperará para reconocer alguna palabra o frase.

Para cambiar el tipo de asr se usa la función: asr.set_asr ("pocketsphinx")

Para configurar la gramática se usa la función: asr.set_grammar ("one|two|three")

Para configurar un nuevo lenguaje: asr.set_language ('en')

- **Audio : constructor**

Audio_list = audio.get_audio_list()

Devuelve un vector con todos los sonidos disponibles.

Para reproducir alguno de los sonidos anteriores: audio.play("animal.kitten")

Para parar un audio: audio.stop(pid)

El argumento de entrada pid es el valor de pid del audio que se desea parar. También existe la función audio.stop_all() para parar todos los sonidos que se estén reproduciendo en ese momento.

- **Base: constructor.**

Funciones para mover la base del robot:

Base.move ("forward", 1, 100). Los argumentos de entradas pueden ser tres: el primer argumento puede ser: forward, backward, left, right dependiendo del lugar hacia el que queremos que se mueva el robot. El segundo argumento es el tiempo que ha de tardar en realizar el movimiento. El tercer argumento es la velocidad que se ha de utilizar.

Para parar los movimientos: base.stop()

- **Color: constructor**

Método para hacer flashes de un color: color.flash_color (0,0,255,3,0.5)

Los tres primeros argumentos determinan el color. El segundo argumento es el tiempo que estará flasheando el robot y el tercero el tiempo de cada flash.

Rgb_value = color.get_color ()

Devuelve el valor del color actual: red, blue y green.

Cambiar el color: color.set_rgb(255,0,0,3).

Los tres primeros argumentos son el color al que se desea cambiar y el tercero el tiempo en que se desea que se produzca el cambio.

La función anterior también puede ejecutar con el comando de entrada: Green, blue o red para los colores en lugar de los tres valores numéricos.

Para borrar la cola: color.stop_all()

- **Display :constructor**

Se usa para abrir una imagen o un archivo, escalar y convertir a blanco y negro para poder mostrarlo en la pantalla del robot.

Funciones:

Display.draw ("/usr/share/airos5/images/logo/aisoy_logo.png")

Muestra en la pantalla del robot la imagen que se encuentre en esa dirección del argumento de entrada.

```
Display.draw_array (
"                  "+
"                  "+
"                  "+
"                  "+
"                  "+
" x              x "+
"  x            x  "+
"   xxxxxxxx    "+
"                  "+
"                  ")
```

Content = display.get_content ()

Devuelve el contenido de lo que está mostrando la pantalla como cadena de caracteres

Speed = display.get_speed ()

Devuelve la velocidad con la que la pantalla va mostrando una serie de textos o cadenas de caracteres.

Type = display.get_type ()

Devuelve el tipo de muestreo por pantalla. Puede ser: "print", "draw" y "image_file"

Para escribir una cadena de caracteres podemos usar la función display.write("test",10). El segundo argumento indica el tiempo definido para escribir la cadena.

- **Face : constructor**

Con este objeto constructor solo se define una función. Dicha función se usa para la detección de caras.

Face_detecte = face.is_detected (10).

Devuelve verdadero si ha detectado una cara y falso en caso contrario.

El argumento es el tiempo en segundos que ha de esperar para el reconocimiento de una cara. Si pasado ese tiempo no la detecta devuelve falso.

- **Performance: constructor**

Emotion = performance.get_emotion()

Devuelve el estado emocional actual.

Para actualizar el estado emocional se usa la función: performance.set_emotion(“happy”)

Los posibles estados emocionales del robot son: normal, sad, happy, angry, indifferent, surprise, disgust, relief, reproach, pride, admiration, scared, sleep y noemotion.

- **Qr : Constructor**

Al igual que para el reconocimiento de caras Aisoy posee una función para el reconocimiento de códigos qr:

Qr_code = qr.get_code(10)

Devuelve el código reconocido.

Servo: constructor

Position = servo.get_servo_pos (“head_h”)

Devuelve la posición de un servo en concreto, determinado en el argumento de entrada. Los argumentos de entradas posibles son: “head_h”, “head_v”,”eyelids” y”eyebrows”.

Para mover esos servos se usa la función: servo.move_servo(“head_h”,50,1). El segundo argumento es la posición a la que se quiere mover el servo y el tercer argumento es el tiempo en segundos que debe tardar en realizar el movimiento.

Si se quiere que el robot haga un movimiento relativo se usará la función: ser.move_servo_relative (“head_h”,50,1)

- **Touch: constructor**

Is_touch = touch.is_touched (“head”)

Devuelve un 1 si el sensor es tocado y un 0 en caso contrario.

El argumento de entrada puede ser: "head","back_left","back_right”,”belly_left”,”belly_right”,"any"

Touch.wait_for_touch (“head”,10)

Esta función espera a que el sensor especificado en el primer argumento sea toca. También es posible definir un tiempo de espera que se hará con el segundo argumento.

Touches = touch.which_touched (10)

Devuelve que sensor está siendo tocado.

- **Tts: constructor**

Vec_lang = tts.get_available_languages ()

Devuelve un vector con los lenguajes disponibles.

Vec_voices = tts.get_get_available_vioces ()

Devuelve un vector con los tipos de voces disponibles.

Emotions = tts.get_emotion ()

Devuelve el estado emocional actual como cadena de caracteres.

Language = tts.get_language()

Devuelve el lenguaje actual.

Tts_name = tts.get_name()

Devuelve el nombre de tts.

Voice= tts.get_voice ()

Devuelve el tipo de voz actual.

Volumen = tts.get_volume ()

Devuelve el nivel del volumen en un rango de 0-100%.

Is_ available = tts.is_language_available ("en")

Devuelve un booleano indicando si el lenguaje especificado en el argumento de entrada está disponible.

Is_ available = tts.is_voice_available ("kal_diphone")

Devuelve un booleano indicando si la voz especificada en el argumento de entrada está disponible.

Para hacer que el robot diga una frase se usa la función: tts.say ("hello")

Para actualizar la emoción: tts.ser_emotion ("happy")

Para cambiar de lenguaje: tts.set_language ("en")

Para cambiar el tipo de tts: tts.set_tts ("pico")

Los argumentos de entrada posible son: "festival","espeak" y "pico"

Para cambiar una voz tts.set_voice ("kal_diphone")

Para cambiar el volumen: tts.set_volume(100)

- **Visión: constructor**

Con la siguiente función es posible determinar si los ojos del robot están cubiertos o no.

Dark = visión.is_dark()

Devuelve un booleano para saber si los ojos están cubiertos o no.

No obstante, se puede usar cualquier herramienta o librería que aporte Python, el SDK de Airos es simplemente una herramienta para interactuar con el robot, pero se puede usar Python sin usar para nada Airos.

2.3.1 Ejemplos de programas en lenguaje Python en Aisoy

2.3.1.1 Ejemplo 1: Reconocimiento de voz

Este ejemplo es un sencillo ejemplo de iniciación a Aisoy, necesario para hacerse con las funciones básicas del robot. Para el reconocimiento de palabras del robot es necesario guardarlas antes. En este programa el robot dice una serie de palabras entre las cuales hay figuras y colores. El usuario debe repetir una de las anteriores que sea una figura. En la figura 38 se muestra el código.

```
from airos5sdk.asr import *
from airos5sdk.tts import *
from airos5sdk.touch import *
tts = Tts()
asr = Asr()
touch = Touch()
asr.set_language('es')
tts.set_language('es')
asr.set_grammar("Rojo|verde|azul|amarillo|cuadrado|rectangulo|circulo")
grammar_lang={
    'es':"rojo|verde|azul|amarillo|buenos dias|buenas tardes"
}
tts.say("vamos a comezar")
tts.say("mis palabras son: Rojo, azul, verde, amarillo, cuadrado, rectangulo y circulo ")
for i in [0,1,2]:
    tts.say("Dime de las palabras anteriores cual es una figura")
    palabra = asr.listen(10)
    print"La palabra es: "+palabra
    if palabra == '':
        tts.say(" no he reconocido ninguna palabra")
    else:
        if palabra != 'cuadrado':
            if palabra!='rectangulo':
                if palabra!='circulo':
                    tts.say("Te has equivocado, me has dicho un color, no una figura")
                else:
                    tts.say("bien has dicho una palabra que no sea un color y es :  "+palabra)
            else:
                tts.say("bien has dicho una palabra que no sea un color y es :  "+palabra)
        else:
            tts.say("bien has dicho una palabra que no sea un color y es : "+palabra)
```

Figura 38: Ejemplo 1 Python Aisoy

2.3.1.2 Ejemplo 2: Pantalla Aisoy

Se ha construido un otro ejemplo para hacer uso de la capacidad de Aisoy de mostrar en su pantalla distintas imágenes. En primer lugar se le pregunta al usuario si desea jugar. Si la contestación es sí, se procede a mostrar por pantalla una imagen con la figura y el usuario debe responder que figura es. Si el robot no reconoce ninguna palabra pedirá al usuario una interacción con sus sensores para seguir; si desea volver a intentarlo debe toca el lado derecho, sino el izquierdo. Como se puede ver en el programa cuando se hace dicha detección de sensores hay dos posible sensores derecho, para que el programa continúe si toca el superior o el inferior. El programa se extiende para tres imágenes distintas aunque en la figura 39 y 40 solo se muestre el procedimiento de una.

Para la ejecución de este programa es necesario el uso de FileZilla, explicado anteriormente. Es necesario crear las imágenes del círculo, cuadrado y triangulo en archivos .jpg o .png y volcarlos a la Raspberry. La carpeta es independiente, en este caso me he creado una propia que se denomina “images” para introducir todas las imágenes.

```
from airos5sdk.tts import  *
from airos5sdk.asr import *
from airos5sdk.display import  *
from airos5sdk.performance import *
from airos5sdk.touch import *
touch = Touch()
asr = Asr()
tts = Tts()
performance = Performance()
display = Display ()
tts.set_language('es')
asr.set_language('es')
asr.set_grammar("si|no|triangulo|cuadrado|circulo")
tts.say("Hola soy Aisoy1, ,  quieres jugar el reconocimiento de formas")

cont = 1
while cont == 1:
    resp = asr.listen(10)
    if resp == '':
        tts.say("no he reconocido niguna palabra")
        tts.say ("vuelve a intentarlo")
    else:
        if resp == 'no':
            tts.say(" bueno pues en otra ocasion jugamos, Adios")
            performance.set_emotion("sad")
            exit()

        else:
            if resp == 'si':
                tts.say ("bien, vamos a comenzar")
                performance.set_emotion("happy")
                cont = 0

tts.say ("segunda figura, te la mostrare durante 10 segundos")
contador =0
while contador<70:
    display.draw("/home/pi/images/circulo.png")
    contador =contador +1

tts.say("Dime que era")
continuar =0
```

Figura 39: Ejemplo 2 Python Aisoy Parte1

```
while continuar == 0:
    tts.say("dime la figura")
    figura = asr.listen (10)
    print ("figura reconocida: "+figura)
    if figura == 'triangulo':
        tts.say("no es un triangulo, lo querrias reintentar")
        reintento=asr.listen(10)
        print("repuesta al reintento:" +reintento)
        if reintento == 'si':
            continuar =0
            print("reintentamos")
        else:
            if reintento == 'no':
                continuar = 1
            else:
                tts.say("no te he entendido, pero vamos a reintentarlo, confio en ti")
                continuar =1
    else:
        if figura == 'cuadrado':
            tts.say (" no es un cuadrado, quieres intentarlo de nuevo")
            resp2 = asr.listen(10)
            if resp2 == 'no':
                continuar  = 1
            else:
                continuar = 0
        else:
            if figura == 'circulo':
                tts.say ("bien, has acertado")
                continuar =1
            else:
                tts.say("no he reconocido niguna palabra ")
                tts.say("si quieres volver a intentarlo toca el lado derecho, sino el izquierdo")
                sensor = touch.which_touched(20)
                if sensor == 'belly_right' or sensor == 'back_right' :
                    continuar =0
                else:
                    if sensor == 'belly_left' or sensor == 'back_left':
                        continuar =1

tts.say("continuarmos con otra figura")
```

Figura 40: Ejemplo 2 Python Aisoy Parte 2

2.3.1.3 Ejemplo 3: Clasificador de Haar

En este apartado se ha desarrollado el clasficador de Haar explicado anteriormente. Para ello se han realizado diversos programas para la detección de diversas partes de la cara y otro para la detección de objeto. Debido al continuo desarrollo del software del robot y los posibles bugs que puede contener estos programas no realizaban lo previsto pese a tener la librería OpenCV instalada en airos. Es por ello que para realizar las pruebas se ha usado el ejecutable de Python, la librería OpenCV y la webcam del ordenador danod un resultado satisfactorio.

2.3.1.3.1 Detección de caras

Cada día se utiliza en un mayor número de dispositivos y aplicaciones la detección facial, como Smartphone, tabletas, ordenadores o cámaras. El gran inconveniente de esta nueva tecnología es el posicionamiento del rostro en la fotografía, ya que si el rostro no está más o menos de frente, probablemente no lo reconozca el sistema y puede inducir a errores. Es por ello que la detección facial se encuentra en área de investigación y en continuo avance.

Es importante diferenciar entre detección facial y reconocimiento facial, hoy en día ambos términos son muy utilizados y pueden llevar a confusión. La detección facial únicamente proporciona información de la ubicación de una cara mientras que el reconocimiento puede darnos información más precisa como por ejemplo si el rostro de un hombre, si es un familiar… etc.

Existen diversas formas de llevar a cabo este algoritmo. En este documento se hará uso de los clasificadores de Haar explicado anteriormente.

Con la función cv2.detectMultiScale podemos obtener las coordenadas de la ubicación del rostro. Con esas coordenadas dibujamos un rectángulo en la región con la función explicada anteriormente cv2.rectangle. En la figura 41 se muestra el código del programa y en la figura 42 el resultado del mismo.

```python
import numpy as np
import cv2


# cargar archivo xml donde estan los algoritmos de reconocimiento
cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cascade.load('haarcascade_frontalface_default.xml')
captura = cv2.VideoCapture(0)
print("inicia el programa")
#cargamos la imagen
_,img = captura.read()
gris = cv2.cvtColor(img , cv2.COLOR_BGR2GRAY)
gris = np.array(gris, dtype='uint8')
caras = cascade.detectMultiScale(gris, 1.3, 5)
print caras
#buscamos todas las caras de la imagen
for (x,y,w,h) in caras:
    cv2.rectangle(img,(x,y),(x + w , y + h),(125,255,0), 2)


cv2.imshow('Detección', img)


k = cv2.waitKey()
if k == ord ('s'):#guardamos imagen y cerramos
        cv2.imwrite('rostro.png', img)
        cv2.destroyAllWindows()
else:
#sino se pulsa tecla no guardamos imagen y cerramos ventana
        cv2.destroyAllWindows()

```

Figura 41: Programa detección de caras

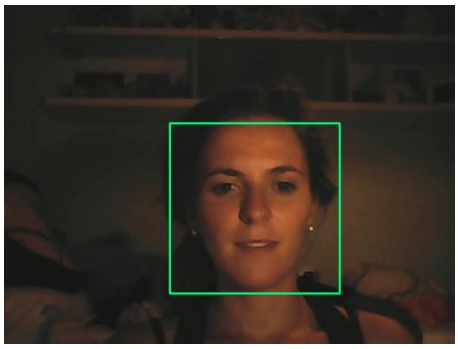


Figura 42: Resultado detección de cara

2.3.1.3.2 Detección de boca

Para continuar con el clasificador, se ha programado la detección de sonrisa.

La detección de sonrisa también está cada vez más involucrada en los nuevos dispositivos, con el algoritmo condicionante de si sonríes la cámara hace la foto. Se introdujo esta función originalmente en las cámaras digitales pero el concepto se ha extendido a las cámaras de los teléfonos móviles y las video cámaras, y tiene posibles aplicaciones para sistemas de seguridad

En lo referente a su programación, es igual que el ejemplo anterior de detección de caras pero el archivo .xml de las muestras es diferente. Ahora este archivo solo contiene muestras de sonrisas y no de caras enteras como el anterior. También es proporcionado por OpenCV 'haarcascade_smile.xml'. A continuación se muestra el programa y el resultado en las figuras 43 y 44 respectivamente.

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
smile_cascade = cv2.CascadeClassifier('haarcascade_smile.xml')

cap = cv2.VideoCapture(0)
i=1

while i==1:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    print faces
    i=2
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

    smile = smile_cascade.detectMultiScale(roi_gray,1.3,20)
    print smile
    for (ex,ey,ew,eh) in smile:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imshow('img',img)
cap.release()
k = cv2.waitKey()
if k == ord ('s'):#guardamos imagen y cerramos
        cv2.imwrite('rostroconrisa.png', img)
        cv2.destroyAllWindows()
else:
#sino se pulsa tecla no guardamos imagen y cerramos ventana
        cv2.destroyAllWindows()

```

Figura 43: Programa de detección de sonrisa

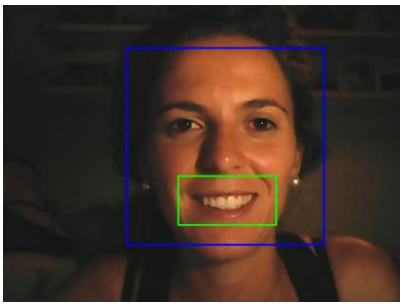


Figura 44: Resultado de detección de sonrisa

2.3.1.3.3 Detección de ojos

OpenCV también nos facilita la posibilidad de detección de ojos, siendo posible por ejemplo detectar si se está guiñando un ojo. El algoritmo es similar al anterior pero el fichero .xml cambia de nombre.

El comando leng() que se puede ver en el programa desarrollado en la figura 45 permite saber el número de ojos detectados, al igual que si el argumento de entrada fuera faces, proporcionaría el número de caras detectadas.

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

cap = cv2.VideoCapture(0)
i=1

while i==1:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    print faces
    i=2
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

        eyes = eye_cascade.detectMultiScale(roi_gray)
        for (ex,ey,ew,eh) in eyes:
            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

print len(eyes)
cv2.imshow('img',img)
cv2.imwrite('deteccionojos.jpg', img)
cap.release()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figura 45: Programa detección de ojos

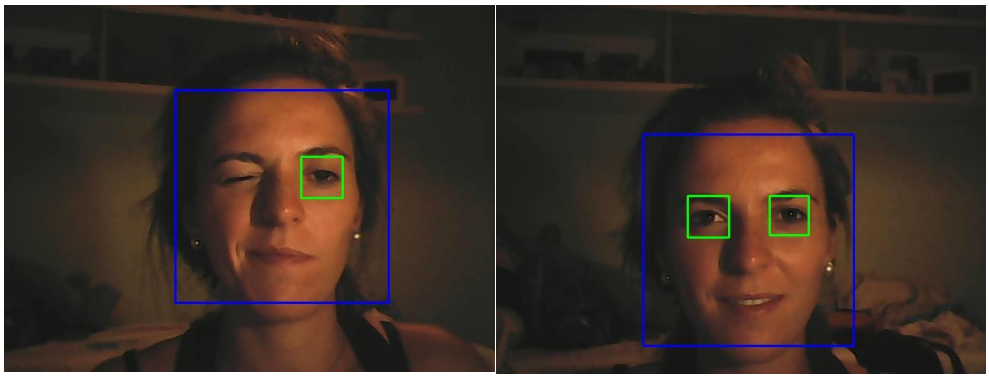


Figura 46: Resultado de detección de ojos

2.3.1.4 Ejemplo: Template Matching

Por el msimo motivo explicado en el apartado anterior este ejemplo se ha realizado con el ejecutable de Python, la librería OpenCV y la webcam del ordenador dando un resultado satisfactorio.

2.3.1.4.1 Detección de objetos

Se ha utilizado una plantilla que es una imagen de una funda de gafas. En la figura 47 se detalla el programa utilizado. En la figura 48 se expone una captura de pantalla del programa en ejecución donde se puede apreciar la no presencia del rectángulo mientras que en la figura 49 se ha detectado la funda y por tanto dibuja el rectángulo sobre ella.

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)
ret, frame = cap.read()

template = cv2.imread('fundarosa.jpg',0)
#devuelve el ancho y el alto de la figura en la plantilla
w,h = template.shape[::-1]
x,y = 0,0
print w, h

while(True):
    ret, frame = cap.read()
    i= cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
#crea una matriz de valores de semanjaza con la imagen principal
    res = cv2.matchTemplate(i,template,3)
#min_val y max_val son los valores maximo y minimo de la matriz anterior
#max_loc y min_loc son puntos correspodientes a los datos anteriores
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
    top_left = max_loc
    bottom_right = (top_left[0] + w, top_left[1] + h)
    x = (top_left[0] + bottom_right[0])/2
    y = (top_left[1] + bottom_right[1])/2
    print x , y

    #El valor de la x cambia cuando ponemos el objeto delante, por lo que se usa de flag
    #para determinar cuando dibujar el rectangulo de deteccion
    if x<=547:
        cv2.rectangle(frame, top_left, bottom_right, (0, 255, 0), 3)

    cv2.imshow('output',frame)
   # out.write(frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.release()
        break

cv2.destroyAllWindows()
```

Figura 47: Programa detección de objeto

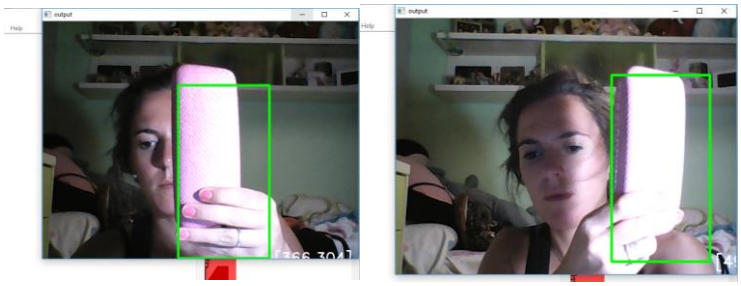


Figura 48: Resultado detección de objeto

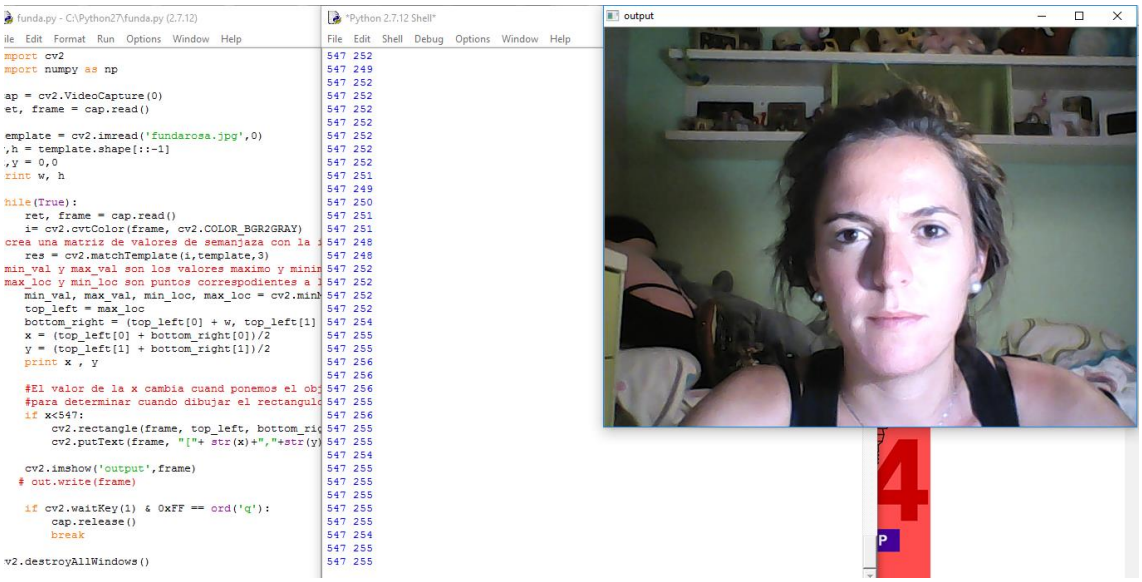


Figura 49: Programa en ejecución sin figura

3 INTEGRACIÓN DE AISOY EN DOMÓTICA

3.1 Introducción

> Se entiende por domótica el conjunto de sistemas capaces de automatizar una vivienda, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado.
>
> *- wikipedia -*

El término domótica viene de la unión de las palabras domus (que significa casa en latín) y tica (de automática, palabra en griego que significa que 'funciona por si sola'). [9]

3.2 Historia

Para hablar del comienzo de la domótica volvemos a los años setenta, cuando después de numerosos experimentos e investigaciones llegaron los primeros dispositivos automatizados en edificios relativamente nuevos de aquella época basada en la tecnología X-10, esta tecnología es la más básica y asequible para la domótica usando una modulación sencilla.

En los años siguientes fueron numerosos países los que se involucraron en la búsqueda de nueva información sobre esta exitosa tecnología. Esta tecnología se implantaría en un principio en edificios de oficinas buscando una mejora de la calidad de vida de los trabajadores aunque posteriormente se fueron implantando también en naves industriales y hogares.

Una de las primeras ciudades que decidió implantar la domótica fue Estado Unidos con un sistema de regulación de temperatura ambiente en pequeños edificios.

A finales de los ochenta y principio de los noventa, cuando se produjo el avance y el gran interés por los PC's, dio lugar a que se empezaran a instalar en dichos edificios un cableado que pondría facilidades para la conexión de todos los aparatos y periféricos necesarios para la domótica en un edificio. Este cableado permitía el transporte de datos e interconexión de dispositivos de seguridad, dichos edificios comenzaron a llamarse edificios inteligentes.

Posteriormente, todos estos automatismos destinados a edificios de oficinas se han ido aplicando también a las viviendas de particulares u otro tipo de edificios dónde el número de necesidades que hay que cubrir es mucho más amplio, dando origen a la vivienda domótica.

Sin embargo en España la domótica no se empezó a implantar hasta primeros de los años noventa cuando se dieron cuenta que poseía numerosos beneficios. Aun así hasta hace poco no se está llevando a cabo investigaciones e instalaciones más notables en la domótica. En la actualidad el número de viviendas o edificios comotizados es todavía muy escaso, pero el interés por esta tecnología continúa aumentando.

El significado de domótica se encuentra en continuo cambio. Es por ello que a día de hoy también se suele utilizar el concepto de vivienda inteligente, ya que la domótica busca la integración de todos los sistemas del hogar, de forma que funcionen todos en perfecta armonía, con la máxima utilidad y la mínima intervención por parte del usuario. [10]

3.3 Características

Los servicios que ofrece la domótica se pueden agrupar según cinco aspectos o ámbitos principales:

3.3.1 Ahorro energético

El ahorro energético es un concepto al que se puede llegar de muchas maneras. En muchos casos no es necesario sustituir los aparatos o sistemas del hogar por otros que consuman menos energía sino una gestión eficiente de los mismos.

Algunos ejemplos son los siguientes:

- Climatización y calderas: Para ello se puede programar y zonificar, utilizando un termostato. Se pueden encender o apagar la caldera usando un control de enchufe, mediante telefonía móvil, fija, Wifi o Ethernet.
- Control de toldos y persianas eléctricas: Realizando algunas funciones repetitivas automáticamente o bien por el usuario o manualmente mediante un mando a distancia. Por ejemplo para proteger el toldo del viento se puede usar un sensor de viento que actúe sobre los toldos cuando sea necesario. O para la protección del sol, mediante otro sensor de sol.
- Gestión eléctrica: Se puede racionalizar el uso de cargas eléctricas mediante la desconexión de equipos de uso no prioritarios en función del consumo eléctrico en un momento dado.
- Gestión de tarifas: Es posible derivar el funcionamiento de algunos aparatos a horas de tarifa reducida.
- Uso de energías renovables.

3.3.2 Confort

Se entiende por confort todos aquellos servicios de la domótica enfocados a obtener el bienestar por medio de ciertos dispositivos que trabajen para nosotros ya sea pasivo, activo o mixto. Algunos aspectos que mejoran nuestro confort en un sistema domótico son:

- Iluminación:
  - ✓ Apagado general de todas las luces de la vivienda.
  - ✓ Activación o desactivación de la iluminación por presencia.
  - ✓ Automatización del apagado/encendido en cada punto de luz.
  - ✓ Atenuación de las luces por franja horaria.

- ❖ Automatización de todos los distintos sistemas/instalaciones/dotándolos de control eficiente y de fácil manejo.
- ❖ Integración del portero al teléfono, o del videoportero al televisor.
- ❖ Control vía Internet
- ❖ Gestión Multimedia y del ocio electrónico.
- ❖ Generación de macros y programas de forma sencilla para el usuario y automatización.

3.3.3 Seguridad

Un aspecto fundamental de los sistemas domóticos consiste en una red de seguridad encargada de proteger tanto los bienes patrimoniales, como la seguridad personal y la vida.

Algunos ejemplos destacados son:

- ❖ Alarmas de intrusión (antiintrusión): Se utilizan para detectar o prevenir la presencia de personas extrañas en una vivienda o edificio.
- ❖ Detectores y alarmas de detección de incendios.
- ❖ Alerta médica y teleasistencia.
- ❖ Acceso a cámaras IP.

3.3.4 Comunicaciones

La domótica no puede ser algo aislado, debe permitir la comunicación hacia el exterior y desde el exterior para avisar de los acontecimientos que sucedan en la casa como para poder controlar las funciones en nuestra ausencia.

Para la trasmisión de la información, interconexión y control entre los elementos del sistema domótico se puede usar cableado propio, el medio más común o cableado compartido haciendo uso de redes existentes o inalámbricas con tecnologías de radiofrecuencia o infrarrojo etc. [11] [9]

3.4 Inconvenientes

No cabe ninguna duda de que los beneficios y la comodidad y el avance que supone la domótica en casa son evidentes. Pero es necesario comentar los riesgos que ésta conlleva. [12]

- ❖ La piratería informática o el robo de contraseñas: un hacker es capaz de acceder a múltiples sistemas de seguridad, por lo que le permite, anticiparse, dominar y apagar el sistema antes de efectuar un robo.
- ❖ Una gestión deficiente: a veces los usuarios no son capaces de controlar que los sistemas han sido desactivados por un intruso, por lo que el acceso es muy vulnerable.

- El riesgo de cortes en la red: cualquier corte o pérdida de conectividad puede dejar al hogar desprotegido.

- Mal funcionamiento en el dispositivo: Las casas inteligentes, a menudo gestionadas a través de móvil como única alternativa de comunicación, en cuanto falla el dispositivo, puede hacer que el sistema se altere y no conecte.

3.5 Tipos de sistemas domóticos

3.5.1 Centralizado

El sistema de control centralizado es un cerebro electrónico encargado de recoger toda la información proporcionada por los sensores distribuidos en los distintos puntos de control de la vivienda, procesarla, y generar las órdenes que ejecutarán los actuadores.

En dicho sistema el usuario puede programar y controlar todos los sensores y actuadores de su hogar.

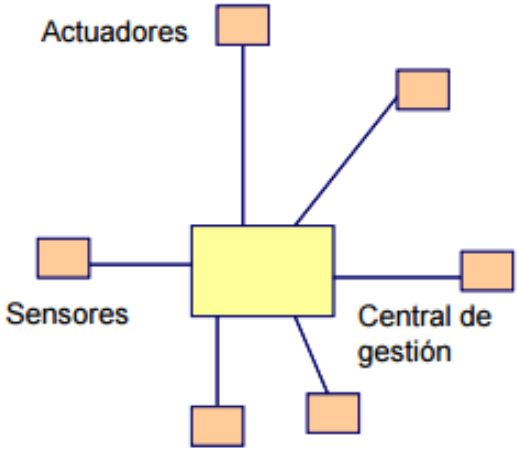


Figura 50: Sistema Centralizado

3.5.2 Descentralizado:

En el sistema domótico descentralizado existe más de un controlador y todos ellos conectados mediante un BUS que se encarga de enviar toda la información entre ellos. Actúan como varios sistemas centralizados en el que cada uno de los controladores se encarga de controlar a los actuadores dependiendo de lo que hayan registrado los sensores y usuario.

Entre sus ventajas destaca la posibilidad de hacer un rediseño de la red, tienen un reducido cableado, se puede ampliar fácilmente y ofrece una gran seguridad de funcionamiento.

Entre los puntos en contra se encuentran la no universalidad de sus elementos de red que conllevan a cierta limitación, el requerimiento de programación compleja y la necesidad de una interfaz de usuario.

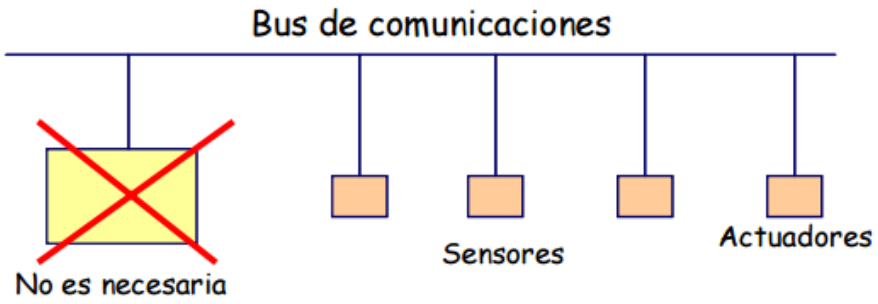


Figura 51: Sistema Descentralizado

3.5.3 Distribuidos

En los sistemas distribuidos, a diferencia de los dos anteriores, su arquitectura es bastante diferente, cada actuador y cada sensor funciona como un controlador que tiene la capacidad de actuar y enviar información al sistema según lo que se recibe de otros dispositivos, lo que significa que cada uno de los dispositivos dentro del sistema cuenta con inteligencia propia y se puede controlar mediante diferentes actividades

Las ventajas de un dispositivo distribuido son su seguridad de funcionamiento, que permiten un profundo rediseño de la red y ampliaciones, sus productos son muy fiables y su coste y cableado no es tan grande. El único inconveniente en este caso es que requiere bastante programación. [10] [13]

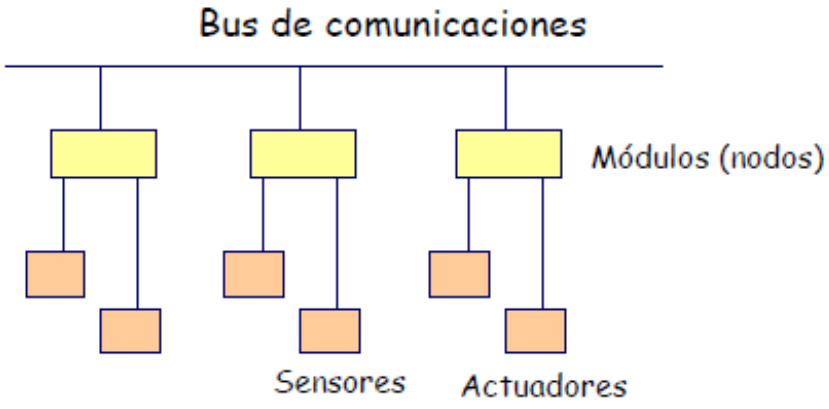


Figura 52: Sistema Distribuido

3.6 Elementos domóticos

3.6.1 La pasarela residencial

Se trata del dispositivo principal que interconecta los distintos dispositivos del edificio que serán utilizados para la instalación del sistema automatizado, haciendo de interfaz común a todos ellos.

3.6.2 Sensores

Son los elementos que recogen la información del entorno (temperatura, humedad, cantidad de luz, presencia de un escape de agua,...) y las envían al sistema de control para que actúe en consecuencia. En algunos casos, los sensores se pueden comunicar directamente con los actuadores. Cabe destacar que no se suelen conectar a la corriente eléctrica, lo cual supone flexibilidad en la instalación.

Algunos de los sensores más comunes son: sensor de temperatura (ver figura 54), detector de gas, detector de humos (ver figura 55), sonda de humedad, sensores de presencia (volumétricos y perimetrales) como se muestra en la figura 53, sensor de iluminación ambiente…etc.

Figura 53: Sensor de Presencia [14]

Figura 54: Sensor de Temperatura [15]

Figura 55: Sensor Detector de humo [16]

3.6.3 Actuadores

Los dispositivos utilizados por el sistema de control para modificar el estado de ciertos equipos o instalaciones son los denominados actuadores. Los hay de diversos tipos como pueden ser el aumento o la disminución de la calefacción o el aire acondicionado, el corte del suministro de gas o agua, el envío de una alarma a una centralita de seguridad, etc.

Los sensores y actuadores pueden estar integrados en el mismo dispositivo en algunos casos.

Entre los más comúnmente utilizados están: los contactores, las electroválvulas de corte de suministro, las válvulas para la zonificación de la calefacción por agua caliente, sirenas o elementos zumbadores para el aviso de alarmas en curso, motores tubulares, bombas y luminarias (ver figura 56) y motores de toldos (ver figura 57). [17]

Figura 56: Actuador: Lámpara

Figura 57: Actuador: Motor de un toldo

3.6.4 Interfaz de usuario

El usuario puede interactuar con el sistema domótico de diversas maneras: [18]

- Interfaz local: La centralita incorpora una pantalla y un teclado.
- Interfaz de voz: Permite programar o conocer el estado del edificio desde cualquier teléfono.
- Interfaz de mensajes móviles: Utilizan la red GSM. Si se produce una incidencia, envía un SMS o MMS.
- Interfaz web: El sistema dispone de un servidor web que permite configurar o conocer el estado actual de una forma gráfica (HTTP).

3.7 Aplicación

3.7.1 IFTTT if this then that

IFTTT es una de las aplicaciones más útiles que existen hoy en día en la plataforma de internet. Esta aplicación tiene una gran ventaja ya que meros usuarios de internet son capaces de usarla. Se trata de una plataforma capaz de conectar y hacer compatibles servicios y objetos que no lo son de fábrica. [15]

IFTTT proviene de las abreviaciones de If This Then That del inglés, que se puede traducir como si esto ocurre, entonces haz lo otro. Nos permite programar tareas que un usuario normal realiza sin necesidad de automatizar pero que con su automatización nos permite una mejor calidad de vida y confort. Un ejemplo puede ser si recibo un correo con el tiempo, programar el aire acondicionado.

En un principio, esta plataforma solo dejaba crear interacciones en objetos y servicios reconocidos por la plataforma. Es decir que cada elemento o servicio que quisiéramos conectar debía tener su canal en IFTTT.

Pero más tarde esto fue evolucionando y la aplicación creo un canal universal que puede ser usado por todos y haría la función de cualquier canal que se necesite, éste canal es capaz de mandar o recibir llamadas http. Este canal se denomina Maker. Permite conectar IFTTT a proyectos personales realizados en Arduino o Raspberry pi.

Este canal Maker abre un mundo de posibilidades a la comunidad. Cualquier cosa que sea programable ya está conectada por IFTTT. Es muy útil para hacer uso de la aplicación IFTTT en otros dispositivos como Raspberry.

Esta plataforma ha introducido recientemente un enlace con Android para poder utilizar nuestro Smartphone en la interconexión de servicios. Han desarrollado su aplicación oficial para Android. Con ella se obtiene acceso a varios triggers o acciones especiales para los dispositivos Android.

Cabe destacar un nuevo concepto muy relacionado con todo esto y que poco a poco se está expandiendo, el denominado “internet de las cosas” en ingles IoT (Internet Of Things). No se trata de lo que está compuesto el internet sino todo lo contrario de todas las cosas que contienen acceso a internet. Este concepto trata de ser para el usuario un entorno transparente donde todo esté conectado útilmente y aprovechable. [19]

Un ejemplo muy cercano y donde aplicaremos en este trabajo es en el hogar. Existen numerosos aparatos o electrodomésticos que se encuentran conectados a internet y que por ello somos capaz de conectar con otro objetos y poder realizar la automatización de un hogar. También hay señores, cada día de más bajo coste, que se utilizan para medir parámetros externos como la intensidad de luz, la temperatura o para la detección de humos. Dichas mediciones son procesadas por un controlador que posteriormente tomará alguna decisión.

Notoriamente aparece un problema en este ámbito, la seguridad. Debido a esa interconexión entre todos los aparatos por medio de la red a Smartphone, tabletas, ordenadores, etc. Llegará un momento en el que el hacker comience a indagar para molestar para hacerse con el control de “nuestras cosas”. Es necesario avanzar mucho en este tipo de seguridad para que tener una casa inteligente sea una realidad. [20]

Esta aplicación se usa por medio de recetas. Estas recetas están formadas por el trigger y la acción formando en su totalidad la condición. Además existen los ingredientes que son “subcondiciones” para personalizar aún más las recetas, con variables o textos, y los canales que son los sitios que pueden ser interconectados en la condición.

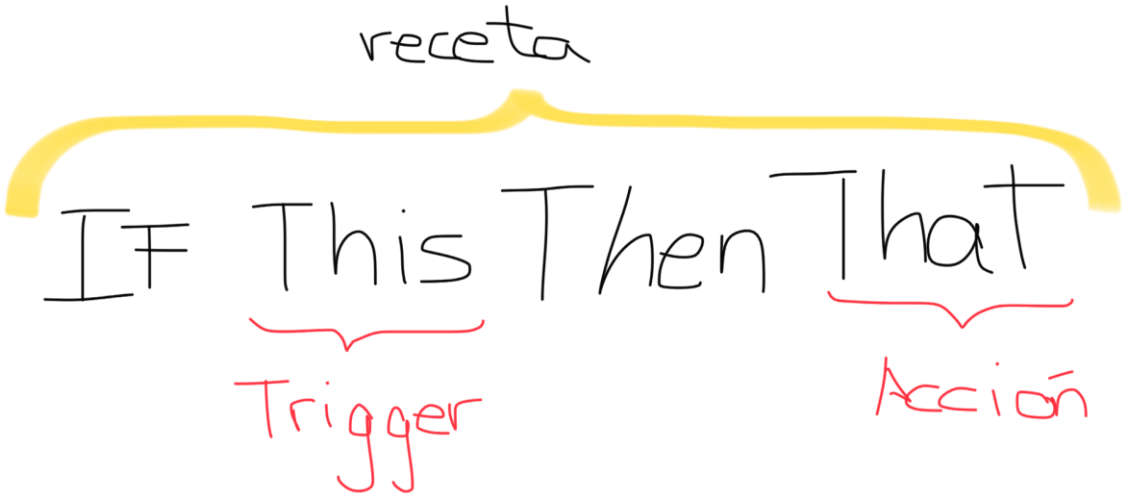


Figura 58: Esquema IFTTT

3.7.2 IFTTT y Aisoy

Para comenzar con la aplicación de Aisoy es necesario implementar una librería de Python.

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Como se ha mencionado en el apartado de Python para acceder a la Raspberry es necesario un intérprete de órdenes, putty.

Una vez dentro, el siguiente paso es instalar la librería requests para simplificar el trabajo del protocolo http y proporciona una lectura de código más clara y permite reducir el número de sentencias. [21] [22]

```
apt-get install Python-requests
```

Una vez instalada esta librería ya es posible usar la herramienta IFTTT con Python.

¿Cómo confeccionar una receta?

1. Crear una cuenta IFTTT en la página www.ifttt.com
2. Acceder a la cuenta
3. Hacer clic en My recipes; create new recipe
4. Aparecerá la pantalla mostrada en la figura 59:

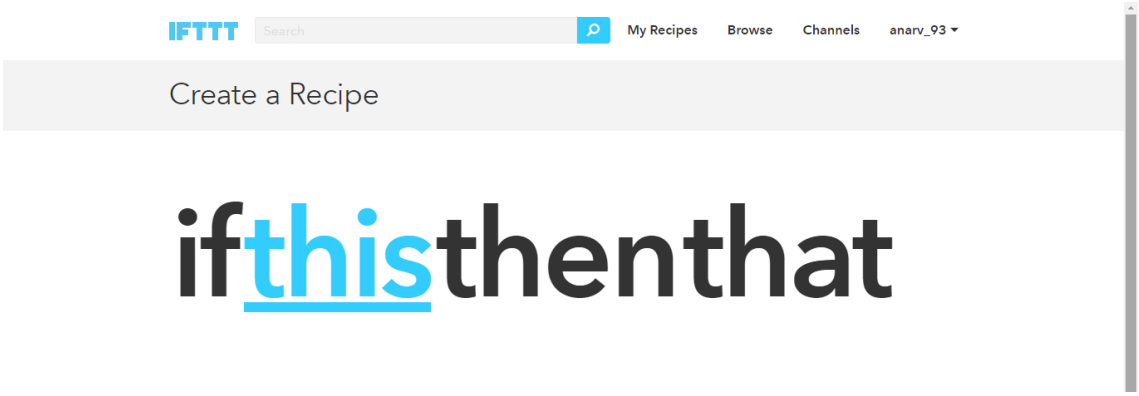


Figura 59: Crear una receta

5. Hacer clic en this para escoger la aplicación destinada a la condición, el "trigger". Existen gran cantidad de canales. Véase la figura 60.

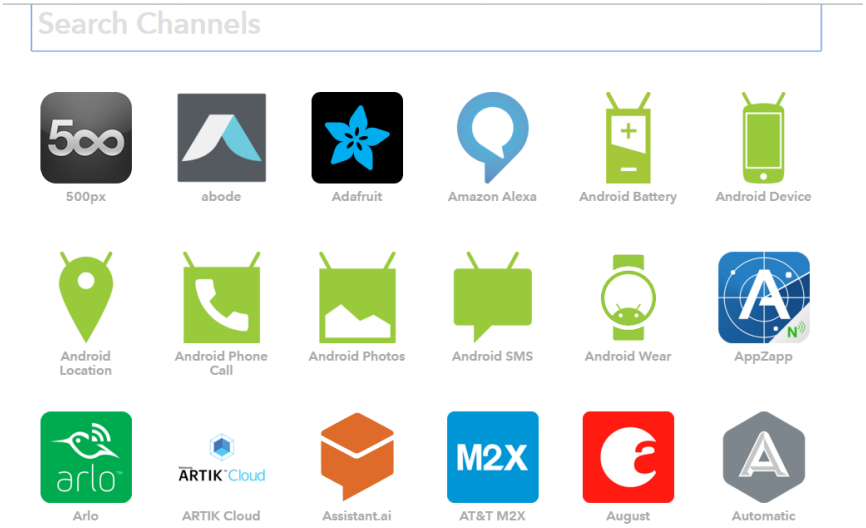


Figura 60: Canales de IFTTT

Sin embargo en este caso se ha escogido la aplicación Maker, que sirve para cualquier servicio. Se hace una búsqueda de dicho canal como se muestra en la figura 61:

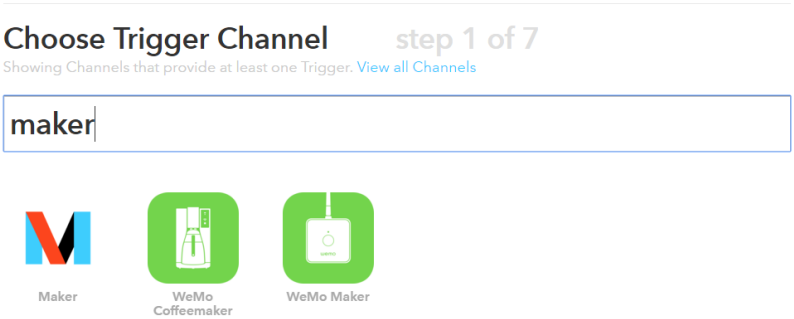


Figura 61: Búsqueda del canal

La aplicación puede tener varias opciones de trigger. En este caso solo tiene una, pero otras aplicaciones como las redes sociales poseen más, por ejemplo publicar un tuit, cambiar el estado… etc. Véase Figura 62.

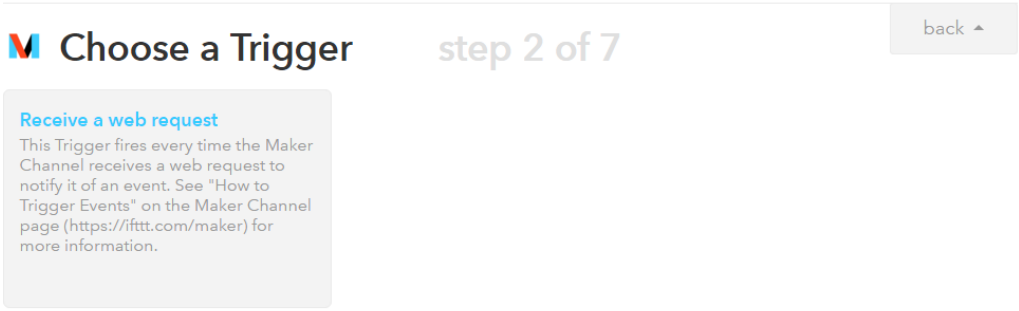


Figura 62: Selección del trigger

6. Poner un nombre al trigger. La interfaz que nos aparecerá será como la imagen de la figura 63 mostrada a continuación.

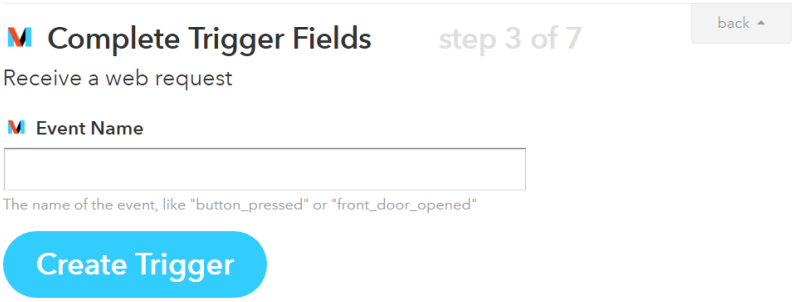


Figura 63: Campos del trigger

Una vez esté configurado el trigger, es posible testearlo. Para ello accedemos a la página www.ifttt.com/maker en dicho enlace y con nuestro usuario activo, hacer clic en “How to trigger event” y aparecerá la pantalla mostrada en la figura 64:

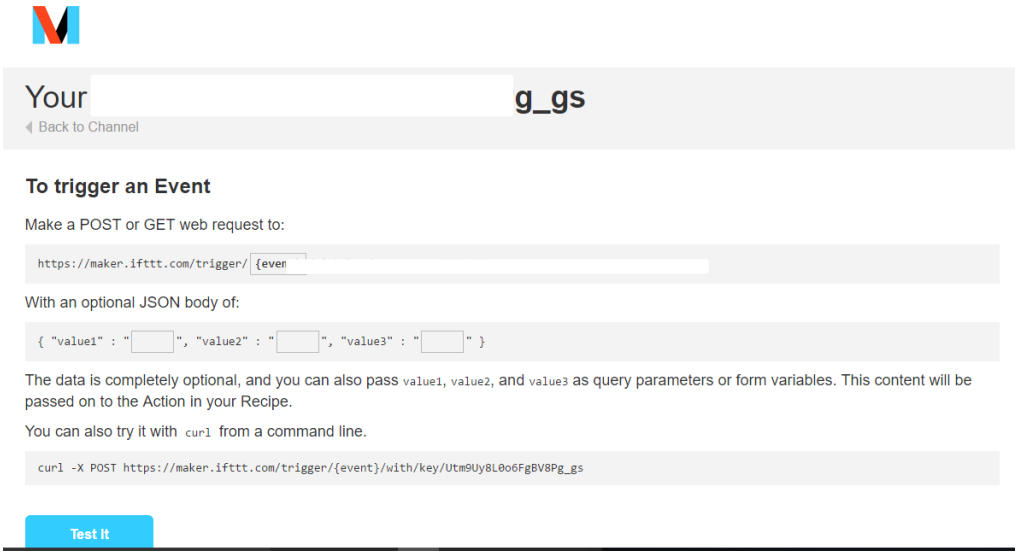


Figura 64: Testear trigger

Al rellenar los campos y hacer clic en Test it, se generará el trigger de la receta y se llevará a cabo la acción que le hallamos asignado.

7. Una vez realizados los pasos anteriores, se ha terminado con la configuración del trigger y aparecerá pantalla que se muestra en la figura 65:

Figura 65: Construcción de la receta

Ya tenemos el trigger configurado como evento de Aisoy. Ahora es necesario configurar la acción.

8. Hacer clic en that y escoger la aplicación que se desee.

En este caso se ha escogido la aplicación Email. Véase la figura 66.

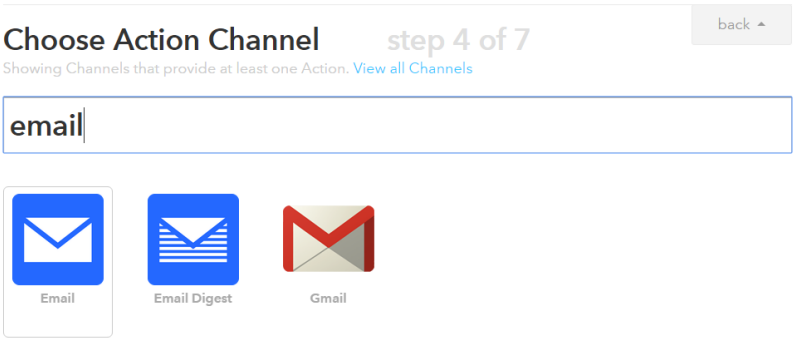


Figura 66: Selección del canal de acción

9. De nuevo es necesario configurar el email. La única acción disponible es enviar un email como se puede ver en la figura 67.

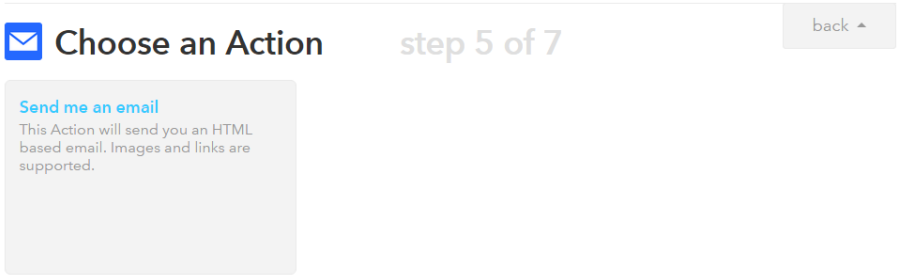


Figura 67: Selección de la acción

10. Completar los campos. Véase Figura 68.

Figura 68: Campos de la acción

En Body: *What* es el nombre que se quiera dar al evento; *when,* la fecha del evento; y *extra data*, el contenido que deseamos que nos llegue por correo.

3.7.2.1 Ejemplo 1

El siguiente código (figura 69) enviará un correo a anarv_93 con los siguientes datos:

a: si se ha tocado el sensor derecho o izquierdo.

b: un valor que pide que escribas por pantalla.

c: un reconocimiento de voz: Enciende la luz.

```
pi@aisoy1: ~
  GNU nano 2.2.6                                File: alerta2.py

from airos5sdk.asr import *
from airos5sdk.tts import *
from airos5sdk.touch import *
import requests
def email_alert(first, second, third):
        report= {}
        report["value1"] = first
        report["value2"] = second
        report["value3"] = third
        requests.post("https://maker.ifttt.com/trigger/sensor_tocado/with/key/Utm9Uy8LOo6FgBV8Pg_gs", data=report)

asr = Asr()
tts = Tts()
touch = Touch()
asr.set_language('es')
tts.set_language('es')
asr.set_grammar("Enciende la luz|buenos dias")
tts.say( "tocame el lado derecho o izquierdo")
sensor = touch.which_touched()
if sensor == 'belly_right' or sensor == 'back_right':
        a= "sensor derecho tochado"
else:
        if sensor == 'belly_left' or sensor == ' back_left':
                a = "sensor izquierdo tocado"

print("introduce algo por pantalla")
b = input()
tts.say("dime enciende la luz")
c = asr.listen(10)
email_alert(a, b, c)
```

Figura 69: Código Ejemplo 1

Compilamos el script con Python alerta2.py

Y el resultado tras ser compilado es el correo mostrado en la figura 70.

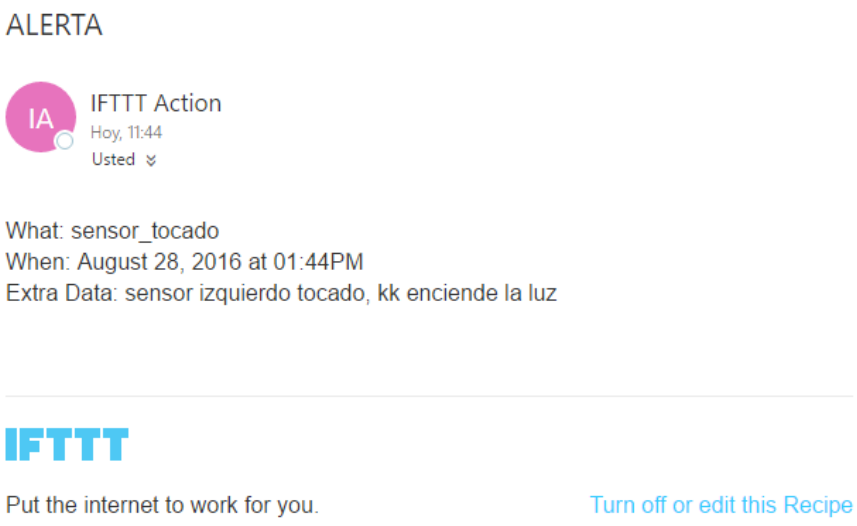


Figura 70: Correo IFTTT

3.7.2.2 Ejemplo 2

En este ejemplo se va a usar de nuevo la aplicación Maker como trigger, sin embargo para la acción se usará twitter. Este código (ver figura 71) consiste en lo siguiente: el robot pedirá que le toquen alguna parte del cuerpo y Twitter publicar un tuit con la información del sensor que ha sido tocado.

pi@aisoy1: ~

GNU nano 2.2.6 File: twiteralerta.py

```
from airos5sdk.asr import *
from airos5sdk.tts import *
from airos5sdk.touch import *
import requests
def twiter_alert(first):
        report= {}
        report["value1"] = first
        requests.post("https://maker.ifttt.com/trigger/Aisoy_event/with/key/Utm9Uy8L0o6FgBV8Pg_gs", data=report)

asr = Asr()
tts = Tts()
touch = Touch()
asr.set_language('es')
tts.set_language('es')
asr.set_grammar("Enciende la luz|buenos dias")
tts.say("tocame alguna parte de mi cuerpo")
sensor = touch.which_touched(10)
print("Lado tocado")
if sensor == 'belly_right' or sensor == 'back_right':
        a= "el lado derecho"
        print("Lado derecho tocado")

else:
        if sensor == 'belly_left' or sensor =='back_left':
                a=" el lado izquierdo"
                print("Lado izquierdo tocado")

        else:
                if sensor == 'head':
                        a="la cabeza"
                        print("Ladocabeza tocado")

twiter_alert(a)
```

Figura 71: Código Ejemplo 2

La respuesta de IFTTT mediante la aplicación Twitter se puede ver en la figura 72.

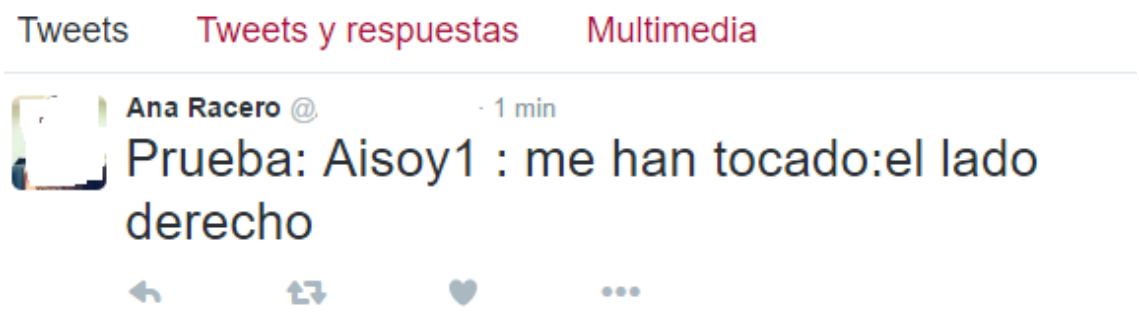


Figura 72: Tuit IFTTT Aisoy

3.7.3 WeMo Bombillas LED inteligentes

Esta parte del trabajo se ha llevado a cabo con integración en domótica del robot Aisoy.

Para ello se ha hecho uso de la aplicación web IFTTT explicada anteriormente y pack de LED bombillas de la marca WeMo. WeMo tiene un canal en IFTTT para poder realizar recetas.

Este producto que se ha escogido para esta aplicación es el kit básico de iluminación LED Belkin WeMo que son un conjunto de dos bombillas inteligentes y el dispositivo que sirve de nexo de unión entre las bombillas y la aplicación de móvil.

Se ha utilizado ésta marca por su precio económico, pero existen otras bombillas como las de la marca Philips que también tienen un canal en la aplicación web IFTTT. Pero el verdadero punto de interés, por la que se justifica su elevado precio pese a ser el más económico, es el elemento inteligente del diseño de la bombilla. Al igual que los demás productos WeMo, las Bombillas LED inteligentes están vinculadas a la red local y a Internet a través del módulo WeMo Enlace.

Belkin es una empresa privada, fue fundada en 1983 y tiene más de 1200 empleados en 21 países. Belkin tiene sede en Playa Vista, California.

Belkin WEMO es una familia de productos simples y personalizados que le permiten controlar los aparatos electrodomésticos desde cualquier lugar dentro e incluso fuera de su red doméstica.

Otros productos WEMO incluyen el Interruptor WEMO Insight, interruptor WEMO y el kit WEMO Interruptor + Motion, todos ellos habilitados para operar con tecnología Wifi con la WEMO App gratuita para dispositivos Android y iOS. A través de la WEMO App es posible controlar múltiples dispositivos electrónicos, dándole una visibilidad inmediata de cada dispositivo conectado.

Las bombillas WeMo Smart LED proporcionan una luz cálida y brillante similar a una tradicional incandescente de 60 vatios, pero con una diferencia importante: sólo consumen 10 vatios de energía y producen muy poco calor. De esta forma, con estas bombillas se puede reducir la cantidad de energía que consumida, y por consiguiente, la factura de la luz.

Según el fabricante este kit de WeMo tiene una vida útil de 23 años basado en un uso promedio diario de 3 horas al día. Con estas bombillas también es posible regular la intensidad luminosa de las mismas directamente desde tu Smartphone. Este kit es posible controlarlo desde un Smartphone, pero también se puede hacer desde IFTTT, que será el objeto de este proyecto para poder comunicarlo con el robot Aisoy1.

Este kit tiene una gran ventaja para cuando estamos fuera de casa ya que podemos simular la apariencia de ocupación en la casa cuando en realidad no hay nadie. Todos los dispositivos como explicaremos posteriormente pueden ser controlados para encenderlos y apagarlos automáticamente. Además, entre otras cosas, permite crear un programa de funcionamiento diario para tu iluminación doméstica. Puedes programar una de tus luces para que se encienda automáticamente al atardecer, reduzca su intensidad cuando ves una película o se apague cuando sales de casa, todo ello desde un Smartphone o Tablet.

Las tecnologías en general, pero sobre todo aquellas que disponen de conexión a internet, son vulnerables y necesitan una protección adecuada. La seguridad d estos dispositivos domótica es algo realmente importante. Detectaron un fallo en estos dispositivos WeMo en enero pero ha sido solucionado. Para tener estos fallos soluciones se debe actualizar la aplicación WeMo.

Las correcciones publicadas por Belkin incluyen:

- Una actualización para el servidor WeMo API del 5 noviembre de 2013, que le impide a un ataque

de inyección XML acceder a otros dispositivos WeMo.

- Una actualización del firmware de WeMo, publicada el 24 de enero de 2014, que agrega la criptografía SSL y la validación de la distribución del firmware de WeMo, elimina el almacenamiento de la clave de seguridad o clave de firma del dispositivo, y la contraseña protege la interfaz del puerto serial para prevenir un ataque malicioso del firmware.

- Una actualización de la aplicación WeMo tanto para iOS (publicada el 24 de enero de 2014) como para Android™ (publicada el 10 de febrero de 2014) que activa la actualización más reciente del firmware.

Primero es necesaria la instalación básica del kit, luego se procederá a hacer la receta en IFTTT y la conexión con el robot.

La instalación es realmente sencilla. [23]

Paso 1:

Enchufar el conector WeMo en un lugar central de la casa. Es necesario tener el menos dos barras de intensidad en dicha localización.

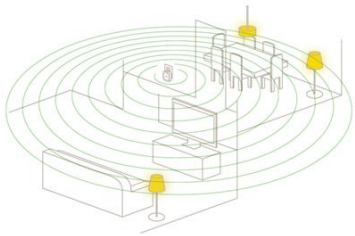


Figura 73: Localización conector WeMo

Paso 2:

Enroscar las bombillas LED inteligentes WeMo y encender el interruptor de la luz.

A continuación se configuran de las bombillas LED inteligentes WeMo.

Paso 3:

Conectar tu dispositivo móvil o Tablet al punto wifi creado por el conector WEMO. La red a la que se debe conectar el dispositivo tiene el nombre WeMo.Link. xxx. Las xxx representan los últimos tres (3) dígitos del número de serie del Conector WEMO. Hay que asegurarse que el dispositivo tenga una conexión activa a internet. Las xxx representan los últimos tres dígitos del número de serie del Conector WEMO.

Figura 74: Paso 3 de configuración de las bombillas

Paso 4:

Iniciar la App WeMo. Puede ser descargada gratuitamente desde Apple® App StoreSM para dispositivos iOS o Google Play™ Store para dispositivos Android™. La aplicación te permite crear una programación

personalizada para encender y apagarlas a horas específicas o simular aleatoriamente la ocupación en tu vivienda. Ésta es una funcionalidad de seguridad ideal para cuando no estás en casa. La aplicación para el control de las bombillas es posible en varios disositivos a la vez.

Paso 5:

La aplicación tratará de detectar las redes cercanas, aparecerá una pantalla como en la figura 75 a la izquierda. Si se le solicita, seleccione su red inalámbrica doméstica. Una vez que muestra que está conectado, activará el Acceso a distancia mostrando la siguiente pantalla como en la figura 75 a la derecha. Aceptar

Figura 75: Paso 5 Configuración de las bombillas

Paso 6:

Aparecerá la siguiente pantalla de la figura 76 como confirmación de que el Conector WEMO se ha instalado correctamente. Una vez instalado el Conector WEMO, continuar con la configuración de las bombillas LED inteligentes WEMO que se conectarán al Conector WEMO, y no al router inalámbrico. Esperar hasta que termine de detectar las bombillas LED inteligentes WEMO. Una vez encontrada las bombillas pulsar Añadir.

Figura 76 : Paso 6 Configuración de las bombillas

Paso 7:

Aparecerá la pantalla de la figura 77. Podemos cambiar y renombrar las bombillas disponibles para poder determinar que bombilla está conectada. Después pulsar guardar.

Figura 77: Paso 7 configuraciones de las bombillas

Una vez realizados los pasos anteriores ya tenemos instalada nuestra bombillas en la aplicación WeMo. No debe salir la siguiente imagen que se muestra en la figura 78:

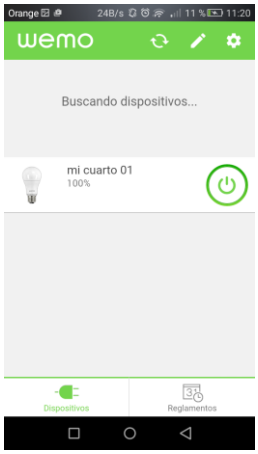


Figura 78: Resultado de configuración de las bombillas

Un aspecto a tener en cuenta muy importante es que el interruptor normal de la bombilla debe estar en modo on para que la aplicación WeMo funcione con normalidad. Si está en modo off desde WeMo no podremos encender la bombilla. Una vez ya tenemos la bombilla configurada, podemos acceder a ella y encender y apagar la bombilla. (Véase Figura 79)

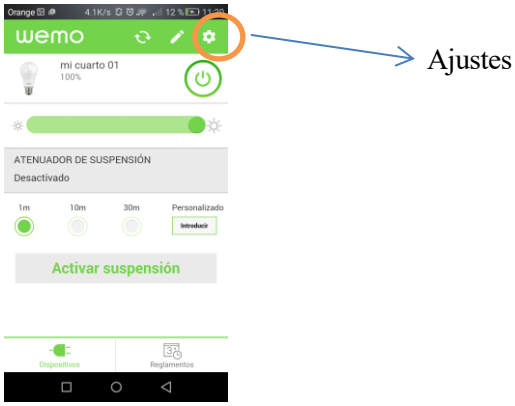


Figura 79: WeMo bombillas

3.7.3.1 Integración con IFTTT

Para la integración de las bombillas con IFTTT es necesario conectar la aplicación con ello. Para ello pulsamos ajustes en la aplicación. Véase figura 76. Buscar y hacer clic en el apartado conectar con IFTTT. Posteriormente hacer clic en connect. Aparecerá una pantalla con los distintos canales que WeMo ofrece con todos sus dispositivos. Buscar y hacer clic WeMo Lighting en nuestro caso. Ya tenemos nuestro dispositivo y nuestras luces conectadas a la plataforma.

Existen multitud de funcionalidades entre esta plataforma IFTTT y WeMo Lighting. A continuación se mostrarán algunas:

- ❖ Cuando llegue a casa, encender WeMo bombillas
- ❖ Cuando me vaya de casa, apagar WeMo bombillas
- ❖ Si está lloviendo, encender las luces.
- ❖ Si se enciende la luz, mandarme un correo.

Es posible también realizar varias recetas con el robot Aisoy1, tantas como acciones sean posibles en el canal de WeMo. Como se explicó anteriormente, cada canal se puede coger como trigger o como acción.

Como trigger tenemos los siguientes ítems:

- Light switched on
    - Trigger campos: Seleccionar la bombilla
- Light switched off
    - Trigger campos: Seleccionar la bombilla
- Light dimmed to specific level
    - Trigger campos: Seleccionar la bombilla y la intensidad

Y como acción:

- Turn on: Encender
    - Actions campos: Seleccionar la bombilla
- Turn off: Apagar
    - Actions campos: Seleccionar la bombilla
- Dim the Light: Atenuar la luz

    - Actions campos: Seleccionar la bombilla y la intensidad

- Turn on a group of lights: Encender un grupo
    - Actions campos: Seleccionar el conjunto de bombilla
- Turn off a group of lights: Apagar
    - Actions campos: Seleccionar el conjunto de bombilla
- Dim a group of lights:
    - Actions campos: Seleccionar el conjunto de bombilla y la intensidad
- Start sleep fader : Iniciar atenuador de sueño
    - Actions campos: Seleccionar el conjunto de bombilla y el tiempo de atenuación en min
- Set sleep fader for a group of lights
    - Actions campos: Seleccionar el conjunto de bombilla y el tiempo de atenuación en min

Ahora se procederá a realizar varias receta con el robot Aisoy1: If Maker then WeMo lighting. En este caso el trigger es el Maker y la acción WeMo lighting. Para realizar el trigger al igual que se ha explicado en los ejemplos anteriores se hace uso de la librería resquest y se envía a una url asociada al trigger.

Cada receta será diferente dependiendo si vamos a querer: apagar la luz, encender la luz o atenuar la luz. Como se ha explicado anteriormente una vez escogido el trigger o la acción es necesario escoger un ítem. (Véase figura 80)

Figura 80: Recetas Domóticas

En la figura 81 se muestra el programa realizado en Python para interconectar con el robot con IFTTT y WeMo.

```
from airos5sdk.asr import *
from airos5sdk.tts import *
from airos5sdk.touch import *
import requests

asr = Asr()
tts = Tts()
touch = Touch()
asr.set_language('es')
tts.set_language('es')
asr.set_grammar("enciende la luz|apaga la luz|atenuar la luz")
print("vocabulario guardado")
i=1
while i==1:
    resp = asr.listen(10)
print "la palabra es:" +resp
if resp == '' :
    tts.say ("no he reconocido nada, intenalo de nuevo ")
else:
    if resp == 'enciende la luz':
        tts.say("encendiendo la luz")
        requests.post("https://maker.ifttt.com/trigger/encender_luz/with/key/Utm9Uy8L0o6FgBV8Pg_gs")
        i=2
    else:
        if  resp == 'apaga la luz':
            tts.say("apagando la luz")
            requests.post("https://maker.ifttt.com/trigger/apagar_luz/with/key/Utm9Uy8L0o6FgBV8Pg_gs")
            i=2
        else:
                if resp == 'atenua la luz':
                    tts.say("atenunado la luz")
                    requests.post("https://maker.ifttt.com/trigger/atenuar_luz/with/key/Utm9Uy8L0o6FgBV8Pg_gs")
                    i=2
```

Figura 81: Programa domótica

También se puede usar el WeMo como trigger. En este caso la receta que sea utilizado es:

If Wemo lighting Then Email (Véase figura 82)

Figura 82: Receta Domótica

Cuando alguien enciende la bombilla “mi cuarto 01” desde cualquier dispositivo con la app WeMo, se mandará un email a anarv_93@hotmail.com. Dicho email será como el de la figura 83.

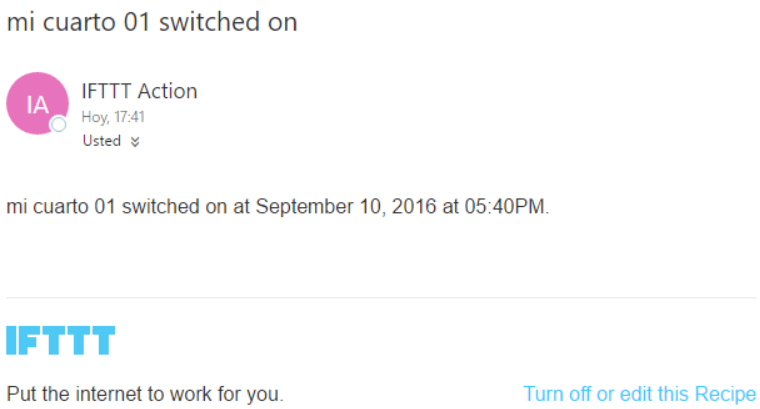


Figura 83: Resultado Receta Domótica

Otra receta más útil es la siguiente: If Android location, then Wemo lighting. (Véase figura 84)

Figura 84: Receta Domótica 2

Si entramos en un área especificada, en este caso un área que redondea nuestra casa, la luz de la bombilla "mi cuarto0 01", se encenderá. Esta área especificada es localizada por el móvil Android, mediante locación GPS y previa conexión del dispositivo con la plataforma. Para ello únicamente es necesario descargar la app IF y realizar los sencillos pasos que va mostrando.

A ROS

Introducción

El sistema operativo del robot (ROS) es un marco flexible para la escritura de software del robot. Es una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear el comportamiento del robot, que es compleja y robusta, a través de una amplia variedad de plataformas robóticas. [24]

ROS tiene dos partes fundamentales: por una parte el sistema operativo, ros, como se describirá a continuación y por otra, ros-pkg, un conjunto de paquetes aportados por la contribución de usuarios, dichos paquetes se encuentran a su vez organizados en conjuntos llamados pilas o en inglés stacks. Las pilas a su vez son un conjunto de procesos que implementan alguna funcionalidad del sistema.

Crear un software de robot de propósito general es verdaderamente difícil. Desde la perspectiva del robot, problemas que parecen triviales a los seres humanos, varían enormemente en ellos, como sus tareas y entornos. Hacer frente a estas variaciones es tan fuerte que ningún individuo, de laboratorio o institución, puede aspirar a hacerlo por su cuenta. Como resultado, ROS fue construido desde cero para fomentar el desarrollo de software de robótica colaborativa. ROS fue diseñado específicamente para que diversos grupos de distintas áreas colaboraran y construyeran sobre el trabajo del otro.

ROS hace uso de un sistema de código abierto, es decir un software libre y gratuito capaz de ser modificado por cualquier persona. Facilita por tanto los servicios propios de un sistema operativo como pueden ser la abstracción del hardware, el control de dispositivo, mensajes de pasos entre procesos, gestión de paquetes... También proporciona herramientas y bibliotecas para la obtención, la construcción, la escritura y la ejecución de código en varios equipos bajo términos de licencia BSD, esta licencia permite libertad para uso comercial e investigador. La librería está orientada para un sistema UNIX (Ubuntu (Linux)) pero también se puede instalar en otros sistemas como Fedora, Mac OS o Microsoft Windows entre otros. Todo ello nos permite simplificar la tarea de crear un comportamiento complejo y robusto de un robot en una amplia variedad de plataformas robóticas.

Historia ROS

ROS es un proyecto grande con muchos antepasados y colaboradores. La necesidad de un marco de colaboración abierta fue apreciada en la comunidad de investigación robótica, y muchos proyectos se han creado para alcanzar este objetivo.

Dichos esfuerzos fueron coordinados en la Universidad de Standford a mediados de la década de 2000, de la mano de la Stanford AI Robot (STAIR) y el programa Personal Robots (PR), que lograron la creación de prototipos de sistemas software dinámicos y flexibles, orientados para usos robóticos. En 2007, Willow Garage, un visionario del mundo de la robótica, proporcionó abundantes recursos para extender todos estos conceptos más allá y dar vida a implementaciones convenientemente testeadas. El esfuerzo se vio impulsado por innumerables investigadores que beneficiaron con su tiempo y experiencia para dar vida a las ideas del núcleo de ROS y sus paquetes de software fundamentales. En todo momento el software fue desarrollado bajo el uso de la licencia de código abierto BSD y poco a poco se ha convertido en una plataforma ampliamente utilizada en la comunidad de investigación robótica. [25]

Desde el principio, fueron muchas entidades las que desarrollaron este sistema. ROS se desarrolló en múltiples instituciones y para diferentes robots, incluyendo muchas de las que recibieron los robots PR2 de Willow Garage.

Figura 85: Robot PR2 de Willow Garage

Este modelo está preparado para que cualquier persona pueda comenzar su investigación con un punto de partida conocido y sea capaz de desarrollar su propio repositorio de código ROS. Posteriormente es elección propio de cada persona decidir que sea de código abierto o no, pudiendo luego ser reconocido por sus logros.

Versiones de ROS

Es posible encontrarnos una versión de ROS incompatible con otra. Normalmente están referidas por un sobrenombre en vez de por una versión numérica. Las versiones, desde la más actual a la primera versión, son: [26]

- 23/Mayo/2016 – Jase Turtle
- 22/Julio/2014 - Indigo Igloo
- 04/Septiembre/2013 - Hydro Medusa
- 31/Diciembre/2012 - Groovy Galapagos
- 23/Abril/2012 - Fuerte
- 30/Agosto/2011 - Electric Emys
- 02/Marzo/2011 - Diamondback
- 03/Agosto/2010 - C Turtle
- 01/Marzo/2010 - Box Turtle
- 22/Enero/2010 - ROS 1.

Conceptos básicos de ROS

ROS asienta una red peer-to-peer, es decir, una red que conecta un gran número de ordenadores (nodos) para compartir cualquier cosa que esté en formato digital. Los conceptos básicos son: [27]

Nodo (node):

Un nodo es un proceso que realiza algún tipo de computación en el sistema. Los nodos se combinan dentro de un grafo, compartiendo información entre ellos, para crear ejecuciones complejas. Un nodo, que hace la función de un ejecutable dentro de un paquete ROS, puede controlar un sensor láser, otro los motores de un robot y otro la construcción de mapas.

Mensajes (Messages)

Son estructuras de datos simples que se pasan entre los dos. Es el contenido de los *Topics*. Existen tipos primitivos estándar como Integer, Floatins point o Boolean. Además se pueden crear tipos personalizados.

Tópico (topic)

Los tópicos son nombres que identifican el contenido de un mensaje. Son canales de información entre los nodos. Un nodo puede emitir o suscribirse a un tópico. En general, los publicadores y subscriptores no son conscientes de la existencia de los otros. La idea es desacoplar la producción de información de su consumición. La información es, por tanto, unidireccional (asíncrona). Si lo que queremos es una comunicación síncrona (petición/respuesta) debemos usar servicios.

El *publisher* publica mensajes de un determinado *topic* y el *subscriber* se subscribe a mensajes de un determinado *topic*.

Servicios (services)

Es la arquitectura cliente-servidor entre nodos. Utilizan dos mensajes uno para la solicitud y otro en la respuesta. El nodo servidor se mantiene a la espera de las solicitudes que realiza el nodo cliente, y responde a ella. La librería cliente de ROS generalmente presenta esta interacción como si fuera una llamada a procedimiento remoto.

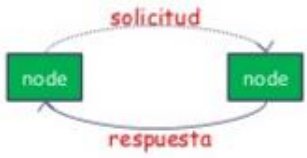


Figura 86: Esquema cliente-servidor

Paquete (package):

El software en ROS está organizado en paquetes. Un paquete puede contener tópicos, un nodo, una librería, conjunto de datos, o cualquier cosa que pueda constituir un módulo. Los paquetes pueden organizarse en pilas (stacks).

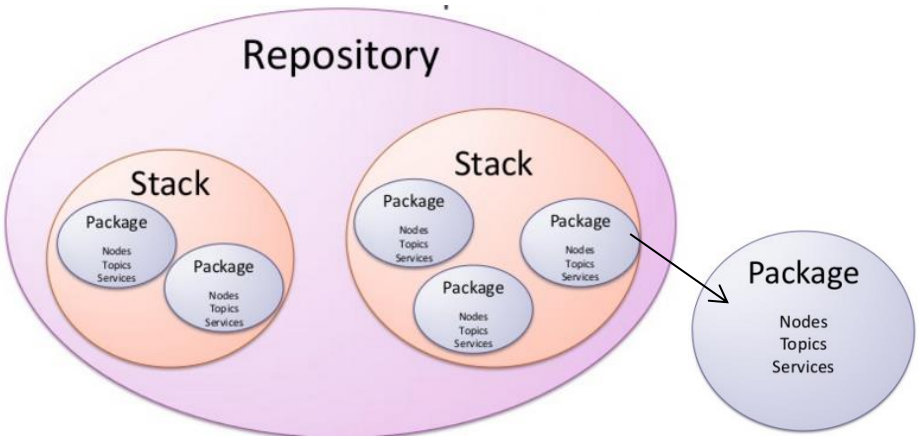


Figura 87: Estructura básica de un repositorio

Pila (stack):

Conjunto de nodos que juntos proporcionan alguna funcionalidad.

Repositorio (Repository)

Son una colección de paquetes y pilas disponibles online.

Bolsas (Bags)

Se trata de archivos para guardar y volver a ejecutar datos de comunicación de ROS. Sirve para memorizar una serie de órdenes y después repetirlas secuencialmente. Además son un mecanismo importante para el almacenamiento de datos, como lecturas de sensores, que pueden ser difícilmente adquiridas pero son necesarias para el desarrollo y testeo de algoritmos.

ROS Master

La principal funcionalidad del Master ("maestro") es permitir a los nodos en ROS localizarse unos a otros. Este nodo maestro registra todos los nodos, servicios y topics que se encuentran en ejecución.

Parameter server

Permite guardar los datos de forma centralizada.

Características Robóticas:

Paso de mensajes

Tras varios años de mejora, se ha desarrollado un conjunto de mensajes estándar para el uso compartido en la robótica. Algunos ejemplos de definiciones de mensajes de conceptos geométricos son poses, transformadas o vectores o para sensores como cámaras. Esto ayuda al ecosistema ROS a aumentar su operatividad con herramientas.

Librerías geométricas de robot

Una lucha constante en la que se encuentran los programadores es la de controlar el posicionamiento de un robot respeto a unas referencias. Sobre todo es importante para el desarrollo de robot humanoides con partes móviles. ROS aborda este problema con la creación de una librería denominada tf, la cual proporciona información sobre donde se encuentra cada parte del cuerpo del robot.

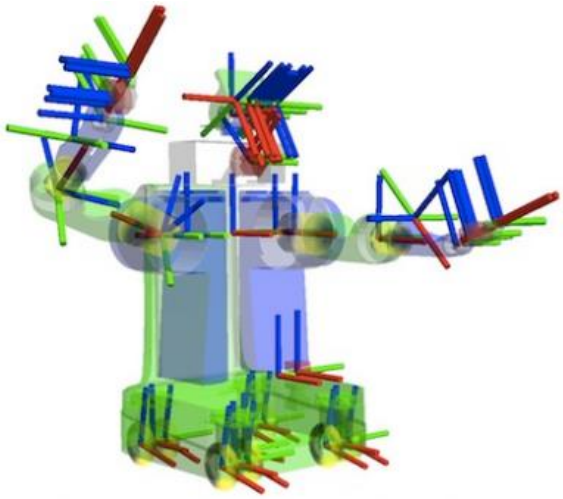


Figura 88: Geometría de Robot en ROS

Existe un conjunto de herramientas para describir y modelar el robot para que pueda ser comprendido por todo el sistema de ROS así como, tf, robot_state_publisher y rviz. Para ello se edita un documento XML, escribiendo en URDF (United Robot Description Format) en el que se especifican todas las dimensiones del robot, desde longitudes, movimientos de articulaciones además de masa e inercia etc.

Una vez definido todo ello, el robot se puede utilizar fácilmente con la librería tf representada en las tres dimensiones con visualizaciones detalladas y los planificadores de movimiento. [28]

Características principales

Comunicación con el entorno

Hoy en día, uno de los aspectos principales a tener en cuenta cuando se implementa una aplicación robótica es la forma de comunicación del robot con el entorno.

Una solución a lo expuesto anteriormente se encuentra el sistema de mensajería integrado en ROS que permite una disminución del tiempo de gestión de la comunicación entre nodos. Es por ello que es posible obtener nuevos datos y reproducirlos en el código sin necesidad de cambio en él.

ROS y las funciones que componen sus librerías tienen la función de serializar y deserializar automáticamente los datos generados por los generadores de códigos de cada lenguaje. Esto produce un ahorro enorme en complejidad de programación disminuyendo notablemente las líneas de código.

Refiriéndonos al desarrollo multilenguaje, podemos afirmar que al existir varios lenguajes de programación los cuales están comprometidos por diversos requerimientos en cuanto a eficiencia de ejecución o facilidad de escritura, la sintaxis ROS nos ofrece soporte para los lenguajes más extendidos en el desarrollo de software para robot como son Python, C++, LISP y Octave.

El middleware ROS proporciona la capacidad de interacciones síncronas de petición-respuesta, además del funcionamiento de su naturaleza asíncrona que hace funcionar efectivamente muchas necesidades de

comunicación robótica.

REFERENCIAS

[1] *Manual de usuario Aisoy1V5.*

[2] «http://www.estamoscreandovida.com/airos-el-sistema-que-permite-a-los-aisoy1-sentir-y-emocionarse/,» [En línea].

[3] [En línea]. Available: https://es.wikipedia.org/wiki/Scratch_(lenguaje_de_programaci%C3%B3n).

[4] «http://www.mclibre.org/consultar/python/otros/in_presentacion.html,» [En línea].

[5] V. J. E. Muñoz, HTML, presente y futuro de la web.

[6] «https://www.codejobs.biz/es/blog/2013/03/03/historia-de-python,» [En línea].

[7] «http://opencv.org/,» [En línea].

[8] «https://uvadoc.uva.es/bitstream/10324/11848/1/TFG-P-165.pdf,» [En línea].

[9] «https://es.wikipedia.org/wiki/Dom%C3%B3tica,» [En línea].

[10] https://www.casadomo.com/articles.aspx?menu=articulos. [En línea].

[11] http://domotica1003.weebly.com/medios-de-transmisionbus.html. [En línea].

[12] http://willisupdate.com/la-tecnologia-domotica-en-casa-a-que-nuevos-riesgos-nos-enfrentamos-en-el-sector-asegurador/. [En línea].

[13] http://hogartec.es/sistemas-domoticos-centralizados-descentralizados-y-distribuidos/. [En línea].

[14] lra.unileon.es. [En línea].

[15] www.domoticadomestica.com. [En línea].

[16] www.arqhys.com. [En línea].

[17] http://es.slideshare.net/aquiles2012/instalaciones-domoticas-41098848. [En línea].

[18] http://ocw.um.es/ingenierias/domotica/material-de-clase-1/tema-1-introduccion-domotica-edificio-inteligente-vocw.pdf. [En línea].

[19] http://fuasocialweb.com/2014/01/ifttt-que-es-y-como-se-usa/. [En línea].

[20] http://www.ticbeat.com/tecnologias/guia-ifttt-que-es-como-funciona-15-recetas-utiles/. [En línea].

[21] http://www.xatakahome.com/trucos-y-bricolaje-smart/the-maker-channel-es-lo-mejor-que-le-ha-pasado-a-ifttt-y-a-la-internet-de-las-cosas. [En línea].

[22] https://github.com. [En línea].

[23] http://www.belkin.com/es/support/article/?lid=es&pid=F5Z0489&aid=17763&scid=1287. [En línea].

[24] http://www.ros.org/about-ros/. [En línea].

[25] http://erlerobotics.com/blog/ros-introduction-es/#history. [En línea].

[26] https://en.wikipedia.org/wiki/Robot_Operating_System#Version_History. [En línea].

[27] https://moodle2015-16.ua.es/moodle/mod/book/tool/print/index.php?id=82546. [En línea].

[28] PFCrmerino_report.

[29] [En línea]. Available: https://es.wikipedia.org/wiki/Scratch_(lenguaje_de_programaci%C3%B3n).

[30] https://en.wikipedia.org/wiki/Robot_Operating_System#Version_History. [En línea].

[31] dfcvb. [En línea].