BOTBLOQ: Ecosistema integral para el diseño, fabricación y programación de robots DIY

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI)
EXPEDIENTE: IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

ACRÓNIMO DEL PROYECTO: BOTBLOQ

ENTREGABLE 5.2.2 Informe descriptivo de la herramienta desarrollada para robots en BOTBLOQ

RESUMEN DEL DOCUMENTO

En este documento se documenta el proceso de adaptación de la herramienta desarrollada en el PT05 para su uso con los robots de Botbloq desarrollados durante el PT04.

Índice

1.- Introducción **3**

2.- Proceso de adaptación de las herramientas **8**

2.1.- Modificación del código base para el uso con Bluetooth 8

2.2.- Adaptación del código base de los proyectos python 10

Escribir un texto en la pantalla del dispositivo móvil 11

Emitir un sonido desde el móvil 11

Recibir datos por voz o texto 11

Encender/apagar la linterna del dispositivo 11

Leer luz ambiente 11

Leer si está cubierto el dispositivo 11

Leer aceleracion 11

Leer giroscopio 12

Leer campo magnético 12

Leer variables 12

2.3.- Modificaciones en la generación de código python 12

2.4.- Creación de los bloques 13

2.5.-Adaptación de Bitbloq Connect 13

2.6.- Adaptación de los robots del PT04 13

2.7.- Adaptación de la herramienta de programación de Botbloq 16

3.- Testing **18**

Vehículo: 18

Manipulador: 18

Serpiente: 21

4.- Referencias **22**

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

1.- Introducción

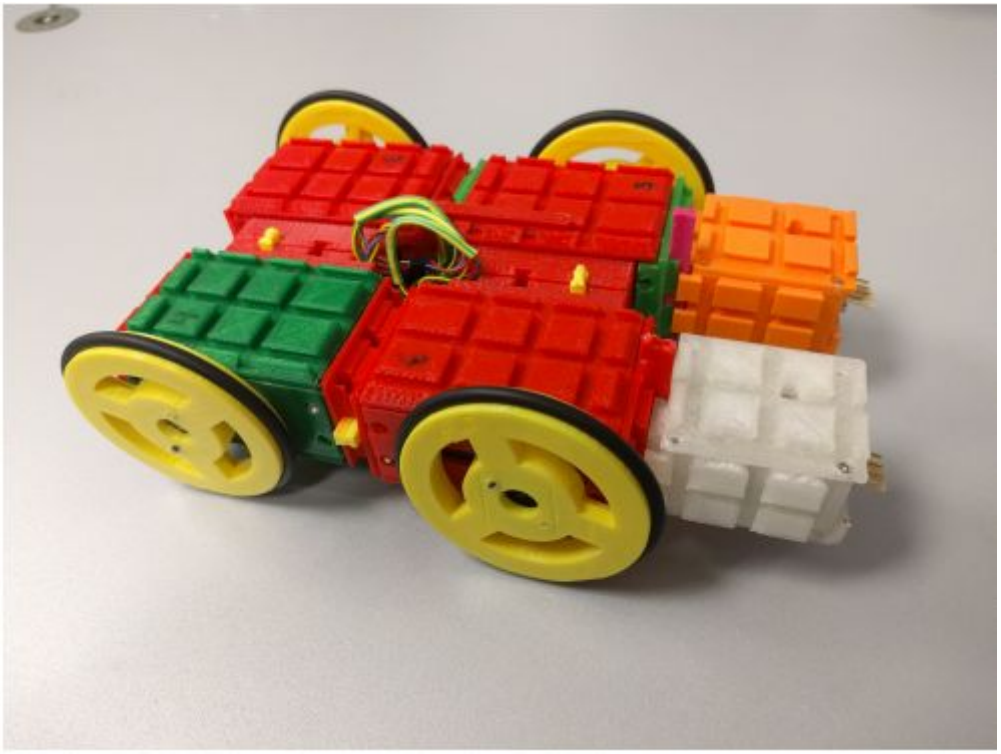
Durante el desarrollo del PT05 en el transcurso del hito 2 se desarrollaron dos herramientas para que los alumnos pudiesen programar usando dispositivos móviles, la aplicación de Android “Bitbloq Connect” y una adaptación de la herramienta desarrollado en el hito 1 en el PT02.

Durante el pasado hito 2 nos centramos en que se pudiera usar un dispositivo móvil como un conjunto de componentes desde una placa microcontroladora, y que se pudieran crear un programa de forma sencilla e intuitiva, usando para eso una interfaz de bloques sencilla. Durante el desarrollo de este hito nos hemos centrado en que los robots de Botbloq creados en el PT04 puedan usar también los dispositivos móviles como un conjunto de componentes.

Los robots del PT04 pueden ser programables mediante ROS usando el lenguaje de programación Python[(1)], o mediante el lenguaje Arduino[(2)], para los robots que usan Arduino se usan las herramientas descritas en el anterior paquete de trabajo ya que es lo mismo que para un kit genérico, así que vamos a cubrir el caso de los robots que se programan con Python.

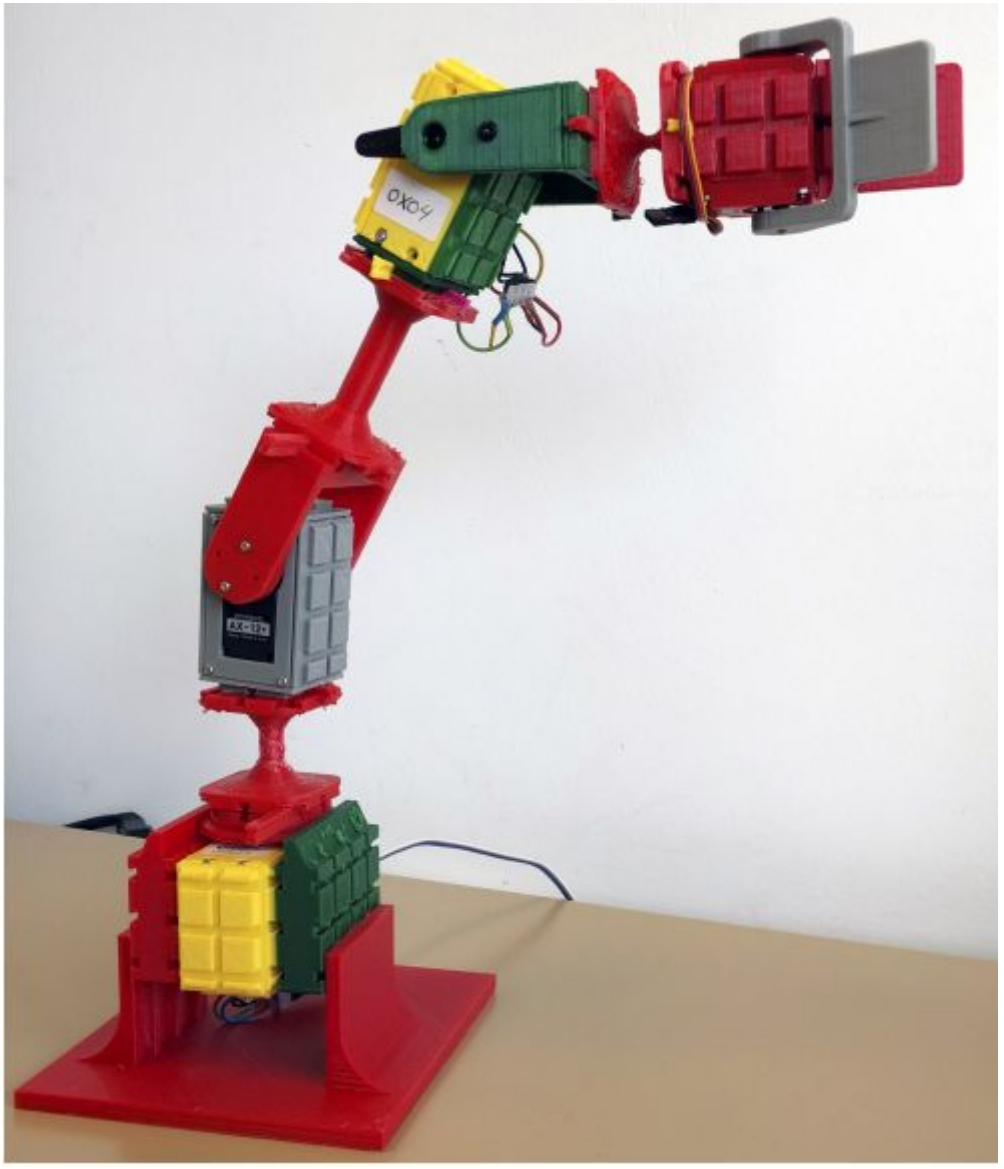
Los robots a programar con dispositivos móviles son:

Vehículo:



Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

Manipulador:

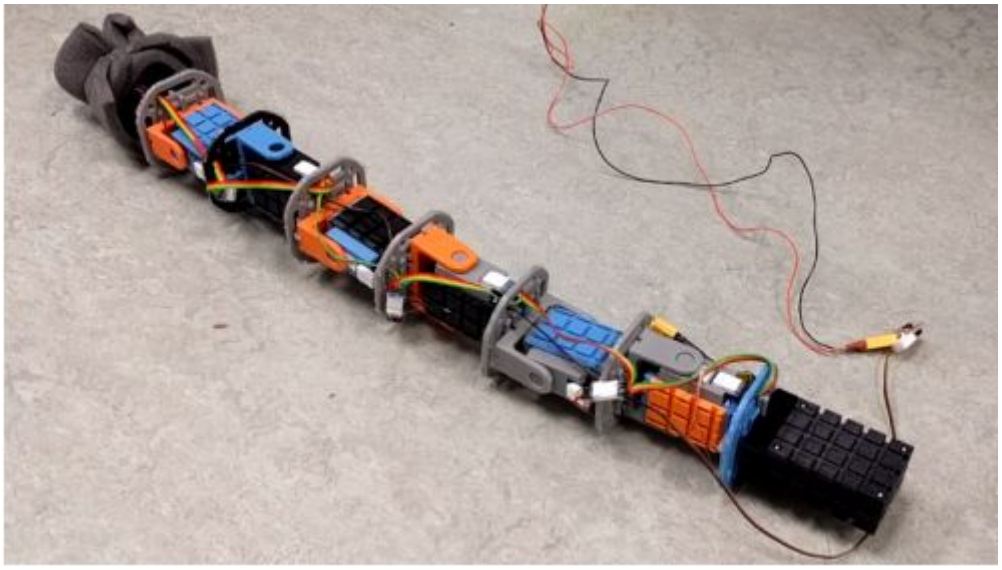


Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

Humanoide:

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

Serpiente:



Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

Hexápodo:

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

2.- Proceso de adaptación de las herramientas

2.1.- Modificación del código base para el uso con Bluetooth

El código base usado en los robots era un fichero con un mínimo contenido necesario para el intérprete de Python en el que se indicaba la codificación '# coding=utf-8', sin embargo se ha modificado para que nada más comenzar el programa se inicie el bluetooth, y se busque una conexión. Para realizar este cambio se crea un perfil:

```
options.uuid = "1101"
options.psm = "3"
options.role = "server"
options.name = "Edison SPP Loopback"
options.service = "spp char loopback"
options.path = "/foo/bar/profile"
options.auto_connect = False
options.record = ""
profile = Profile(bus, options.path)
mainloop = GObject.MainLoop()
opts = {
        "AutoConnect" :    options.auto_connect,
}
if (options.name):
    opts["Name"] = options.name
if (options.role):
    opts["Role"] = options.role
if (options.psm is not None):
    opts["PSM"] = dbus.UInt16(options.psm)
if (options.channel is not None):
    opts["Channel"] = dbus.UInt16(options.channel)
if (options.record):
```


Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

```
    opts["ServiceRecord"] = options.record
if (options.service):
    opts["Service"] = options.service
if not options.uuid:
    options.uuid = str(uuid.uuid4())
manager.RegisterProfile(options.path, options.uuid, opts)
```

y luego en la función que salta cuando hay una nueva conexión se añade el bucle lo que permite al código buscar continuamente la conexión mediante bluetooth.

```
class Profile(dbus.service.Object):
        fd = -1

        @dbus.service.method("org.bluez.Profile1",
                                        in_signature="",
out_signature="")
        def Release(self):
                print("Release")
                mainloop.quit()

        @dbus.service.method("org.bluez.Profile1",
                                        in_signature="",
out_signature="")
        def Cancel(self):
                print("Cancel")

        @dbus.service.method("org.bluez.Profile1",
                                in_signature="oha{sv}", out_signature="")
        def NewConnection(self, path, fd, properties):
                self.fd = fd.take()
                print("NewConnection(%s, %d)" % (path, self.fd))


                server_sock = socket.fromfd(self.fd, socket.AF_UNIX,
socket.SOCK_STREAM)
```


Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

```
                server_sock.setblocking(1)
                #server_sock.send("This is Edison SPP loopback test\nAll
data will be loopback\nPlease start:\n")

                try:
                 #hasta aqui se copia y pega todo todo
                 #aqui se mete el código generado en bitbloq
#desde aqui se copia y pega todo
                except IOError:
                    pass
                server_sock.close()
                print("all done")
#parte final
        @dbus.service.method("org.bluez.Profile1",
                                in_signature="o", out_signature="")
        def RequestDisconnection(self, path):
                print("RequestDisconnection(%s)" % (path))
                if (self.fd > 0):
                        os.close(self.fd)
                        self.fd = -1
```

2.2.- Adaptación del código base de los proyectos python

Se ha modificado el código base para añadir todas las funciones para interactuar con el dispositivo móvil al principio del código, también se ha añadido a todo el código generado por los bloques un bucle, para que esté repitiendose continuamente para emular el comportamiento de lenguaje Arduino[(2)].

Todas las funciones usan la conexión a través de un socket con el dispositivo móvil para enviar o recibir los comandos específicos que realizan las acciones.
Estos comandos fueron descritos en el anterior entregable, por lo que en este entregable nos centraremos en representar las funciones añadidas para los robots de Botbloq.

A continuación se detallan las funciones añadidas para interactuar con el dispositivo móvil:

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

Escribir un texto en la pantalla del dispositivo móvil

```
def escribe_texto(server_sock,texto):
    server_sock.send("%s\n" % texto)
```

Emitir un sonido desde el móvil

```
def emitir_sonido(server_sock, sonido ):
    server_sock.send("playSound-%s\n" % sonido)
    time.sleep(1)
```

Recibir datos por voz o texto

```
def recibe_texto(server_sock):
    data = server_sock.recv(1024)
    return data
```

Encender/apagar la linterna del dispositivo

```
def enciende_linterna(server_sock):
    server_sock.send("turnonFlashlight-\n")


def apaga_linterna(server_sock):
    server_sock.send("turnoffFlashlight-\n")
```

Leer luz ambiente

```
def leer_luz(server_sock):
    server_sock.send("readLight-\n")
    dato = recibe_texto(server_sock)
    print("nivel luz: %s" %dato)
    return dato
```

Leer si está cubierto el dispositivo

```
def recibir_estacubierto(server_sock, cv):
    server_sock.send("readProx-%s\n" % cv) #Fallo
    dato = recibe_texto(server_sock)
    print("cubierto: %s" %dato)
    return dato
```

Leer aceleracion

```
def recibir_aceleracion(server_sock, message, axis):
    server_sock.send("%s%s\n" %(message, axis))
```


Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

```
    dato = recibe_texto(server_sock)
    print("aceleration: %s" %dato)
    return dato
```

Leer giroscopio

```
def recibir_giroscopio(server_sock, axis):
    server_sock.send("readGyros-%s\n" % axis)
    dato = recibe_texto(server_sock)
    print("giroscopio: %s" %dato)
    return dato
```

Leer campo magnético

```
def recibir_campomagnetico(server_sock, axis):
    server_sock.send("readMagnetic-%s\n" % axis)
    dato = recibe_texto(server_sock)
    print("magnetic field: %s" %dato)
    return dato
```

Leer variables

```
def recibir_orientacion(server_sock, variable):
    server_sock.send("readOrientation-%s\n" % variable)
    dato = recibe_texto(server_sock)
    print("variable: %s" %dato)
    return dato
```

2.3.- Modificaciones en la generación de código python

Durante el anterior Hito se desarrolló una librería para generar código python reutilizando los mismo bloques que se usan para la generación de código Arduino[2], de forma que al usuario luego se le hace transparente si está programando Arduino o Python. Es la forma más sencilla de enseñar ya que permite centrarse en la lógica de la programación y no tanto en la sintaxis.
Se han realizado modificaciones en la generación de la plantilla como hemos mencionado anteriormente añadiendo estas nuevas constantes globales :

- DEFAULT_IMPORTS_CODE
- HEADERCODE
- CLASSES_CODE
- FINAL_CODE

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

Ya que para la generación de código es necesario que inyectar código autogenerado entre esos pedazos de código. Por ilustrarlo con un ejemplo, cuando se utiliza un vehículo, la importación de las librerías necesarias tiene que ir justo debajo de DEFAULT_IMPORTS_CODE y antes del HEADERSCODE.

Se ha modificado el comportamiento cuando se genera código de un bloque de tipo output ya que hasta entonces no existían en python.

Y con esto se han modificado todos los bloques del dispositivo para darles soporte a python, y además se ha incluido el bloque de espera, ya que se ha visto necesario para poder crear programas más complejos.

2.4.- Creación de los bloques

La modificación en los bloques se ha realizando rellenando el campo de Python con la misma sintaxis creada durante el hito 2.

No ha hecho falta modificar las reglas creadas durante el Hito 2, se han podido reutilizar la misma estructura, sólo se han añadido las condiciones para la generación del código definido en la parte superior del código base.

2.5.-Adaptación de Bitbloq Connect

No ha sido necesaria modificación alguna de la aplicación Bitbloq Connect, ya que se diseñó para que se pudiera abrir un puerto serie con cualquier otro dispositivo, y recibiera unos comandos y devolviera otros comandos. Por tanto todas las modificaciones realizadas a lo largo de 2017 han sido correcciones de errores esporádicos, pero sin necesidad de cambios para ser usada por los robots de Botbloq. Su documentación se encuentra en el anterior entregable.

2.6.- Adaptación de los robots del PT04

Se ha tenido que activar el bluetooth dentro de las placas Intel Edison usadas en los robots de Botbloq. Para tal fin se ha modificado el fichero de configuración de definición de las políticas del DBUS BlueZ [3] con vi:
root@edison:~# vi /etc/dbus-1/system.d/bluetooth.conf

Añadiendo la línea *<allow send_interface="org.bluez.profile1"/>*

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

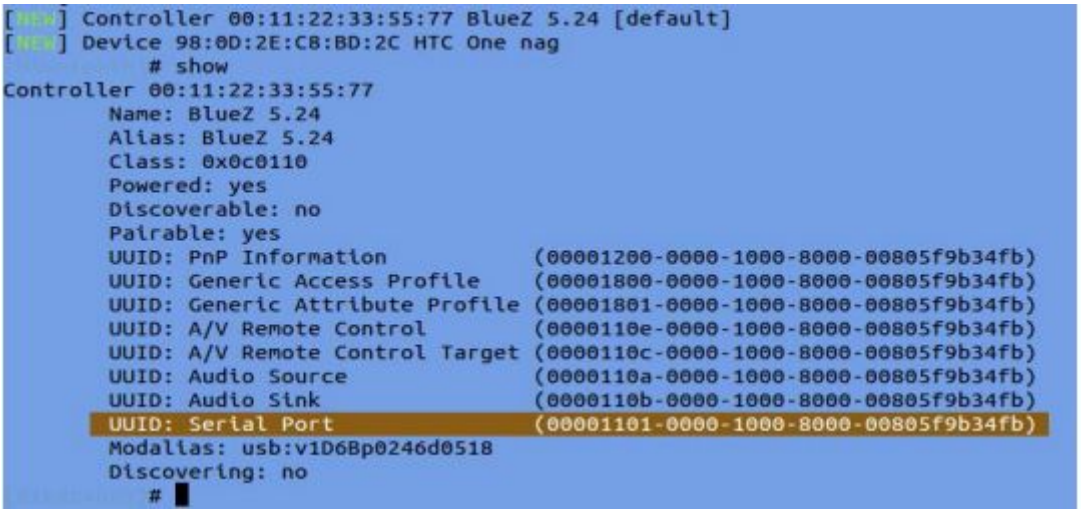
```
ram@ram-desktop: ~
<!-- This configuration file specifies the required security policies
     for Bluetooth core daemon to work. -->

<!DOCTYPE busconfig PUBLIC "-//freedesktop//DTD D-BUS Bus Configuration 1.0//EN"
 "http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>

  <!-- ../system.conf have denied everything, so we just punch some holes -->

  <policy context="default">
    <allow own="org.bluez"/>
    <allow own="org.bluez.obex"/>
    <allow send_destination="org.bluez"/>
    <allow send_destination="org.bluez.obex"/>
    <allow send_interface="org.bluez.Agent1"/>
    <allow send_interface="org.bluez.MediaEndpoint1"/>
    <allow send_interface="org.bluez.Profile1"/>
    <allow send_interface="org.bluez.ThermometerWatcher1"/>
    <allow send_interface="org.bluez.AlertAgent1"/>
    <allow send_interface="org.bluez.HeartRateWatcher1"/>
    <allow send_interface="org.bluez.CyclingSpeedWatcher1"/>
  </policy>

</busconfig>
~
```

Ese es el perfil que luego se genera en el código base descrito en el apartado 2.1. Después se tiene que emparejar, con un dispositivo con Bluetooth, y una vez realizado el emparejamiento se puede probar la conexión. Para probarla se puede ejecutar el comando “bluetoothctl”:

```
root@edison:~# bluetoothctl
[NEW] Controller 00:11:22:33:55:77 BlueZ 5.24 [default]
[NEW] Device 98:0D:2E:C8:BD:2C HTC One nag
          # show
Controller 00:11:22:33:55:77
        Name: BlueZ 5.24
        Alias: BlueZ 5.24
        Class: 0x0c0110
        Powered: yes
        Discoverable: no
        Pairable: yes
        UUID: PnP Information           (00001200-0000-1000-8000-00805f9b34fb)
        UUID: Generic Access Profile    (00001800-0000-1000-8000-00805f9b34fb)
        UUID: Generic Attribute Profile (00001801-0000-1000-8000-00805f9b34fb)
        UUID: A/V Remote Control        (0000110e-0000-1000-8000-00805f9b34fb)
        UUID: A/V Remote Control Target (0000110c-0000-1000-8000-00805f9b34fb)
        UUID: Audio Source              (0000110a-0000-1000-8000-00805f9b34fb)
        UUID: Audio Sink                (0000110b-0000-1000-8000-00805f9b34fb)
        Modalias: usb:v1D6Bp0246d0518
        Discovering: no
          #
```

y se puede ver como no muestra un Puerto Serie. Pero al lanzar nuestro programa, podemos comprobar que al lanzar de nuevo el comando, se realiza la conexión y se crea el Puerto Serie:

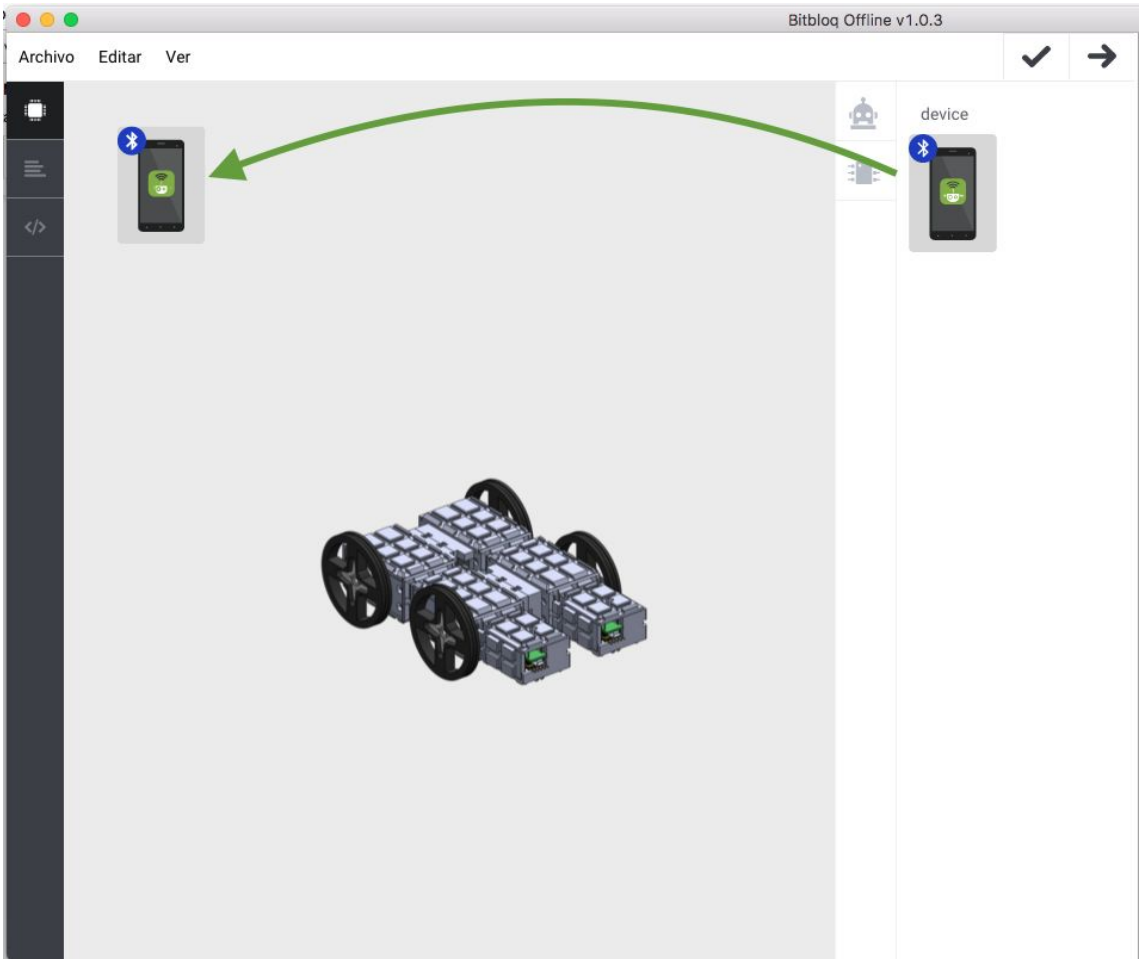
Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020



Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

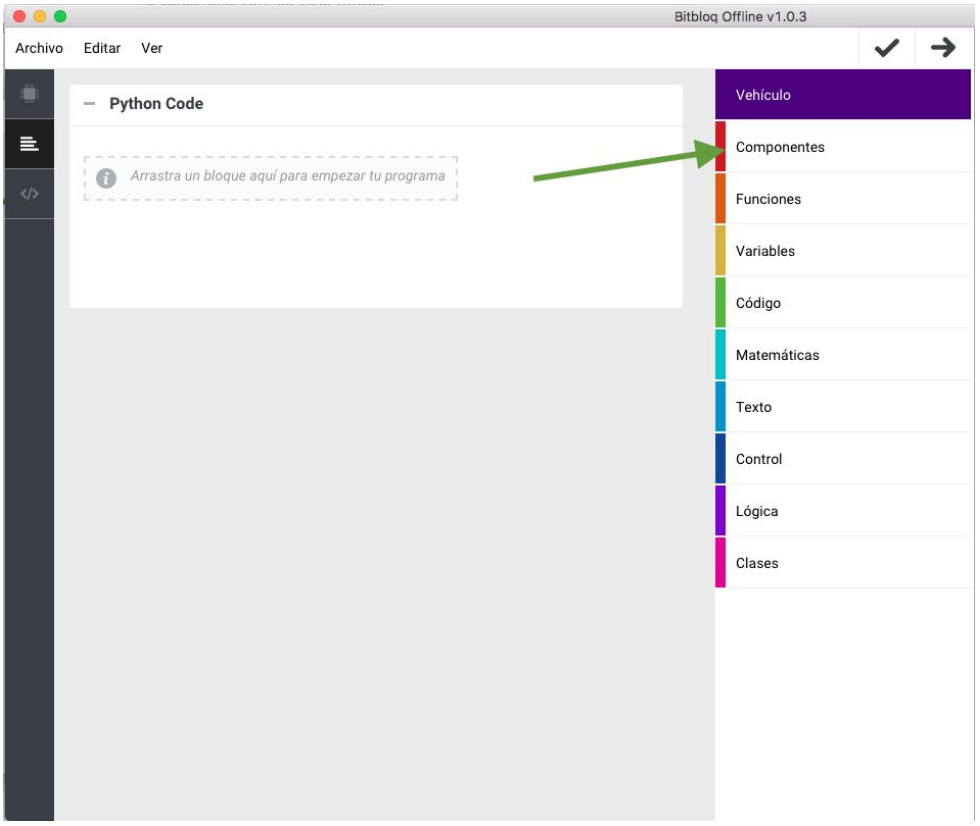
2.7.- Adaptación de la herramienta de programación de Botbloq

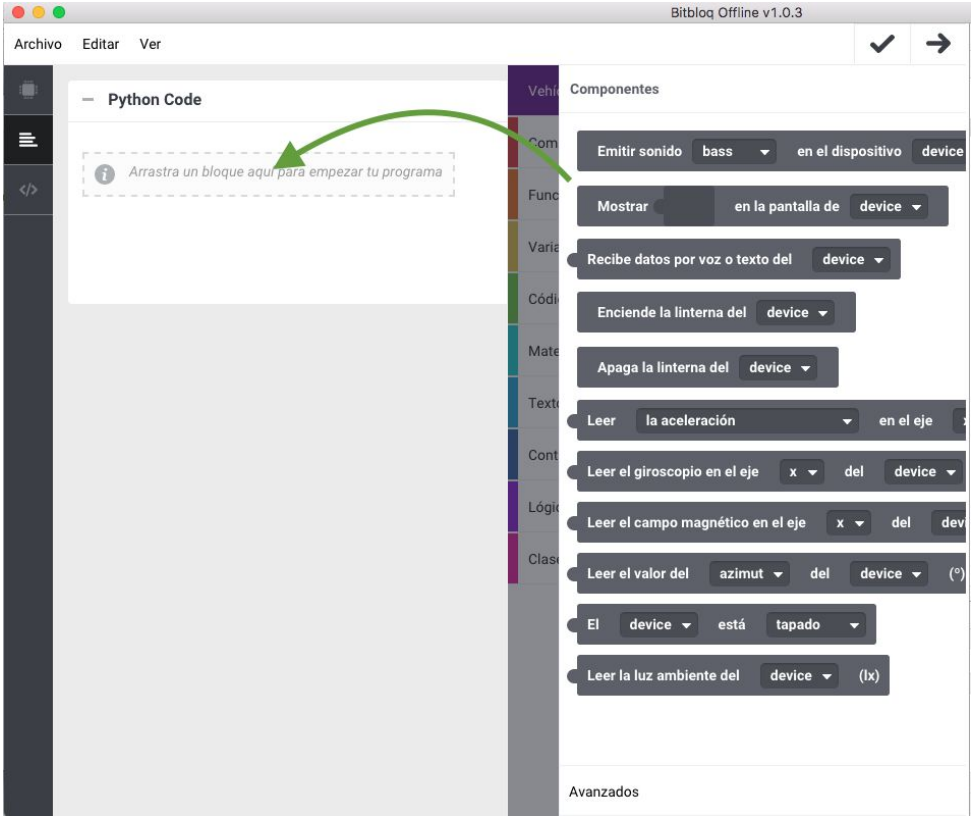
Se ha incorporado el menú de componentes para los robots de Botbloq en la herramienta.
De esta forma, se puede arrastrar al campo de trabajo para indicar que disponemos de un dispositivo móvil y que queremos utilizarlo.



Cuando el nuevo componente es añadido, en el menú de bloques de componentes, aparecen todos los bloques para interactuar con el dispositivo móvil desde los robots de Botbloq.

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020





Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

3.- Testing

Para poder probar la aplicación se ha usado los propios robots y se han creado programas de ejemplos, vamos a enumerar algunos de los ejemplos para ver que las placas se conectan de forma satisfactoria con los robots y que se comunicaban entre ellos.

Vehículo:

El vehículo se mueve recto durante 2 segundos si recibe un comando de "Adelante" desde el dispositivo móvil.

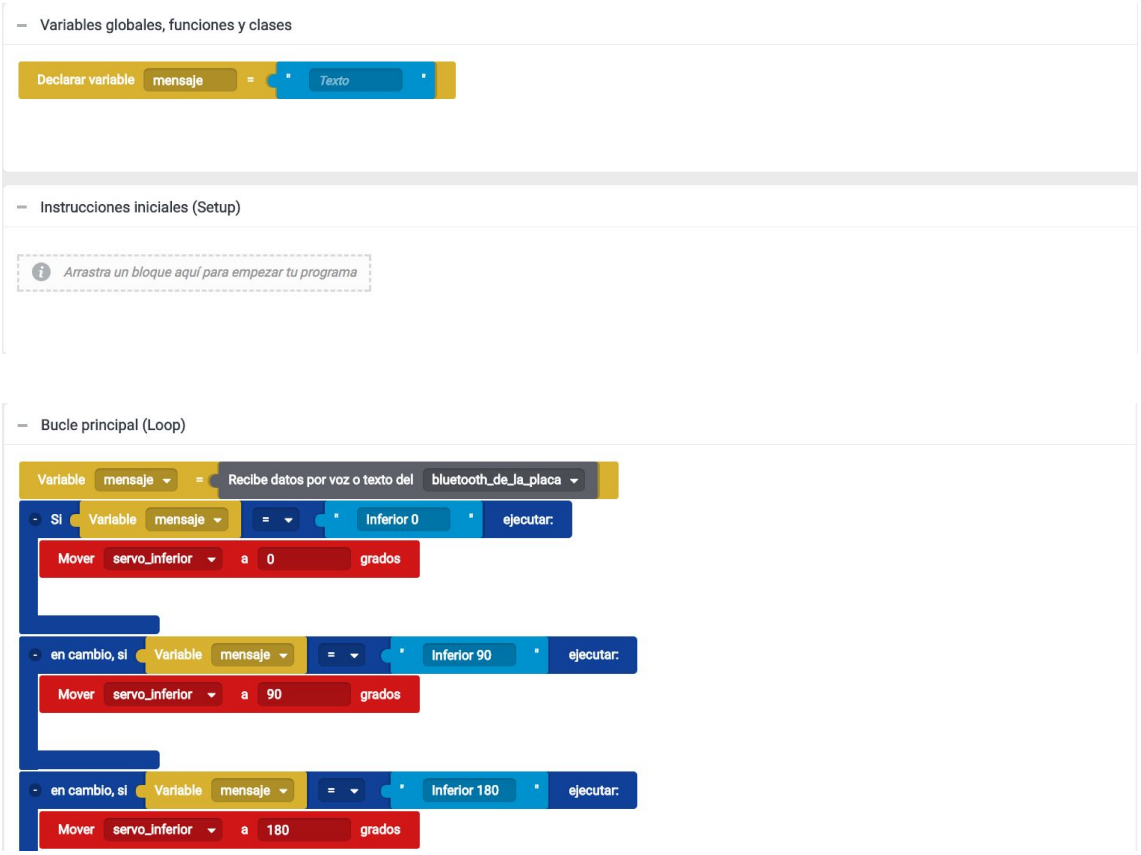
Manipulador:

Este programa mueve las juntas del manipulador de unos grados a otros, de forma progresiva.

Fichero Manipulador_serie_V2.bitbloq

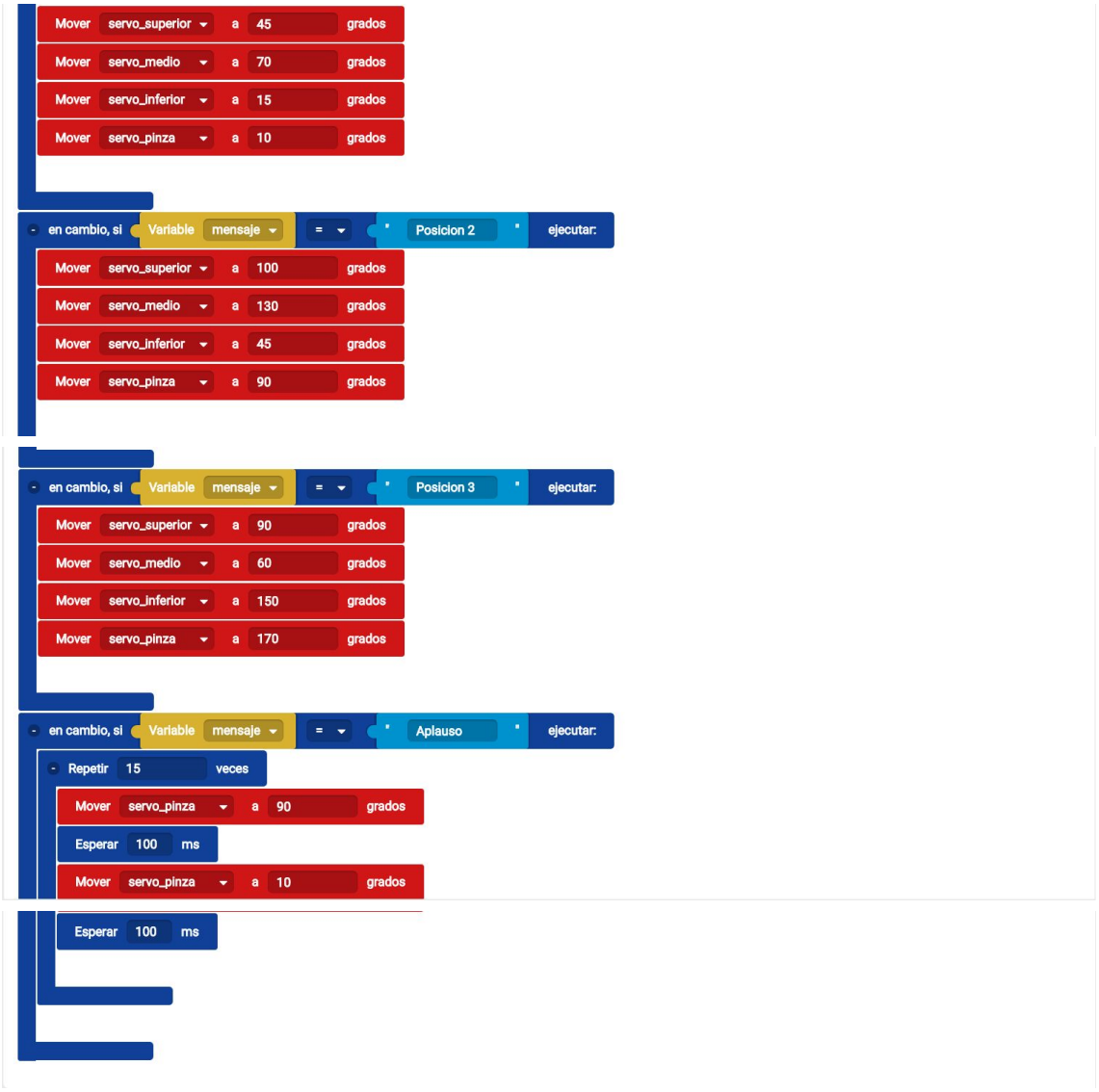
Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

Ejemplo con el manipulador y un dispositivo móvil, es también el siguiente programa que permite enviarle comandos desde el móvil para teleoperarlo. (incluido en el fichero Brazo_robotico_Botbloq.bitbloq)



Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020



Serpiente:

Serpiente que rueda hacia un lado si hay luz y hacia otro si no la detecta:

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

4.- Referencias

- [1] Python: https://es.wikipedia.org/wiki/Python
- [2] Lenguaje Arduino : https://es.wikipedia.org/wiki/Arduino
- [3] Official Linux Bluetooth protocol stack : http://www.bluez.org/

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020