

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

ENTREGABLE E.3.3. Módulos de programación de BOTBLOQ

RESUMEN DEL DOCUMENTO

Se ha adaptado la herramienta Bitbloq para permitir la programación de robots Botbloq, para ellos se ha transformado la herramienta en un generador de código python y se le ha añadido una interfaz para comunicarse con los robots que se encuentran en la misma red. De esta forma se pueden programar los robots con el mismo tipo de bloque definidos para los anteriores robots genéricos.

Índice

1.- ADAPTACIÓN DE LA HERRAMIENTA BITBLOQ PARA LOS ROBOTS BOTBLOQ	4
1.1.- ADAPTACIÓN DE LA LIBRERÍA DE BLOQUES PARA GENERACIÓN DE CÓDIGO PYTHON	4
1.2.- CREACIÓN DE LOS BLOQUES.....	9
1.2.1.- <i>Vehículo</i>	9
1.2.2.- <i>Manipulador</i>	16
1.2.3.- <i>Humanoide</i>	34
1.2.4.- <i>Serpiente</i>	48
1.2.5.- <i>Hexápodo</i>	59
1.3.- ADAPTACIÓN DE LA HERRAMIENTA ACTUAL PARA SU USO CON LOS ROBOTS DE BOTBLOQ.....	66
2.- VALIDACIÓN DE LA HERRAMIENTA	72

1.- Adaptación de la herramienta Bitbloq para los robots Botbloq

1.1.- Adaptación de la librería de bloques para generación de código python

Los robots de Botbloq necesitan de código en python para poder ser ejecutados, mientras que las placas contenidas en los Kits genéricos utilizan Arduino.

Debido a eso se ha añadido a la librería de bloques (<https://github.com/bq/bloqs>) la posibilidad de generación de código python a partir de los mismo bloques que ya se usaban antes para generar Arduino.

Para hacer dicha modificación se añadió un campo nuevo a los bloques en los que indicaba el código a generar.

Si recordamos los bloques están definidos en un formato .json de este tipo:

```
"readSensor"  
],  
  "name": "d9cd0dcb-2f1b-4a24-b493-4bb7195674b7"  
},  
{  
  "type": "connector--input",  
  "accept": "connector--output",  
  "acceptType": [  
    "all"  
  ],  
  "suggestedBloqs": [  
    "number",  
    "string",  
    "boolean",  
    "selectVariable",  
    "readSensor"  
  ],  
  "name": "4fc4f29-fd22-40e6-a12b-85d567f1f97f"  
}  
],  
  "bloqClass": "bloq-if",  
  "content": [  
    [  
      {  
        "alias": "text",  
        "value": "bloq-if-if"  
      },  
      {  
        "bloqInputId": "ARG1",  
        "alias": "bloqInput",  
        "acceptType": [  
          "all"  
        ],  
        "suggestedBloqs": [  
          "number",  
          "string",  
          "selectVariable",  
          "readSensor"  
        ],  
        "name": "d9cd0dcb-2f1b-4a24-b493-4bb7195674b7"  
      },  
      {  
        "id": "OPERATOR",  
        "alias": "staticDropdown",  
        "options": [  
          {  
            "label": "Si",  
            "value": "true"  
          },  
          {  
            "label": "No",  
            "value": "false"  
          }  
        ]  
      }  
    ]  
  ]  
}
```

```
"label": "=",
"value": "=="  
},  
{  
    "label": "!=",
    "value": "!="  
},  
{  
    "label": ">",
    "value": ">"  
},  
{  
    "label": ">=",
    "value": ">="  
},  
{  
    "label": "<",
    "value": "<"  
},  
{  
    "label": "<=",
    "value": "<="  
}  
]  
},  
{  
    "bloqInputId": "ARG2",
    "alias": "bloqInput",
    "acceptType": [
        "all"
    ],
    "suggestedBloqs": [
        "number",
        "string",
        "boolean",
        "selectVariable",
        "readSensor"
    ],
    "name": "4fca4f29-fd22-40e6-a12b-85d567f1f97f"
},
{
    "alias": "text",
    "value": "bloq-if-exec"
}
]  
],
```

```
"code": "if({ARG1} {OPERATOR} {ARG2}){{STATEMENTS}}",
"arduino": {
  "code": "if({ARG1} {OPERATOR} {ARG2}){{STATEMENTS}}"
},
"python": {
  "codeLines": [
    {
      "code": "if({ARG1} {OPERATOR} {ARG2}):"
    },
    {
      "indentation": 1,
      "code": "{STATEMENTS}"
    }
  ]
}
```

Resaltamos en color azul el identificador único del bloque, en color amarillo el fragmento que define la generación de código Arduino existente y luego en color verde la nueva definición que permite la generación de código Python.

Con esta definición que todo bloque lleva encima, se procesa en una librería nueva llamada python-generation

(<https://github.com/bq/bloqs/blob/master/src/scripts/python-generation.js>) y el resultado es una generación de código en Python.

La generación de código Python, no tiene la complejidad de los tipos de datos que tiene el código Arduino, pero si que tiene la complejidad de ser un código donde la indentación es vital para definir las estructuras en el código. Por esa razón en la definición se tendrán que indicar siempre las líneas de código y su indentación.

Las opciones para definir código Python son:

- *Libraries*: apartado donde indicaremos si necesita librerías externas, por ejemplo en los bloques de Botbloq del manipulador es necesario indicar la clase base “BotbloqManipulator”. Este dato es de tipo Array de strings.
- *needInstanceOf*: En este apartado se define si el código generado por el bloque va a necesitar de una instancia de un tipo de objeto específico, este es un dato de tipo Array de objetos que poseen un campo “name” (para especificar el nombre de la instancia del objeto que se va a crear), y un campo “type” (en donde se especifica el tipo de la instancia a crear).

```
"python": {
  "libraries": [
    "BotbloqManipulator"
  ],
  "needInstanceOf": [
    {
      "name": "manipulator",
```

```
"type": "BotbloqManipulator"  
}  
,  
"codeLines": [  
{  
  "code": "manipulator.move({X}, {Y}, {Z})"  
}  
]  
}
```

- *codeLines*: Este dato es de tipo Array de objetos y será donde se definan las líneas de código que generara el bloque. Cada objeto puede ser de dos tipos y pueden estar combinados sin problemas:
 - un “code”, que tendrá un campo obligatorio definido como “code” en donde se pondrá el pseudocódigo creado en la librería, y como optativo se podrá añadir un campo llamado “indentation” de tipo numérico donde se indica que esa línea tendrá una indentación extra.
 - un “conditional”, definido por un campo único llamado “conditional”, donde se difinen los campos “aliasId” para indicar el elemento del cual depende el condicional, y un campo “code” que es un mapa de valores, donde se define en pseudocódigo el código a generar según el valor del campo “aliasId”

```
{"python": {  
  "codeLines": [  
    {  
      "conditional": {  
        "aliasId": "MODE",  
        "code": {  
          "+": "for {VAR} in range({INIT}, {FINAL}):",  
          "-": "for {VAR} in range({FINAL}, {INIT}, -1):"  
        }  
      }  
    },  
    {  
      "indentation": 1,  
      "code": "{STATEMENTS}"  
    }  
  ]  
}
```

Esta nueva librería procesa los bloques, se le envía el árbol del programa que empieza con un bloque de main, y se van viendo las ramas mediante bucles recursivos para ir agrupando las importaciones de código necesarias, las instancias

necesarias por el código generado por los bloques , y después va procesando bloque a bloque el código que generan rellenando las partes marcadas por los símbolos “{” y “}” por el contenido de los elementos del bloque tales como el elemento de entrada de texto, o el selector de opciones. Generando así el código final en código Python sin errores de compilación y correcto. La librería se encuentra en el Anexo 3.3. fichero python-generation.js, los bloques se pueden encontrar en el Anexo 3.3. en la carpeta bloqs.

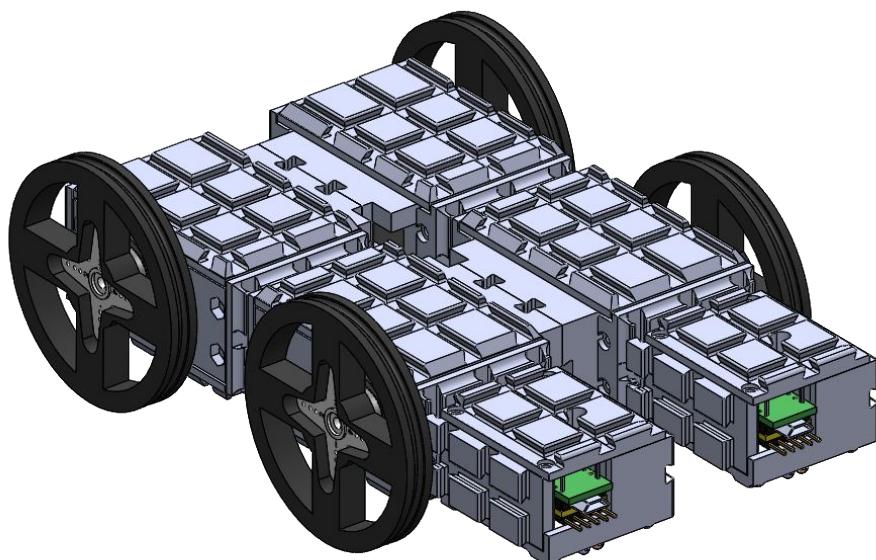
1.2.- Creación de los Bloques

Para que los usuarios puedan programar los robots de Botbloq vamos a necesitar crear nuevos bloques que permitan generar el código necesario.

Por ello vamos a detallar uno a uno los robots de Botbloq de forma superficial (más información sobre esos robots puede ser encontrada en los entregables del paquete 4).

1.2.1.-Vehículo

Este robot es un vehículo de 4 ruedas, formado por 4 servos que permiten su movimiento. Además lleva incorporado dos sensores de infrarrojos para poder ser utilizado para crear un robot sigue-líneas.



Este robot podrá andar hacia delante, hacia atrás, girar a la derecha e izquierda y por supuesto obtener información de los sensores.

1.2.1.1.- Bloque “*botbloqVehicleMove*”

Para los movimientos básicos hemos creado un bloque llamado “**botbloqVehicleMove**”:



Este bloque permite seleccionar la dirección, el tiempo por el cual se moverá el vehículo y la velocidad.

Este bloque generará un código de este tipo:

Archivo Editar Ver



```
# coding=utf-8
import BotbloqVehicle
vehicle = BotbloqVehicle.BotbloqVehicle()
vehicle.move("1000", "5", "FORWARD")
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqVehicleMove",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ]
}
```

10

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

```
        }
    ],
    "bloqClass": "bloq-botbloq-vehicle-move",
    "content": [
        [
            {
                "id": "MOVEMENT",
                "alias": "staticDropdown",
                "options": [
                    {
                        "label": "Avanzar",
                        "value": "FORWARD"
                    },
                    {
                        "label": "Retroceder",
                        "value": "BACKWARD"
                    },
                    {
                        "label": "Girar a la derecha",
                        "value": "TURN_RIGHT"
                    },
                    {
                        "label": "Girar a la izquierda",
                        "value": "TURN_LEFT"
                    }
                ]
            },
            {
                "alias": "text",
                "value": "durante"
            },
            {
                "id": "DELAY",
                "alias": "numberInput",
                "value": 1000
            },
            {
                "alias": "text",
                "value": "ms a una velocidad de"
            },
            {
                "id": "SPEED",
                "alias": "numberInput",
                "value": 5
            }
        ]
    ]
}
```

```
],  
"code": "",  
"python": {  
    "libraries": [  
        "BotbloqVehicle"  
    ],  
    "needInstanceOf": [  
        {  
            "name": "vehicle",  
            "type": "BotbloqVehicle"  
        }  
    ],  
    "codeLines": [  
        {  
            "code": "vehicle.move(\"{DELAY}\",\"{SPEED}\",\"{MOVEMENT}\")"  
        }  
    ]  
}  
}
```

1.2.1.2.- Bloque “**botbloqVehicleStop**”

Para poder detener el vehículo antes de tiempo o crear una interrupción se ha creado el bloque “**botbloqVehicleStop**”:



Parar

Este bloque permite la detención inmediata del vehículo, permitiendo a los usuarios que hagan una programación determinada, como por ejemplo un recorrido del vehículo, pero que luego permita interrumpir esa programación, como por ejemplo, al detectar un obstáculo si se le añade un sensor de ultrasonidos.

Este bloque generará un código de este tipo:



The screenshot shows a code editor window with a dark theme. On the left is a vertical toolbar with icons for file operations (New, Open, Save, etc.) and code folding. The main area contains the following Python code:

```
1 # coding=utf-8
2
3 import BotbloqVehicle
4
5
6 vehicle = BotbloqVehicle.BotbloqVehicle()
7
8
9 vehicle.stop()
10
11
12
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqVehicleStop",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-botbloq-vehicle-stop",
  "content": [
    [
      {
        "alias": "text",
        "value": "Parar"
      }
    ]
  ],
  "code": "",
  "python": {
    "libraries": [

```

13

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
"BotbloqVehicle"
],
"needInstanceOf": [
{
  "name": "vehicle",
  "type": "BotbloqVehicle"
}
],
"codeLines": [
{
  "code": "vehicle.stop()"
}
]
}
```

1.2.1.3.- Bloque “botbloqVehicleIRSensor”

Para controlar los sensores de Infrarrojos se ha creado este bloque “botbloqVehicleIRSensor”



Este sensor dará un valor booleano positivo si ha detectado algo en el sensor de infrarrojos que se le indica en el desplegable. Al ser un sensor es un bloque de tipo Output que tiene que ser colocado dentro de otro bloque. Permite la creación de sigue-líneas al incluirlo dentro de sentencias de control para modificar la trayectoria del vehículo según lo que detecte

Este bloque generará un código de este tipo:

Archivo Editar Ver



```
1 # coding=utf-8
2
3 import BotbloqVehicle
4
5
6 vehicle = BotbloqVehicle.BotbloqVehicle()
7
8
9 variableDeclarada = vehicle.readIRSensor("RIGHT")
10
11
12
```

Definición del bloque en formato JSON:

```
{
  "type": "output",
  "name": "botbloqVehicleIRSensor",
  "connectors": [
    {
      "type": "connector--output",
      "accept": "connector--input"
    }
  ],
  "bloqClass": "bloq-botbloq-vehicle-ir",
  "content": [
    [
      {
        "alias": "text",
        "value": "Leer sensor"
      },
      {
        "id": "SIDE",
        "alias": "staticDropdown",
        "options": [
          {
            "label": "derecho",
            "value": "RIGHT"
          }
        ]
      }
    ]
  ]
}
```

15

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



Centro para el
Desarrollo
Tecnológico
Industrial

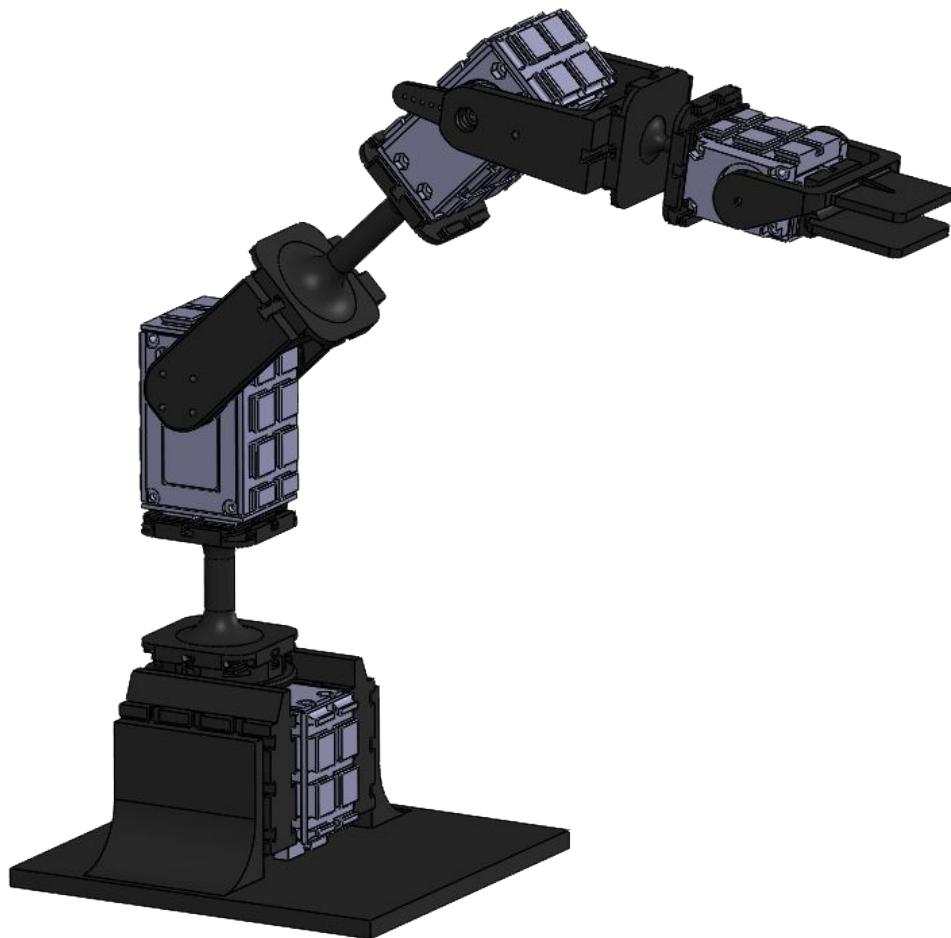


UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
        "label": "izquierdo",
        "value": "LEFT"
    }
]
}
],
"code": "",
"returnType": {
    "type": "simple",
    "value": "int"
},
"python": {
    "libraries": [
        "BotbloqVehicle"
    ],
    "needInstanceOf": [
        {
            "name": "vehicle",
            "type": "BotbloqVehicle"
        }
    ],
    "codeLines": [
        {
            "code": "vehicle.readIRSensor(\"{SIDE}\")"
        }
    ]
}
}
```

1.2.2.- Manipulador

Este robot es un brazo robótico con una pinza compuesto por 4 servos que permite crear programas para la manipulación de objetos.



Las acciones a realizar serán las de mover el brazo a una determinada posición X Y Z, y abrir y cerrar la pinza.

1.2.2.1.- Bloque “botbloqManipulatorCanMove”

Para poder saber si un robot manipulador puede realizar un movimiento se ha creado este bloque.



Los brazos manipuladores no siempre pueden moverse a una determinada posición, para poder saberlo con antelación y poder programar esos casos se ha creado este bloque.

Este bloque generará un código de este tipo:

17

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



```
Archivo Editar Ver

1 # coding=utf-8
2
3 import BotbloqManipulator
4
5
6 manipulator = BotbloqManipulator.BotbloqManipulator()
7
8
9 variableParaAlmacenarElValor = vehicle.canMove(0, 0, 0)
10
11
12
```

Definición del bloque en formato JSON:

```
{
  "type": "output",
  "name": "botbloqManipulatorCanMove",
  "connectors": [
    {
      "type": "connector--output",
      "accept": "connector--input"
    }
  ],
  "bloqClass": "bloq-botbloq-manipulator-canmove",
  "content": [
    [
      {
        "alias": "text",
        "value": "¿Puede moverse a "
      },
      {
        "alias": "text",
        "value": "x:"
      },
      {
        "id": "X",
        "alias": "numberInput",
        "value": 0
      }
    ]
}
```

```
"alias": "text",
"value": "y:",
},
{
"id": "Y",
"alias": "numberInput",
"value": 0
},
{
"alias": "text",
"value": "z:",
},
{
"id": "Z",
"alias": "numberInput",
"value": 0
},
{
"alias": "text",
"value": "?"
}
]
],
"code": "",
"returnType": {
"type": "simple",
"value": "bool"
},
"python": {
"libraries": [
"BotbloqManipulator"
],
"needInstanceOf": [
{
"name": "manipulator",
"type": "BotbloqManipulator"
}
],
"codeLines": [
{
"code": "vehicle.canMove({X}, {Y}, {Z})"
}
]
}
}
```

1.2.2.2.- Bloque “*botbloqManipulatorMoveOnSpace*”

Este bloque permitirá mandar una orden de movimiento al brazo para que se mueva a unas coordenadas X Y Z.

Desplazarse a x: 0 y: 0 z: 0

Este bloque generará un código de este tipo:



```
# coding=utf-8
import BotbloqManipulator
manipulator = BotbloqManipulator.BotbloqManipulator()
manipulator.move(0, 0, 0)
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqManipulatorMoveOnSpace",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-botbloq-manipulator-moveonspace",
  "content": [
```

20

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
[  
 {  
   "alias": "text",  
   "value": "Desplazarse a"  
 },  
 {  
   "alias": "text",  
   "value": "x:"  
 },  
 {  
   "id": "X",  
   "alias": "numberInput",  
   "value": 0  
 },  
 {  
   "alias": "text",  
   "value": "y:"  
 },  
 {  
   "id": "Y",  
   "alias": "numberInput",  
   "value": 0  
 },  
 {  
   "alias": "text",  
   "value": "z:"  
 },  
 {  
   "id": "Z",  
   "alias": "numberInput",  
   "value": 0  
 }  
 ]  
 ],  
 "code": "",  
 "python": {  
   "libraries": [  
     "BotbloqManipulator"  
   ],  
   "needInstanceOf": [  
     {  
       "name": "manipulator",  
       "type": "BotbloqManipulator"  
     }  
   ],  
   "codeLines": [  
     ...  
   ]  
 }
```

```
{  
    "code": "manipulator.move({X}, {Y}, {Z})"  
}  
]  
}  
}
```

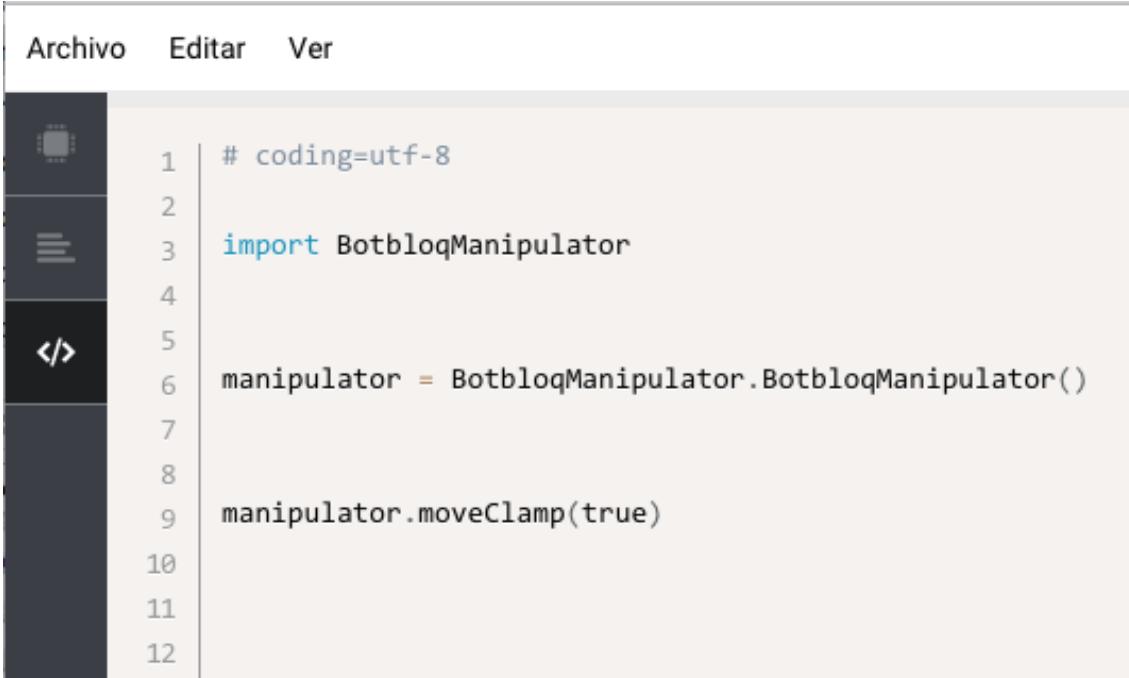
1.2.2.3.- Bloque “botbloqManipulatorMoveClamp”

Este bloque permite manejar la pinza



Los robots manipuladores suelen tener una pinza en su extremo, con este bloque podremos realizar acciones con esta pinza, para que programada junto con los bloques del movimiento se pueda, por ejemplo, desplazar el brazo a una posición XYZ, agarrar un objeto cerrando la pinza, mover el brazo a otra posición y soltar el objeto abriendo la pinza.

Este bloque generará un código de este tipo:



```
Archivo Editar Ver  
  
1 # coding=utf-8  
2  
3 import BotbloqManipulator  
4  
5  
6 manipulator = BotbloqManipulator.BotbloqManipulator()  
7  
8  
9 manipulator.moveClamp(true)  
10  
11  
12
```

Definición del bloque en formato JSON:

```
{  
  "type": "statement",  
  "name": "botbloqManipulatorMoveClamp",  
  "connectors": [  
    {  
      "type": "connector--top",  
      "accept": "connector--bottom"  
    },  
    {  
      "type": "connector--bottom",  
      "accept": "connector--top"  
    }  
  ],  
  "bloqClass": "bloq-botbloq-manipulator-moveclamp",  
  "content": [  
    [  
      {  
        "id": "ACTION",  
        "alias": "staticDropdown",  
        "options": [  
          {  
            "label": "Abrir",  
            "value": true  
          },  
          {  
            "label": "Cerrar",  
            "value": false  
          }  
        ]  
      },  
      {  
        "alias": "text",  
        "value": "la pinza"  
      }  
    ],  
    "code": "",  
    "python": {  
      "libraries": [  
        "BotbloqManipulator"  
      ],  
      "needInstanceOf": [  
        {  
          "type": "connector--top",  
          "accept": "connector--bottom"  
        }  
      ]  
    }  
  ]  
}
```

23

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
        "name": "manipulator",
        "type": "BotbloqManipulator"
    }
],
"codeLines": [
{
    "code": "manipulator.moveClamp({ACTION})"
}
]
}
```

1.2.2.4.- Bloque “*botbloqManipulatorRotateJoints*”

Este bloque permite rotar una junta específica del manipulador



Aunque para mover el manipulador a determinadas posiciones ya tenemos el otro bloque, puede ser que en ocasiones se quiera rotar una junta de forma específica, tal vez para crear una nueva trayectoria no contemplada.

Este bloque generará un código de este tipo:

Archivo Editar Ver

```
1 # coding=utf-8
2
3 import BotbloqManipulator
4
5
6 manipulator = BotbloqManipulator.BotbloqManipulator()
7
8
9 manipulator.moveJoint(1, 0)
10
11
12
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqManipulatorRotateJoints",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-botbloq-manipulator-rotatejoints",
  "content": [
    [
      {
        "alias": "text",
        "value": "Mover la articulación"
      },
      {
        "id": "JOINT_NUMBER",
        "alias": "staticDropdown",
        "options": [
          {
            "label": "1"
          },
          {
            "label": "2"
          },
          {
            "label": "3"
          },
          {
            "label": "4"
          },
          {
            "label": "5"
          }
        ]
      }
    ]
  ]
}
```

25

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
"label": "1",
  "value": "1"
},
{
  "label": "2",
  "value": "2"
},
{
  "label": "3",
  "value": "3"
}
],
{
  "id": "DEGREES",
  "alias": "numberInput",
  "value": 0
},
{
  "alias": "text",
  "value": "grados"
}
],
],
"code": "",
"python": {
  "libraries": [
    "BotbloqManipulator"
  ],
  "needInstanceOf": [
    {
      "name": "manipulator",
      "type": "BotbloqManipulator"
    }
  ],
  "codeLines": [
    {
      "code": "manipulator.moveJoint({JOINT_NUMBER}, {DEGREES})"
    }
  ]
}
```

1.2.2.5.- Bloque “*botbloqManipulatorCanMoveAdvanced*”

26

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

Esta es una versión para usuarios avanzados del bloque "botbloqManipulatorCanMove"



Este bloque al igual que su versión más sencilla permite saber si el manipulador es capaz de moverse a una determinada posición X Y Z.

Esta versión es para usuarios que quieran realizar acciones más avanzadas, como por ejemplo determinar los valores X Y Z mediante una operación matemática o realizar la comprobación dentro de un bucle de control para determinar valores correctos. Esta versión permite la inclusión de bloques dentro de los campos X Y Z para que esos parámetros puedan ser dinámicos.

Este bloque generará un código de este tipo:



```
# coding=utf-8
import BotbloqManipulator
manipulator = BotbloqManipulator.BotbloqManipulator()
puedeMoverse = vehicle.canMove(0, 0, 0)
```

Definición del bloque en formato JSON:

```
{
  "type": "output",
  "name": "botbloqManipulatorCanMoveAdvanced",
  "connectors": [
    {
      "type": "connector--output",
      "accept": "connector--input"
    }
  ]
}
```

```
},
{
  "type": "connector--input",
  "accept": "connector--output",
  "acceptType": [
    "all"
  ],
  "suggestedBloqs": [
    "number",
    "selectVariable"
  ],
  "name": "e0a1520d-02f9-45d9-a5b6-9db1ec23db7d"
},
{
  "type": "connector--input",
  "accept": "connector--output",
  "acceptType": [
    "all"
  ],
  "suggestedBloqs": [
    "number",
    "selectVariable"
  ],
  "name": "59f2d85e-d90d-4b6b-a7d6-1c3bfbe78c56"
},
{
  "type": "connector--input",
  "accept": "connector--output",
  "acceptType": [
    "all"
  ],
  "suggestedBloqs": [
    "number",
    "selectVariable"
  ],
  "name": "98729877-8960-42d7-b38d-2ad9d50d58e6"
},
],
"bloqClass": "bloq-botbloq-manipulator-canmove",
"content": [
  [
    {
      "alias": "text",
      "value": "¿Puede moverse a "
    },
    {

```

```
"alias": "text",
"value": "x:",
},
{
"bloqInputId": "X",
"alias": "bloqInput",
"acceptType": [
"all"
],
"suggestedBloqs": [
"number",
"selectVariable"
],
"name": "e0a1520d-02f9-45d9-a5b6-9db1ec23db7d"
},
{
"alias": "text",
"value": "y:",
},
{
"bloqInputId": "Y",
"alias": "bloqInput",
"acceptType": [
"all"
],
"suggestedBloqs": [
"number",
"selectVariable"
],
"name": "59f2d85e-d90d-4b6b-a7d6-1c3bfbe78c56"
},
{
"alias": "text",
"value": "z:",
},
{
"bloqInputId": "Z",
"alias": "bloqInput",
"acceptType": [
"all"
],
"suggestedBloqs": [
"number",
"selectVariable"
],
"name": "98729877-8960-42d7-b38d-2ad9d50d58e6"
}
```

```
        },
        {
          "alias": "text",
          "value": "?"
        }
      ],
      "code": "",
      "returnType": {
        "type": "simple",
        "value": "bool"
      },
      "python": {
        "libraries": [
          "BotbloqManipulator"
        ],
        "needInstanceOf": [
          {
            "name": "manipulator",
            "type": "BotbloqManipulator"
          }
        ],
        "codeLines": [
          {
            "code": "vehicle.canMove({X}, {Y}, {Z})"
          }
        ]
      }
    }
  }
```

1.2.2.6.- Bloque “botbloqManipulatorMoveOnSpaceAdvanced”

Este bloque es la versión avanzada del bloque “botbloqManipulatorMoveOnSpace”



Esta versión más avanzada del bloque permite introducir otros bloques en los parámetros X Y Z de la función de una forma dinámica, como por ejemplo añadiendo dentro una variable o el resultado de una operación aritmética. Puede ser usado, por ejemplo, para introducir el bloque dentro de una sentencia de control como la del bucle “for”, que modifique la variable usada en las coordenadas X Y Z para tener mayor control del movimiento y modificar por ejemplo la velocidad.

Este bloque generará un código de este tipo:

```
Archivo Editar Ver

1 # coding=utf-8
2
3 import BotbloqManipulator
4
5
6 manipulator = BotbloqManipulator.BotbloqManipulator()
7
8
9 manipulator.move(0, 0, 0)
10
11
12
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqManipulatorMoveOnSpaceAdvanced",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    },
    {
      "type": "connector--input",
      "accept": "connector--output",
      "acceptType": [
        "all"
      ],
      "suggestedBloqs": [
        "number",
        "selectVariable"
      ]
    }
  ]
}
```

```
"name": "ba673bc3-79d6-41cd-a97d-f3835862795a"  
},  
{  
  "type": "connector--input",  
  "accept": "connector--output",  
  "acceptType": [  
    "all"  
  ],  
  "suggestedBloqs": [  
    "number",  
    "selectVariable"  
  ],  
  "name": "3ea0c85a-bf9a-4398-9a84-52e8ae475921"  
},  
{  
  "type": "connector--input",  
  "accept": "connector--output",  
  "acceptType": [  
    "all"  
  ],  
  "suggestedBloqs": [  
    "number",  
    "selectVariable"  
  ],  
  "name": "79b36904-34da-4925-8daf-44a0fc2f9613"  
}  
],  
  "bloqClass": "bloq-botbloq-manipulator-moveonspace",  
  "content": [  
    [  
      {  
        "alias": "text",  
        "value": "Desplazarse a"  
      },  
      {  
        "alias": "text",  
        "value": "x:"  
      },  
      {  
        "bloqInputId": "X",  
        "alias": "bloqInput",  
        "acceptType": [  
          "all"  
        ],  
        "suggestedBloqs": [  
          "number",  
          "operator",  
          "text"  
        ]  
      }  
    ]  
  ]  
}
```

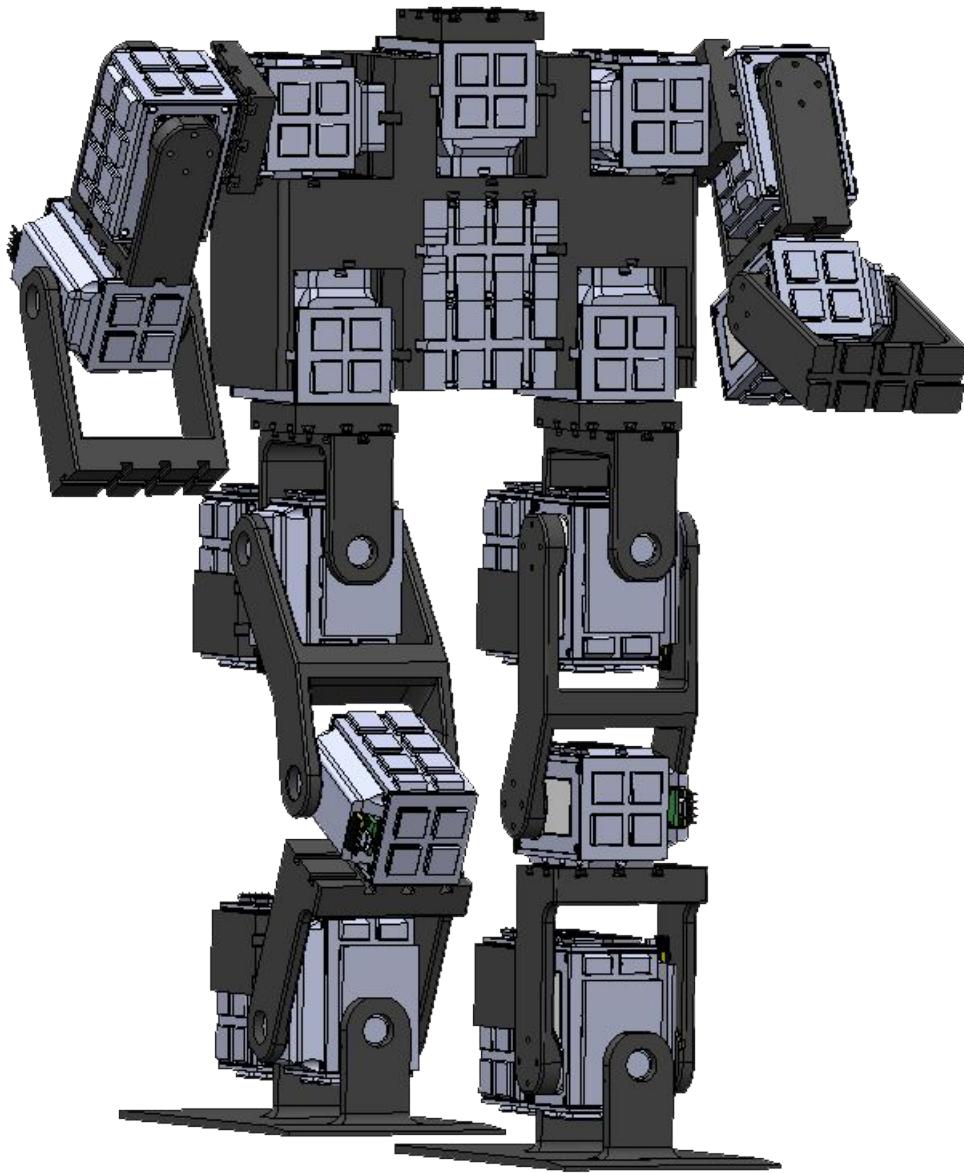
```
        "selectVariable"
    ],
    "name": "ba673bc3-79d6-41cd-a97d-f3835862795a"
},
{
    "alias": "text",
    "value": "y:"
},
{
    "bloqInputId": "Y",
    "alias": "bloqInput",
    "acceptType": [
        "all"
    ],
    "suggestedBloqs": [
        "number",
        "selectVariable"
    ],
    "name": "3ea0c85a-bf9a-4398-9a84-52e8ae475921"
},
{
    "alias": "text",
    "value": "z:"
},
{
    "bloqInputId": "Z",
    "alias": "bloqInput",
    "acceptType": [
        "all"
    ],
    "suggestedBloqs": [
        "number",
        "selectVariable"
    ],
    "name": "79b36904-34da-4925-8daf-44a0fc2f9613"
}
],
],
"code": "",
"python": {
    "libraries": [
        "BotbloqManipulator"
    ],
    "needInstanceOf": [
    {
        "name": "manipulator",

```

```
        "type": "BotbloqManipulator"  
    }  
],  
"codeLines": [  
    {  
        "code": "manipulator.move({X}, {Y}, {Z})"  
    }  
]  
}  
}
```

1.2.3.- Humanoide

Un humanoide es una replica de un humano creada mediante varios servos que permitan al robot hacer movimiento típicos como andar, subir escaleras o agacharse.



1.2.3.1.- Bloque “botbloqHumanBendDown”

Este bloque realiza un movimiento de agacharse en el humanoide

Agacharse

Este bloque generará un código de este tipo:

35

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

Archivo Editar Ver



```
# coding=utf-8
import BotbloqHuman
human = BotbloqHuman.BotbloqHuman()
human.benddown()
```

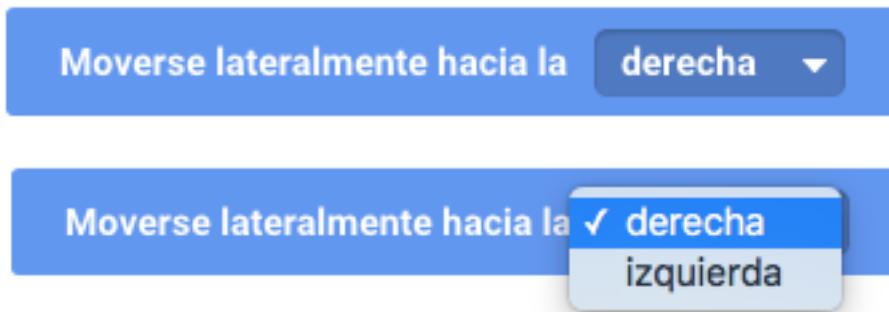
Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqHumanBendDown",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-botbloq-human-benddown",
  "content": [
    {
      "alias": "text",
      "value": "Agacharse"
    }
  ]
}
```

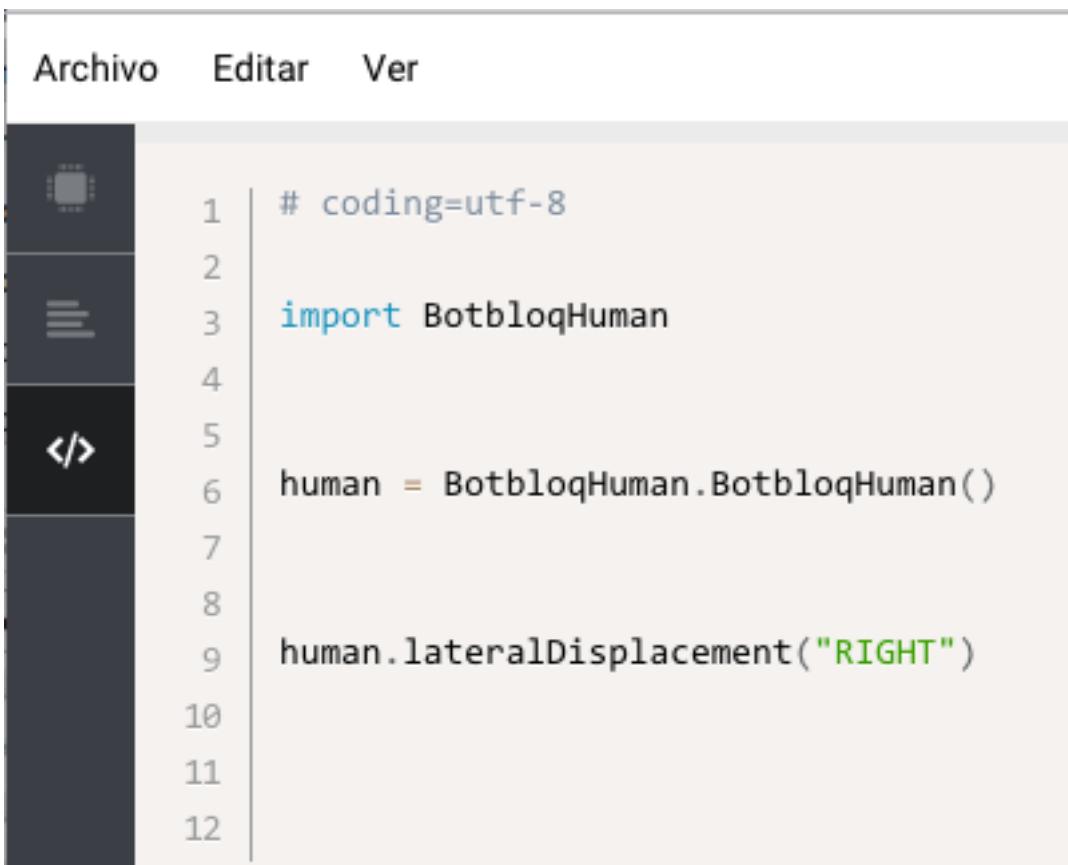
```
]
],
"code": "",
"python": {
  "libraries": [
    "BotbloqHuman"
  ],
  "needInstanceOf": [
    {
      "name": "human",
      "type": "BotbloqHuman"
    }
  ],
  "codeLines": [
    {
      "code": "human.benddown()"
    }
  ]
}
```

1.2.3.2.- Bloque “*botbloqHumanLateralDisplacement*”

Este bloque sirve para ordenarle al robot que se mueva de forma lateral, tanto hacia la derecho como hacia la izquierda.



Este bloque generará un código de este tipo:



```
# coding=utf-8
import BotbloqHuman
human = BotbloqHuman.BotbloqHuman()
human.lateralDisplacement("RIGHT")
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqHumanLateralDisplacement",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-botbloq-human-lateraldisplacement",
  "content": [
    [
      {
        "alias": "text",
        "value": "Moverse lateralmente hacia la"
      }
    ]
  ]
}
```

```
{  
  "id": "FOOT",  
  "alias": "staticDropdown",  
  "options": [  
    {  
      "label": "derecha",  
      "value": "RIGHT"  
    },  
    {  
      "label": "izquierda",  
      "value": "LEFT"  
    }  
  ]  
},  
"code": "",  
"python": {  
  "libraries": [  
    "BotbloqHuman"  
  ],  
  "needInstanceOf": [  
    {  
      "name": "human",  
      "type": "BotbloqHuman"  
    }  
  ],  
  "codeLines": [  
    {  
      "code": "human.lateralDisplacement(\"{FOOT}\")"  
    }  
  ]  
}  
}
```

1.2.3.3.- Bloque “botbloqHumanMove”

Este bloque permite programar movimientos al humanoide.



- ✓ Avanzar
- Retroceder
- Girar a la derecha
- Girar a la izquierda

Podremos programar que avance, retroceda o que gire en ambas direcciones, el robot realizará esos movimiento de forma continua.

Este bloque generará un código de este tipo:

Archivo Editar Ver



```
# coding=utf-8
import BotbloqHuman
human = BotbloqHuman.BotbloqHuman()
human.move("FORWARD")
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqHumanMove",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    }
  ]
}
```

40

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

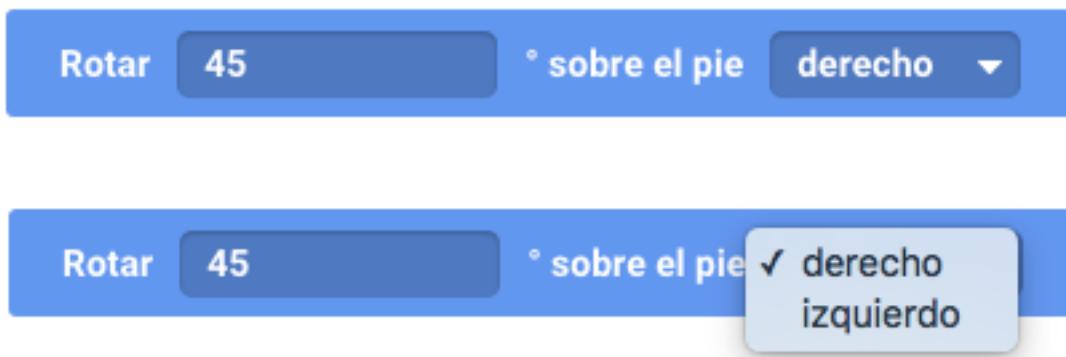
```
},
{
  "type": "connector--bottom",
  "accept": "connector--top"
}
],
"bloqClass": "bloq-botbloq-human-move",
"content": [
[
  {
    "id": "MOVEMENT",
    "alias": "staticDropdown",
    "options": [
      {
        "label": "Avanzar",
        "value": "FORWARD"
      },
      {
        "label": "Retroceder",
        "value": "BACKWARD"
      },
      {
        "label": "Girar a la derecha",
        "value": "TURN_RIGHT"
      },
      {
        "label": "Girar a la izquierda",
        "value": "TURN_LEFT"
      }
    ]
  }
],
"code": "",
"python": {
  "libraries": [
    "BotbloqHuman"
  ],
  "needInstanceOf": [
    {
      "name": "human",
      "type": "BotbloqHuman"
    }
  ],
  "codeLines": [
    {

```

```
"code": "human.move(\"{MOVEMENT}\")"  
}  
]  
}  
}
```

1.2.3.4.- Bloque “*botbloqHumanRotateByFoot*”

Este bloque permite al humanoide rotar un determinado ángulo sobre uno de sus pies. De forma que se puedan realizar mejores giros.



Este bloque generará un código de este tipo:

42

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

Archivo Editar Ver



```
# coding=utf-8
import BotbloqHuman
human = BotbloqHuman.BotbloqHuman()
human.rotateByFoot(45, "RIGHT")
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqHumanRotateByFoot",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-botbloq-human-rotatefoot",
  "content": [
    [
      {
        "alias": "text",
        "value": "Rotar"
      }
    ]
  ]
}
```

43

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
{  
  "id": "DEGREES",  
  "alias": "numberInput",  
  "value": 45  
},  
{  
  "alias": "text",  
  "value": "° sobre el pie"  
},  
{  
  "id": "FOOT",  
  "alias": "staticDropdown",  
  "options": [  
    {  
      "label": "derecho",  
      "value": "RIGHT"  
    },  
    {  
      "label": "izquierdo",  
      "value": "LEFT"  
    }  
  ]  
},  
]  
,  
"code": "",  
"python": {  
  "libraries": [  
    "BotbloqHuman"  
  ],  
  "needInstanceOf": [  
    {  
      "name": "human",  
      "type": "BotbloqHuman"  
    }  
  ],  
  "codeLines": [  
    {  
      "code": "human.rotateByFoot({DEGREES},{FOOT})"  
    }  
  ]  
}  
}
```

1.2.3.5.- Bloque “botbloqHumanStop”

44

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

Este bloque detiene al humanoide.

Detener

Ya que el bloque de avanzar permite definir una dirección y que el humanoide vaya sin para, para poder pararlo es necesario este bloque.

Este bloque generará un código de este tipo:

Archivo Editar Ver



```
# coding=utf-8
import BotbloqHuman
human = BotbloqHuman.BotbloqHuman()
human.stop()
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqHumanStop",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ]
}
```

45

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
        "accept": "connector--top"
    }
],
"bloqClass": "bloq-botbloq-human-stop",
"content": [
[
{
    "alias": "text",
    "value": "Detener"
}
]
],
"code": "",
"python": {
"libraries": [
"BotbloqHuman"
],
"needInstanceOf": [
{
    "name": "human",
    "type": "BotbloqHuman"
}
],
"codeLines": [
{
    "code": "human.stop()"
}
]
}
}
```

1.2.3.6.- Bloque “botbloqHumanUpstairs”

Con este bloque le indicaremos al humanoide que queremos que suba escaleras.

Subir escaleras.

Este bloque generará un código de este tipo:

Archivo Editar Ver



```
# coding=utf-8
import BotbloqHuman
human = BotbloqHuman.BotbloqHuman()
human.upstairs()
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqHumanUpstairs",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-botbloq-human-upstairs",
  "content": [
    [
      {
        "alias": "text",
        "value": "Subir escaleras."
      }
    ]
  ]
}
```

47

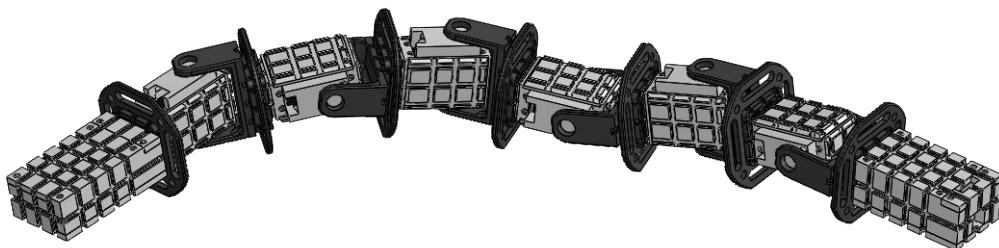
Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289

Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa
Operativo Plurirregional de Crecimiento Inteligente 2014-2020

```
],
  "code": "",
  "python": {
    "libraries": [
      "BotbloqHuman"
    ],
    "needInstanceOf": [
      {
        "name": "human",
        "type": "BotbloqHuman"
      }
    ],
    "codeLines": [
      {
        "code": "human.upstairs()"
      }
    ]
  }
}
```

1.2.4.- Serpiente

La serpiente es un conjunto de servos coordinados para imitar y por tanto moverse como una serpiente.



Tendremos bloques para poder ir hacia delante, hacia atrás, desplazarse lateralmente , para que pueda rodar y para que pueda rotar.

1.2.4.1.- Bloque “*botbloqSnakeLateralDisplacement*”

Este bloque permite programar a la serpiente para que se mueva de forma lateral, tanto a la derecha como a la izquierda.

Moverse lateralmente hacia la **derecha** ▾

Moverse lateralmente hacia la ✓ **derecha izquierda**

Este bloque generará un código de este tipo:

```
Archivo Editar Ver

# coding=utf-8
import BotbloqSnake
snake = BotbloqSnake.BotbloqSnake()
snake.lateralDisplacement("RIGHT")
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqSnakeLateralDisplacement",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
  ]}
```

49

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
{  
  "type": "connector--bottom",  
  "accept": "connector--top"  
}  
],  
"bloqClass": "bloq-botbloq-snake-lateraldisplacement",  
"content": [  
  [  
    {  
      "alias": "text",  
      "value": "Moverse lateralmente hacia la"  
    },  
    {  
      "id": "SIDE",  
      "alias": "staticDropdown",  
      "options": [  
        {  
          "label": "derecha",  
          "value": "RIGHT"  
        },  
        {  
          "label": "izquierda",  
          "value": "LEFT"  
        }  
      ]  
    }  
  ],  
  "code": "",  
  "python": {  
    "libraries": [  
      "BotbloqSnake"  
    ],  
    "needInstanceOf": [  
      {  
        "name": "snake",  
        "type": "BotbloqSnake"  
      }  
    ],  
    "codeLines": [  
      {  
        "code": "snake.lateralDisplacement(\"{SIDE}\")"  
      }  
    ]  
  }  
}
```

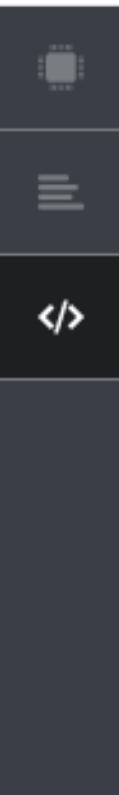
1.2.4.2.- Bloque “botbloqSnakeMove”

Con este bloque podremos programar a la serpiente para que se desplace de forma continua hacia delante y hacia atrás.



Este bloque generará un código de este tipo:

Archivo Editar Ver



```
# coding=utf-8
import BotbloqSnake
snake = BotbloqSnake.BotbloqSnake()
snake.move("FORWARD")
```

Definición del bloque en formato JSON:

51

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

```
{  
  "type": "statement",  
  "name": "botbloqSnakeMove",  
  "connectors": [  
    {  
      "type": "connector--top",  
      "accept": "connector--bottom"  
    },  
    {  
      "type": "connector--bottom",  
      "accept": "connector--top"  
    }  
],  
  "bloqClass": "bloq-botbloq-snake-move",  
  "content": [  
    [  
      {  

```

```
        }
    ]
}
],
"code": "",

"python": {

"libraries": [
    "BotbloqSnake"
],
"needInstanceOf": [
{
    "name": "snake",
    "type": "BotbloqSnake"
}
],
"codeLines": [
{
    "code": "snake.move(\"{MOVEMENT}\")"
}
]
}
}
```

1.2.4.3.- Bloque “*botbloqSnakeRoll*”

Con este bloque podremos hacer que la serpiente ruede hacia la derecha o hacia la izquierda.

Rodar hacia la **derecha** ▾

Rodar hacia la **✓ derecha izquierda**

Este bloque generará un código de este tipo:

```
Archivo Editar Ver

# coding=utf-8
import BotbloqSnake
snake = BotbloqSnake.BotbloqSnake()
snake.roll("RIGHT")
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqSnakeRoll",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
  ]}
```

54

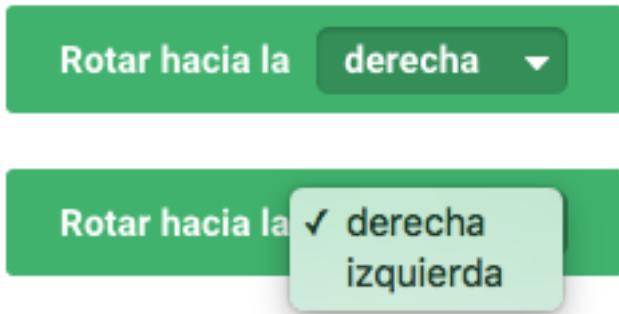
Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289

Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

```
{  
  "type": "connector--bottom",  
  "accept": "connector--top"  
}  
],  
"bloqClass": "bloq-botbloq-snake-roll",  
"content": [  
  [  
    {  
      "alias": "text",  
      "value": "Rodar hacia la"  
    },  
    {  
      "id": "SIDE",  
      "alias": "staticDropdown",  
      "options": [  
        {  
          "label": "derecha",  
          "value": "RIGHT"  
        },  
        {  
          "label": "izquierda",  
          "value": "LEFT"  
        }  
      ]  
    }  
  ],  
  "code": "",  
  "python": {  
    "libraries": [  
      "BotbloqSnake"  
    ],  
    "needInstanceOf": [  
      {  
        "name": "snake",  
        "type": "BotbloqSnake"  
      }  
    ],  
    "codeLines": [  
      {  
        "code": "snake.roll(\"{SIDE}\")"  
      }  
    ]  
  }  
}
```

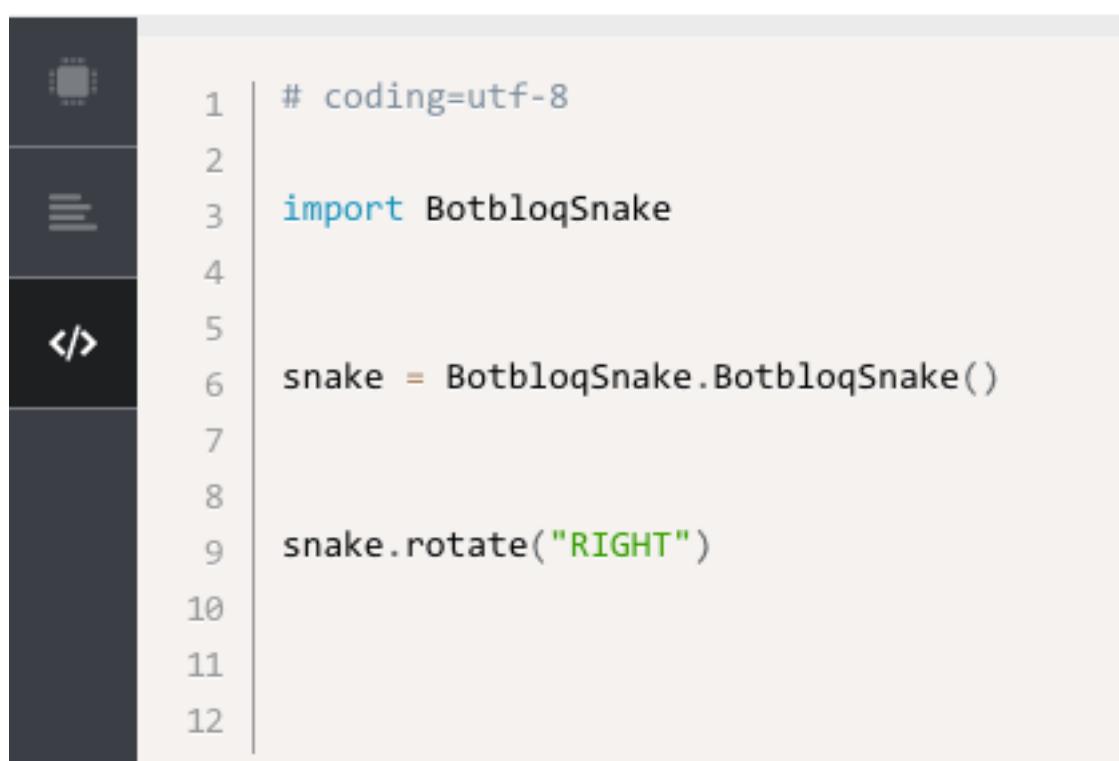
1.2.4.4.- Bloque "botbloqSnakeRotate"

Este bloque hará que la serpiente pueda rotar.



Este bloque generará un código de este tipo:

Archivo Editar Ver



```
# coding=utf-8
import BotbloqSnake
snake = BotbloqSnake.BotbloqSnake()
snake.rotate("RIGHT")
```

Definición del bloque en formato JSON:

```
{  
  "type": "statement",  
  "name": "botbloqSnakeRotate",
```

56

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

```
"connectors": [  
    {  
        "type": "connector--top",  
        "accept": "connector--bottom"  
    },  
    {  
        "type": "connector--bottom",  
        "accept": "connector--top"  
    }  
],  
"bloqClass": "bloq-botbloq-snake-rotate",  
"content": [  
    [  
        {  
            "alias": "text",  
            "value": "Rotar hacia la"  
        },  
        {  
            "id": "SIDE",  
            "alias": "staticDropdown",  
            "options": [  
                {  
                    "label": "derecha",  
                    "value": "RIGHT"  
                },  
                {  
                    "label": "izquierda",  
                    "value": "LEFT"  
                }  
            ]  
        }  
    ],  
    "code": "",  
    "python": {  
        "libraries": [  
            "BotbloqSnake"  
        ],  
        "needInstanceOf": [  
            {  
                "name": "snake",  
                "type": "BotbloqSnake"  
            }  
        ],  
        "codeLines": [  
            {
```

```
"code": "snake.rotate(\"{SIDE}\")"  
}  
]  
}  
}
```

1.2.4.5.- Bloque “botbloqSnakeStop”

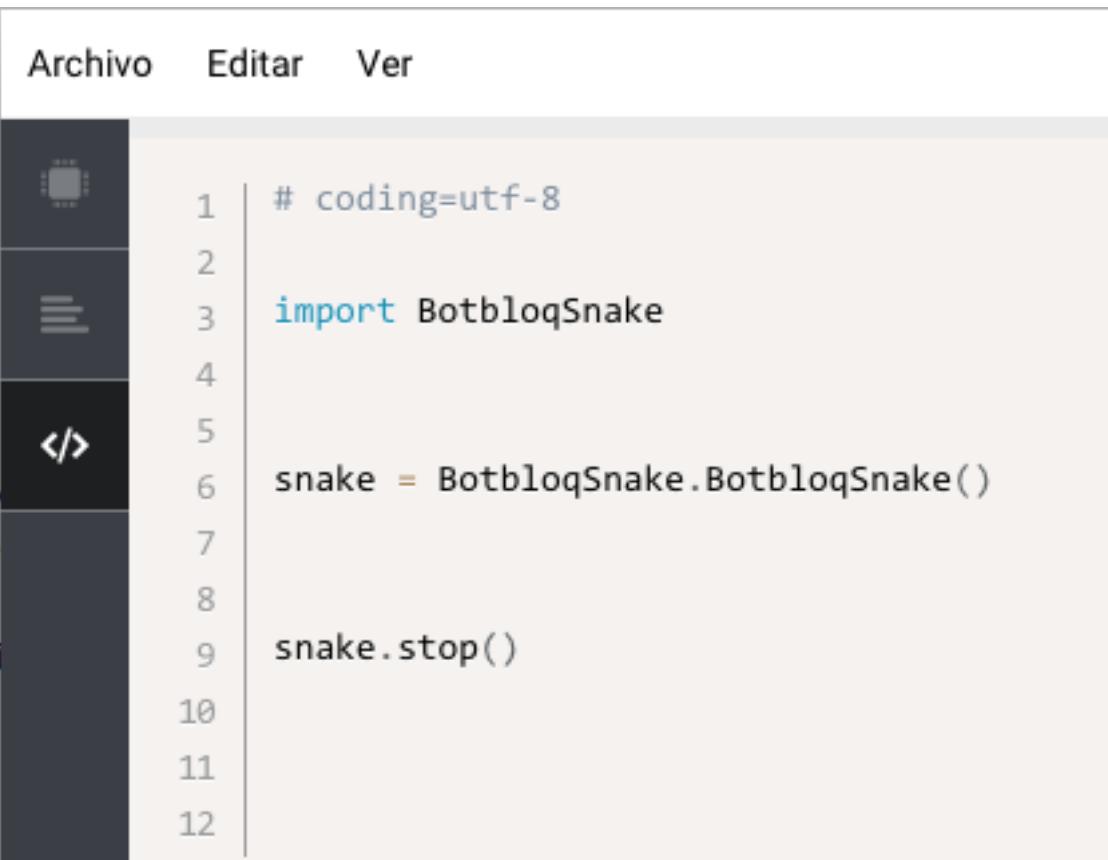
Con este bloque podremos programar a la serpiente para que se detenga.



Detener

Con los bloques de movimiento la serpiente no va a parar de moverse, si se quiere añadir una interrupción a su movimiento hay que usar este bloque

Este bloque generará un código de este tipo:



```
Archivo Editar Ver  
  
1 # coding=utf-8  
2  
3 import BotbloqSnake  
4  
5  
6 snake = BotbloqSnake.BotbloqSnake()  
7  
8  
9 snake.stop()  
10  
11  
12
```

Definición del bloque en formato JSON:

58

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
{  
  "type": "statement",  
  "name": "botbloqSnakeStop",  
  "connectors": [  
    {  
      "type": "connector--top",  
      "accept": "connector--bottom"  
    },  
    {  
      "type": "connector--bottom",  
      "accept": "connector--top"  
    }  
  ],  
  "bloqClass": "bloq-botbloq-snake-stop",  
  "content": [  
    [  
      {  
        "alias": "text",  
        "value": "Detener"  
      }  
    ]  
  ],  
  "code": "",  
  "python": {  
    "libraries": [  
      "BotbloqSnake"  
    ],  
    "needInstanceOf": [  
      {  
        "name": "snake",  
        "type": "BotbloqSnake"  
      }  
    ],  
    "codeLines": [  
      {  
        "code": "snake.stop()"  
      }  
    ]  
  }  
}
```

1.2.5.- Hexápodo

59

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

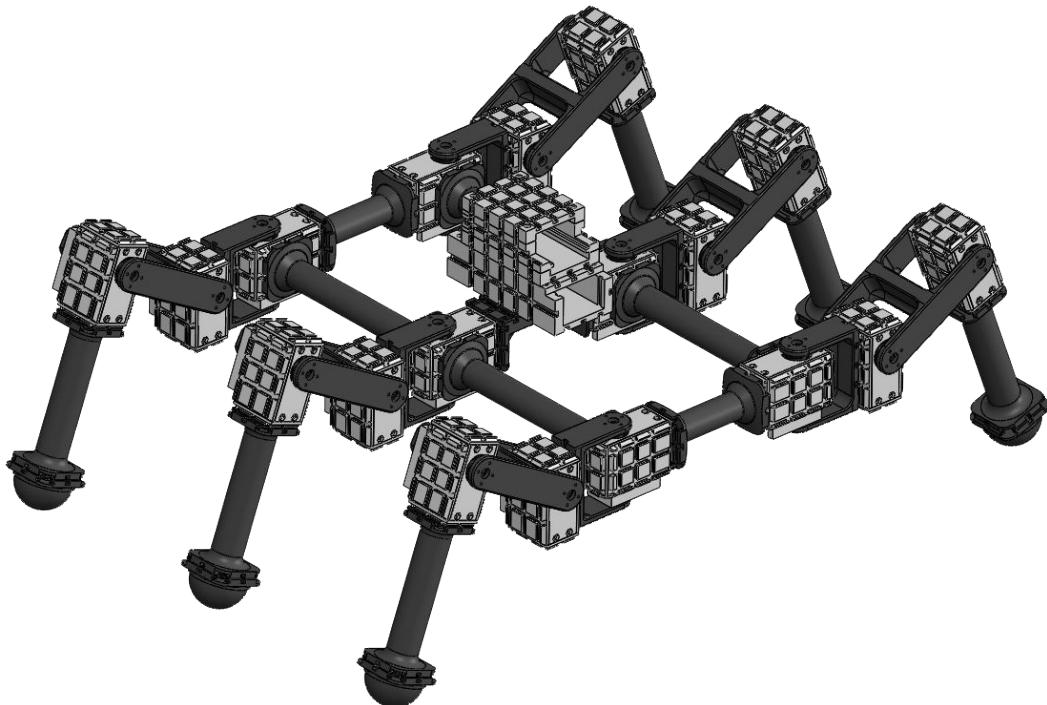


Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

Este robot hace usos de todos sus servos para emular a un hexápedo. Le permite moverse por terreno irregular sin los problemas de otros como el vehículo o el humanoide, y le da una sensación de araña que atrae mucho la curiosidad de los niños.



1.2.5.1.- Bloque “botbloqHexapodLateralDisplacement”

Con este bloque podremos programar el hexápedo para que camine lateralmente.

Moverse lateralmente hacia la **derecha** ▾

Moverse lateralmente hacia la **✓ derecha izquierda**

Este bloque generará un código de este tipo:

60

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

Archivo Editar Ver

```
1 # coding=utf-8
2
3 import BotbloqHexapod
4
5
6 hexapod = BotbloqHexapod.BotbloqHexapod()
7
8
9 hexapod.lateralDisplacement("RIGHT")
10
11
12
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqHexapodLateralDisplacement",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-botbloq-hexapod-lateraldisplacement",
  "content": [
    [
      {
        "alias": "text",
        "value": "Moverse lateralmente hacia la"
      },
      {
        "id": "SIDE",
        "value": "Derecha"
      }
    ]
  ]
}
```

```
"alias": "staticDropdown",
"options": [
{
  "label": "derecha",
  "value": "RIGHT"
},
{
  "label": "izquierda",
  "value": "LEFT"
}
],
"code": "",
"python": {
  "libraries": [
    "BotbloqHexapod"
  ],
  "needInstanceOf": [
    {
      "name": "hexapod",
      "type": "BotbloqHexapod"
    }
  ],
  "codeLines": [
    {
      "code": "hexapod.lateralDisplacement(\"{SIDE}\")"
    }
  ]
}
}
```

1.2.5.2.- Bloque “*botbloqHexapodMove*”

Este bloque nos permitirá mover el robot hacia delante, hacia atrás, girar a la derecha y a la izquierda. Dándonos total control sobre su movilidad.

Avanzar ▾

62

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

- ✓ Avanzar
- Retroceder
- Girar a la derecha
- Girar a la izquierda

Este bloque generará un código de este tipo:

Archivo Editar Ver



```
# coding=utf-8
import BotbloqHexapod
hexapod = BotbloqHexapod.BotbloqHexapod()
hexapod.move("FORWARD")
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqHexapodMove",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ]
}
```

63

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
        }
    ],
    "bloqClass": "bloq-botbloq-hexapod-move",
    "content": [
        [
            {
                "id": "MOVEMENT",
                "alias": "staticDropdown",
                "options": [
                    {
                        "label": "Avanzar",
                        "value": "FORWARD"
                    },
                    {
                        "label": "Retroceder",
                        "value": "BACKWARD"
                    },
                    {
                        "label": "Girar a la derecha",
                        "value": "TURN_RIGHT"
                    },
                    {
                        "label": "Girar a la izquierda",
                        "value": "TURN_LEFT"
                    }
                ]
            }
        ],
        "code": "",
        "python": {
            "libraries": [
                "BotbloqHexapod"
            ],
            "needInstanceOf": [
                {
                    "name": "hexapod",
                    "type": "BotbloqHexapod"
                }
            ],
            "codeLines": [
                {
                    "code": "hexapod.move(\"{MOVEMENT}\")"
                }
            ]
        }
    ]
}
```

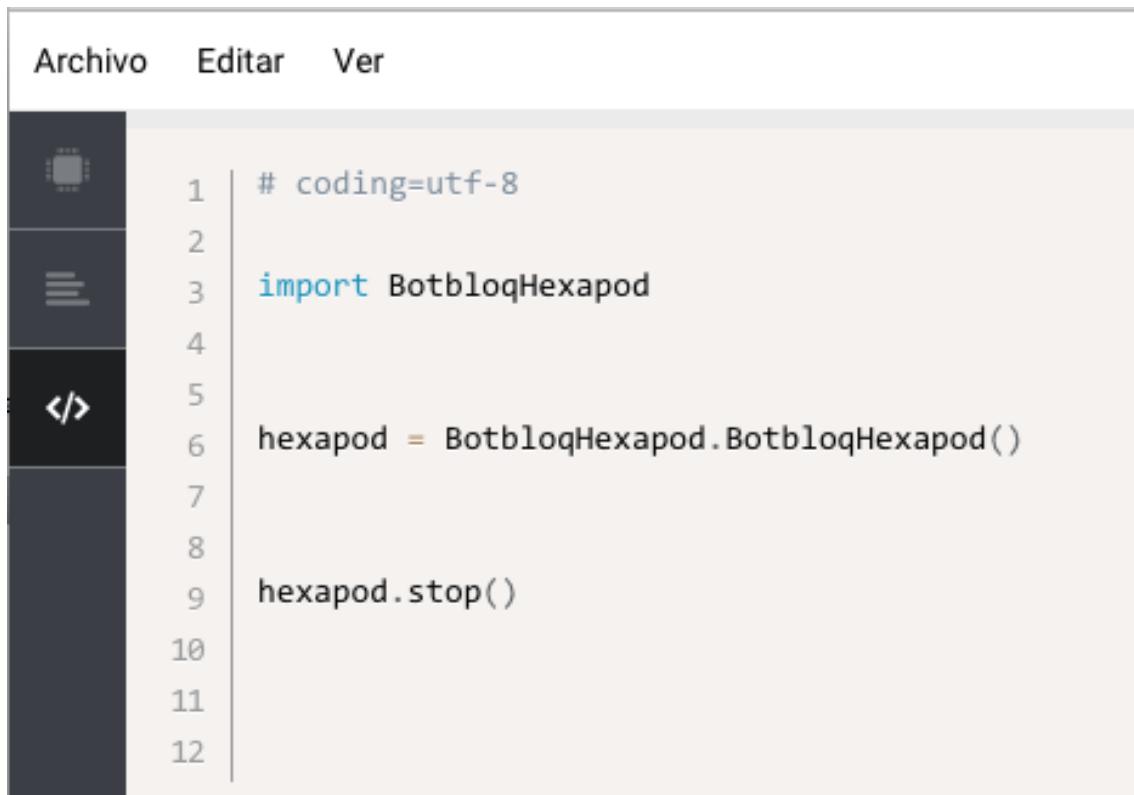
}

1.2.5.3.- Bloque “*botbloqHexapodStop*”

Con este bloque podremos hacer que se pare el robot y de esa forma detener cualquiera de los movimientos iniciados por los otros bloques.

Detener

Este bloque generará un código de este tipo:



```
Archivo Editar Ver

# coding=utf-8

import BotbloqHexapod

hexapod = BotbloqHexapod.BotbloqHexapod()

hexapod.stop()
```

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "botbloqHexapodStop",
  "connectors": [
    {
      "type": "connector--top",
```

65

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

```
"accept": "connector--bottom"
},
{
  "type": "connector--bottom",
  "accept": "connector--top"
}
],
"bloqClass": "bloq-botbloq-hexapod-stop",
"content": [
  [
    {
      "alias": "text",
      "value": "Detener"
    }
  ]
],
"code": "",
"python": {
  "libraries": [
    "BotbloqHexapod"
  ],
  "needInstanceOf": [
    {
      "name": "hexapod",
      "type": "BotbloqHexapod"
    }
  ],
  "codeLines": [
    {
      "code": "hexapod.stop()"
    }
  ]
}
}
```

1.3.- Adaptación de la herramienta actual para su uso con los robots de Botbloq

Durante el desarrollo y prueba de la aplicación Bitbloq en centros de toda España y de fuera, nos dimos cuenta de que un modelo 100% online no era suficiente, tampoco un modelo híbrido de visitar una web y compilar y cargar en local.

Eso nos llevo a un desarrollo offline de la aplicación reutilizando muchísimos de los elementos de la web, de tal forma que pudimos hacer una solución offline

100% (<https://github.com/bq/bitbloq-offline>) que pudiera funcionar sin problemas.

Una vez desarrollada la hemos actualizado para poder dar soporte a los robots de Botbloq, para poder ser llevada a centros para su validación, así como al ser una aplicación offline, su actualización y desarrollo es mucho más rápido.

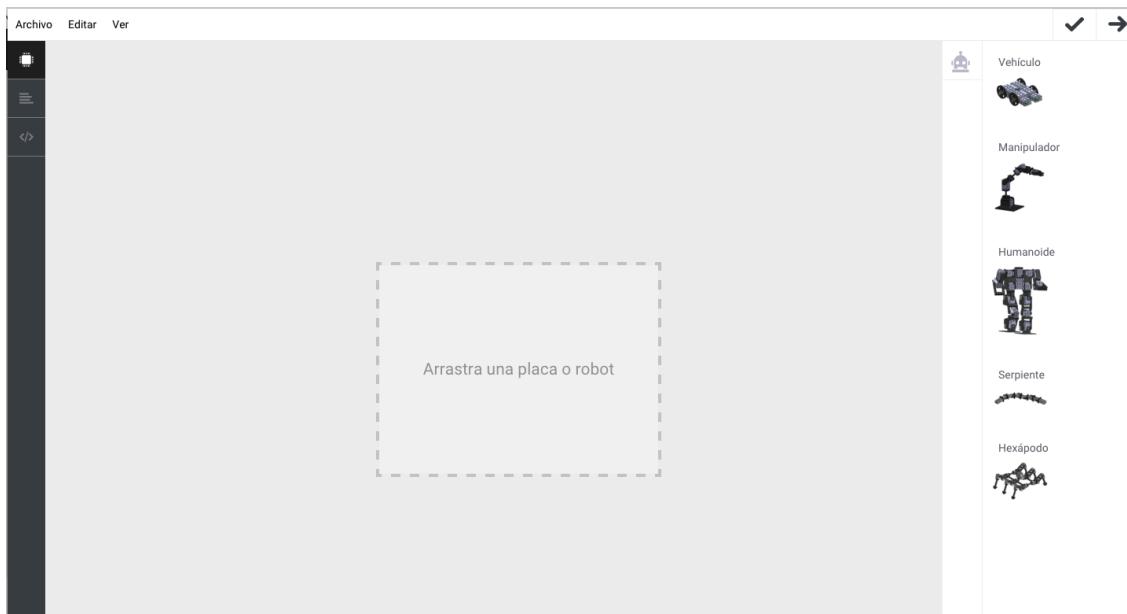
La herramienta se creó mediante la herramienta Electron (<http://electron.atom.io/>) para poder reutilizar todos los elementos, interfaces y librerías del desarrollo online. Hablaremos muy breve de esta herramienta ya que en su propia web está bien documentada. Electron es de forma muy resumida una versión portable del navegador Google Chrome con una versión portable de NodeJS, lo cual permite crear aplicaciones de escritorio multiplataforma, que son renderizadas en un navegador web y que por debajo al tener NodeJS da acceso a mucha de la funcionalidad del ordenador, como acceso al sistema de archivos, al puerto serie, poder lanzar otras aplicaciones como por ejemplo Web2board.

La aplicación consta de la pantalla de desarrollo de proyectos, con la selección de elementos hardware, software y la vista de código. De una versión modificada del programa web2board, para ser ejecutada en local utilizando la herramienta Platformio.org (<http://platformio.org/>), que ha sido empaquetada de forma portable por el equipo de Botbloq para esta aplicación.

Se usa el mismo formato de fichero, para poder importar y exportar proyectos de ambas plataformas, la online y la offline, para que los usuarios puedan trabajar sus proyectos sin importar el entorno.

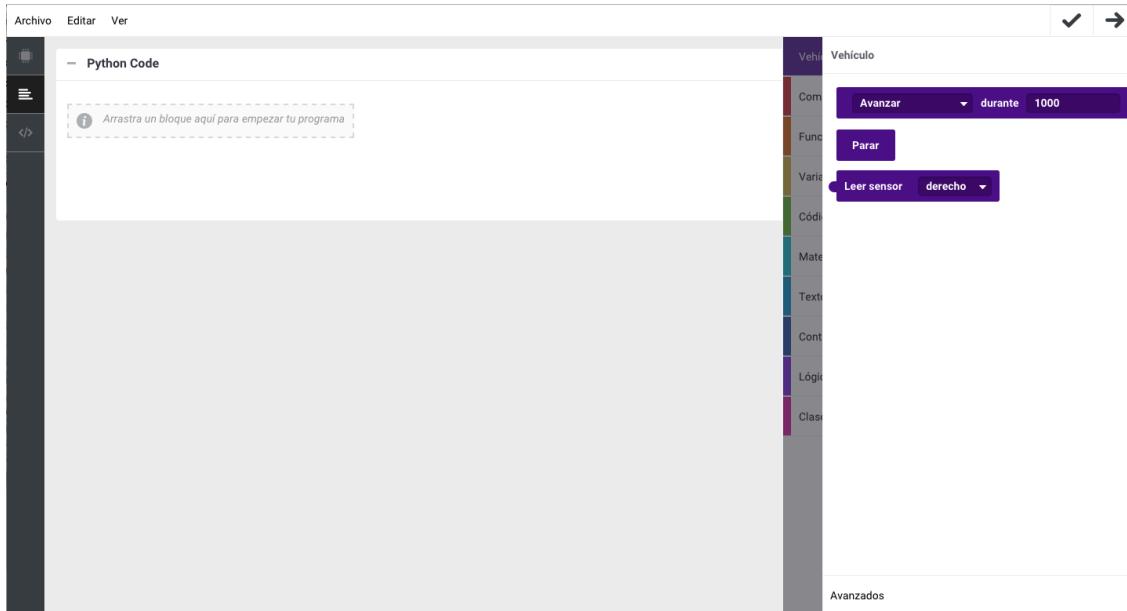
Con esta tecnología y la aplicación desarrollada y probada, nos lanzamos a modificarla para añadir los nuevos robots (Vehículo, Manipulador, Humanoide, Serpiente, Hexápodo).

Empezamos añadiendo los robots en la parte de hardware:



Después actualizamos la librería de bloques para tener todos los bloques nuevos que hemos generado para estos robots. Una vez actualizada, modificamos la ventana de software para que al estar seleccionado un robot salieran sus bloques.

Vehículo:

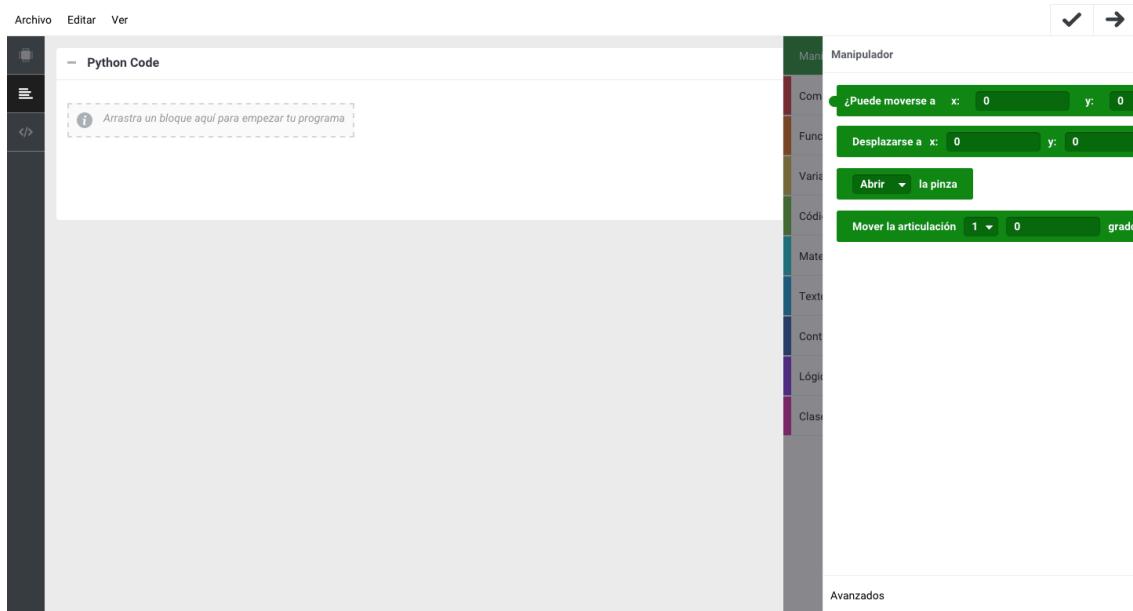


Manipulador:

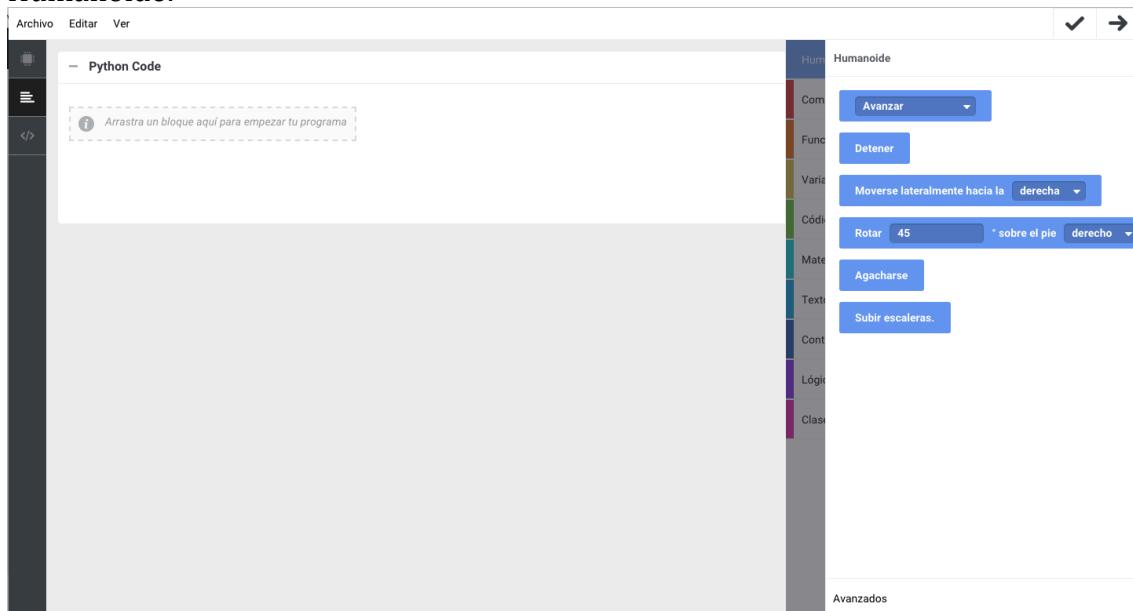
68

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289

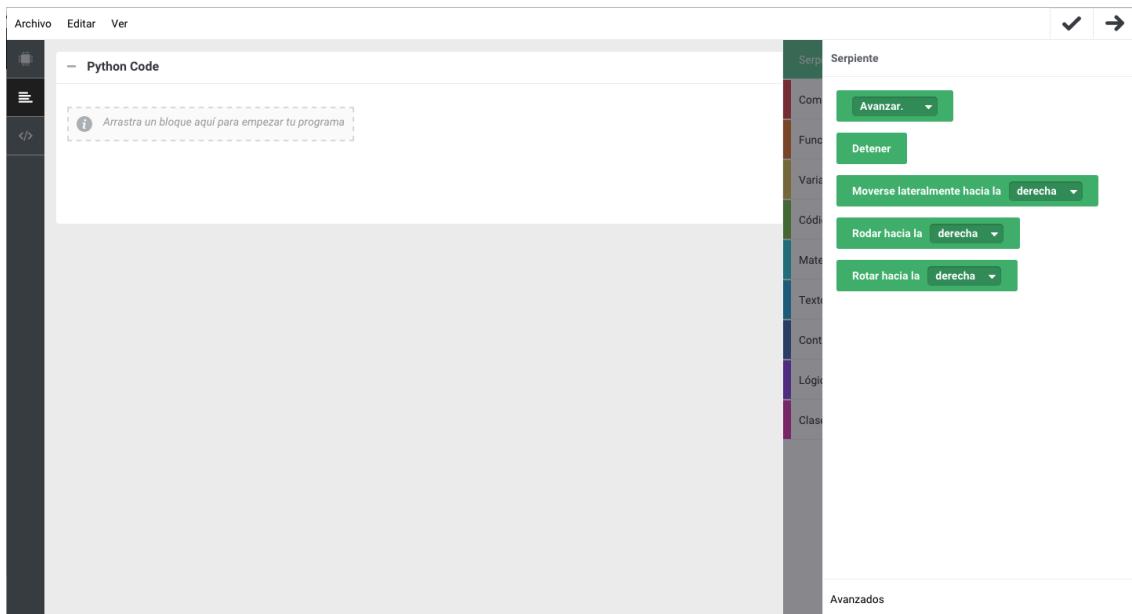
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020



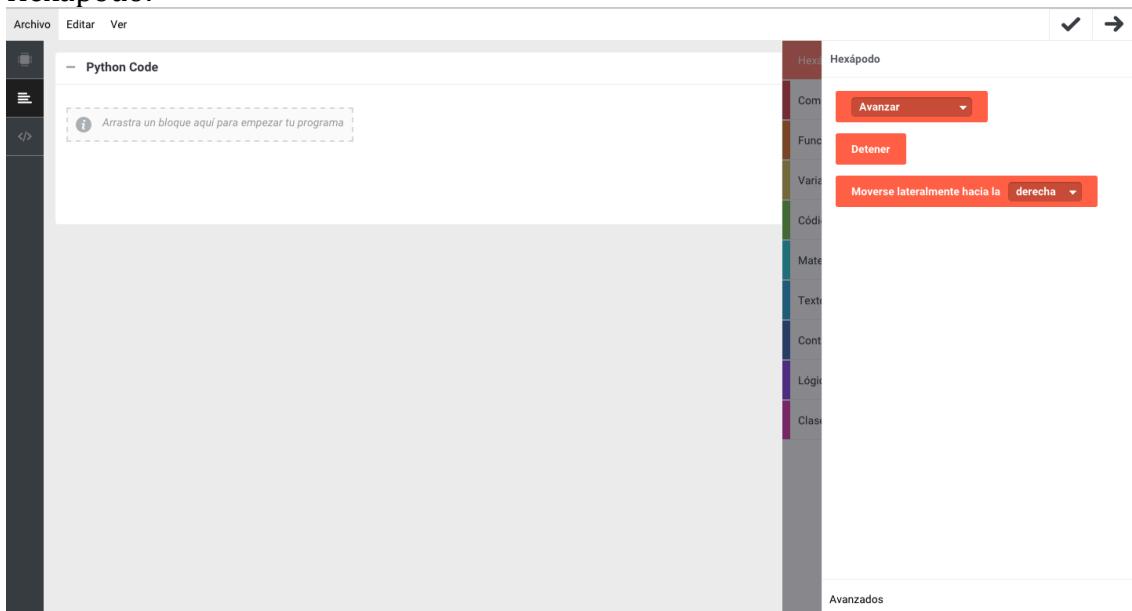
Humanoide:



Serpiente:



Hexápedo:



Despues se crean clases en python para que el código sea lo más legible posible, esas clases serán una por cada robot, y encapsularan el código python interno. Estarán situadas en al ruta “app/res/botbloq/”, con estos nombres:

- “BotbloqVehicle.py”
- “BotbloqManipulator.py”
- “BotbloqHuman.py”
- “BotbloqSnake.py”
- “BotbloqHexapod.py”

Ejemplo:

70

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

--Clase BotbloqVehicle:

```
import mraa
import time
import ctypes

class BotbloqVehicle():

    def __init__(self):
        self.x = mraa.I2c(0)

    def moveWheel(self, address, speed):
        self.x.address(address)
        self.x.writeByte(ctypes.c_uint8(182).value)
        time.sleep(0.001)
        self.x.writeByte(ctypes.c_uint8(int(speed)).value)
        time.sleep(0.001)

    def move(self, delay, speed, direction):
        velocidad_avance = int(90.0 + float(velocidad) * 90 / 100)
        velocidad_retroceso = int(90.0 - float(velocidad) * 90 / 100)

        if (direction == "FORWARD"):
            moveWheel(0x04, velocidad_avance)
            moveWheel(0x06, velocidad_avance)
            moveWheel(0x03, velocidad_retroceso)
            moveWheel(0x05, velocidad_retroceso)
        elif (direction == "BACKWARD"):
            moveWheel(0x04, velocidad_retroceso)
            moveWheel(0x06, velocidad_retroceso)
            moveWheel(0x03, velocidad_avance)
            moveWheel(0x05, velocidad_avance)
        elif (direction == "TURN_LEFT"):
            moveWheel(0x04, velocidad_retroceso)
            moveWheel(0x06, velocidad_retroceso)
            moveWheel(0x03, velocidad_retroceso)
            moveWheel(0x05, velocidad_retroceso)
        elif (direction == "TURN_RIGHT"):
            moveWheel(0x04, velocidad_avance)
            moveWheel(0x06, velocidad_avance)
            moveWheel(0x03, velocidad_avance)
            moveWheel(0x05, velocidad_avance)
        time.sleep(delay)

    def wait(self, delay):
        moveWheel(0x04, 90)
```

```
moveWheel(0x06, 90)
moveWheel(0x03, 90)
moveWheel(0x05, 90)

time.sleep(delay)

def stop(self):
    moveWheel(0x04, 90)
    moveWheel(0x06, 90)
    moveWheel(0x03, 90)
    moveWheel(0x05, 90)

exit()

def readIRSensor(self, side):
    if side == "LEFT":
        address = 0x32
    else:
        address = 0x33
    self.x.address(address)
    return self.x.readByte()
```

Fin del ejemplo--

Una vez realizado modificamos los botones de acción, en este caso Verificar y Cargar. Para el botón de compilar y verificar, lo que se hace es generar el código python, añadir las clases externas necesarias y generar un fichero con el nombre del proyecto y extensión .py completo, una vez generado ejecutamos comandos de Python para verificar el código tal y como lo hace el propio Python (python -m py_compile “pythonCode/program.py”).

La carga al robot es más compleja, ya que a diferencia de cómo se hace para las placas Arduino, que es usando el puerto serie del ordenador conectado a una placa y cargando el programa con uno de los protocolos (stk500v1, stk500v2, etc...), en este caso se accederá por scp para subir el fichero a la placa. Tanto el robot como el ordenador ejecutando Botbloq deben de estar en la misma red y se tiene que añadir la contraseña y la ip del robot dentro de la herramienta.

2.- Validación de la herramienta

72

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



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

La validación de la herramienta se complementa con la mostrada en el entregable 4.7 "Informe sobre resultados de validación", la herramienta también ha sido validada mediante talleres y campamentos, tanto a familias como a profesores.

Además es una herramienta aceptada por gran cantidad de centros, tanto Europeos como de fuera de Europa ya que ha pasado 2016 con más de 120.000 usuarios (es una cifra de los usuarios con conexión a internet ya que al sacar una versión offline, no se puede contabilizar el impacto total que será mayor).

Como parte de esa validación se han dispuesto varias vías para recoger las sugerencias de usuarios:

- Un foro: Disponible en <http://bitbloq.bq.com/#/forum>, donde se potencia la creación de una comunidad para que los usuarios puedan ayudarse entre si, y que es revisada por técnicos de Bitbloq para resolver dudas técnicas, ayudar a usuarios a incluir nuevas placas y ofrecer tanto información como consejos para las clases.
- Línea telefónica: Disponible para que puedan llamar y se les ayude con cualquier problema técnico que puedan tener.
- Email: Disponibles emails en distintos idiomas para que puedan escribir.
- Formulario de contacto en la herramienta: En distintos puntos de la herramienta pueden encontrar botones de ayuda que les muestran un formulario que es respondido por nuestro servicio técnico.

2.1.- Talleres para profesores, niños y familias.

2.1.1.-Clases Extraescolares para niños en Escuela La Maquinista.

Se realizaron clases extraescolares para niños de 7 a 10 años del 11 de Enero de 2016 hasta el 20 de Junio de 2016 , se crearon 4 grupos realizando 1 hora por grupo a la semana. Fueron un total de 41 asistentes. Fueron realizados a través de la empresa colaboradora TresDosU. Anexo de su oferta formativa en colegios (ExtraescolarsTresDosU_16-17.pdf)

2.1.2.- Clases Extraescolares para niños en Escuela Parc de la Ciutadella

Se realizaron clases extraescolares para niños de 7 a 10 años del 11 de Enero al 30 de Mayo de 2016, se creó un grupo de 1 hora a la semana. Fueron un total de 11 niños. También realizados por la empresa colaboradora TresDosU.

2.1.3.- Proyecto fundación Mapfre

Una de las misiones de la Fundación Mapfre es concienciar a los jóvenes de la importancia de la seguridad vial y prevención a través de la tecnología. Es por eso que junto a BQ se inició una colaboración, en la que se desarrolló un proyecto de

creación de contenidos, talleres con familias y formación al profesorado en los que a través de la robótica y la impresión 3D se busca introducir la educación en seguridad vial de modo ameno y atractivo a través de la tecnología.

Con esto se demostró que la herramienta es lo suficientemente intuitiva y que puede ser usada en ámbitos más amplios y no solo para enseñar a programar, una vez que los niños ya saben programar, esta herramienta les permite realizar proyectos y enseñar de una forma distinta. Generando mucho interés entre los niños y motivándoles de una forma muy buena.

El proyecto se estructura en torno a cuatro líneas de trabajo:

- Desarrollo de contenidos atractivos sobre educación vial para actividades de tecnología curriculares y extracurriculares de carácter lúdico.
- Desarrollar entre los niños competencias tecnológicas que en el futuro incentiven proyectos de emprendimiento social sensibilizados con la seguridad vial y la prevención.
- Formar al profesorado de modo que pueda introducir en el aula dichos proyectos de forma curricular.
- Realizar actividades para sensibilizar a los niños y familias sobre las cuestiones claves de la seguridad vial y la prevención.

Prácticamente todos los contenidos se desarrollan alrededor del modelo de bloques y con la herramienta Bitbloq.

Se desarrollaron 4 proyectos de educación vial con robótica usando Bitbloq:

2.1.3.1.- *Luz verde a tu ciudad: cómo ser un buen peatón*

Este proyecto puede ser encontrado aquí: <http://diwo.bq.com/luz-verde-a-tu-ciudad-como-ser-un-buen-peaton/>, los documentos generados están adjuntos al entregable en la carpeta de Proyectos Fundación Mapfre.

Programando nuestro semáforo

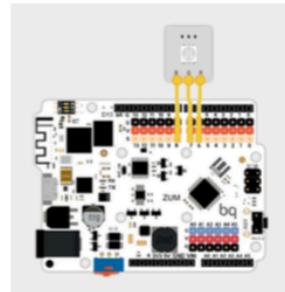
Vamos a programar un semáforo con dos luces, una verde utilizando un LED normal y otra roja, utilizando el LED RGB.

El LED RGB, está formado por tres luces distintas: una roja, una verde y una azul, cuya combinación hace que se obtengan muchos otros colores.

Para programar nuestro LED RGB, lo que necesitamos conocer es lo siguiente:

- **¿Cómo lo conecto?**

Este componente tiene 4 cables. Deberemos conectar el cable negro o GND a cualquiera de los pines digitales de color negro, correspondientes a tierra, y los otros cables a los pines digitales de señal, los mismos en los que los hayamos conectado en Bitbloq.



- **¿Cómo lo apago?**

Para que permanezca apagado, deberemos utilizar el siguiente bloque y poner todos los valores a 0.

Encender el led RGB led_RGB_0 con un valor de rojo de 0, un valor de verde de 0 y un valor de azul de 0

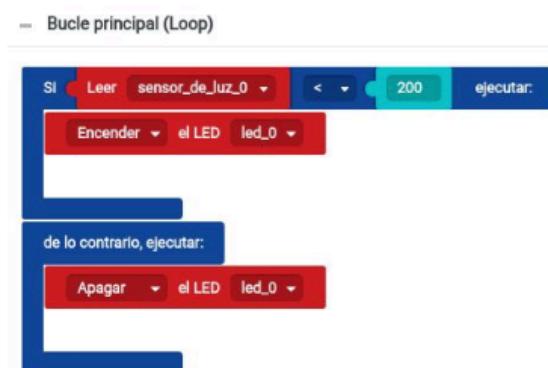
- **¿Puedo crear otros colores?**

Utilizando el bloque anterior pero asignando valores correspondientes al color. Para consultar los valores que tiene un color podemos utilizar el siguiente enlace: http://rapidtables.com/web/color/RGB_Color.htm.

Imagen de ejemplo del proyecto

2.1.3.2.- Patinadores y skaters, ¿los nuevos peatones?

Este proyecto puede ser encontrado aquí: <http://diwo.bq.com/patinadores-y-skaters-los-nuevos-peatones/>, los documentos generados están adjuntos al entregable en la carpeta de Proyectos Fundación Mapfre.



Detectando si hay o no luz

Para este ejercicio, conecta a tu placa el sensor de luz y un LED.

Utilizaremos el bloque condicional Si... ejecutar..., de manera que, si el sensor de luz detecta un valor menor de 200, encenderá el LED, de lo contrario, lo apagará.

Carga el programa en tu placa y comprueba cómo el LED permanece apagado cuando hay luz y se enciende cuando lo tapamos con la mano.

bq

4

Fundación MAPFRE

Programando el radar de tramo

Para comenzar con la programación del radar, necesitaremos los siguientes componentes: 2 LED y 2 sensores de luz (LDR). Para poder realizar la programación, deberemos conocer cuál va a ser la función que van a tener estos componentes.

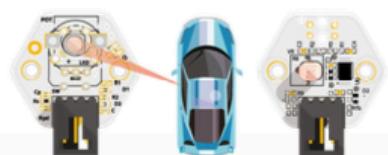


Imagen de ejemplo del proyecto

2.1.3.3.- Desplazándose a pedales: los ciclistas

Este proyecto puede ser encontrado aquí: <http://diwo.bq.com/ciclistas-desplazandose-a-pedales/>, los documentos generados están adjuntos al entregable en la carpeta de Proyectos Fundación Mapfre.

76

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

¡A programar!

Antes de empezar, es importante que conozcamos cómo funciona Bitbloq. Para ello, vamos a crear un programa que encienda un LED.

¿Cómo conectar el LED a la placa?, ¿y en Bitbloq?

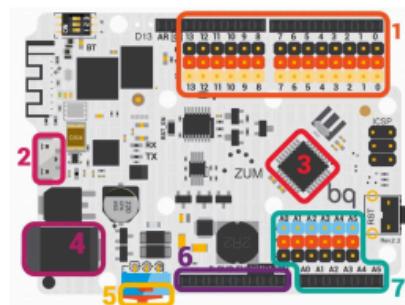
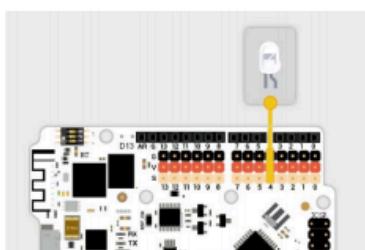
Antes de conectar un componente a la placa, deberemos pensar si es un componente **analógico**, aquellos que pueden tener más de dos valores, o **digital**, aquellos con solo dos valores (0 o 1).

En nuestro caso, el LED es un componente digital ya que solo tiene dos estados, o encendido (1) o apagado (0). Por ello, deberemos conectarlos a los pines digitales de nuestra placa controladora.

¿Cuáles son los pines digitales de la placa controladora?

Son los que corresponden con el número 1 de la siguiente imagen.

* Debemos intentar no utilizar los pines digitales 0 y 1 de la placa, ya que a través de estos dos pines, la placa se comunica con el ordenador y otros aparatos por lo que en ocasiones pueden dar problemas.



Una vez que tenemos el LED conectado a la placa, deberemos ir a Bitbloq y en la pestaña *Hardware*, arrastrar nuestra placa y el LED, conectándolo al mismo

Imagen de ejemplo del proyecto

2.1.3.4.- Circulando en coche. ¡Ponte el cinturón y arrancamos!

Este proyecto puede ser encontrado aquí: <http://diwo.bq.com/circulando-en-coche-ponte-el-cinturon-y-arrancamos/>, los documentos generados están adjuntos al entregable en la carpeta de Proyectos Fundación Mapfre.

- **¿Cómo lo conecto a la placa?**

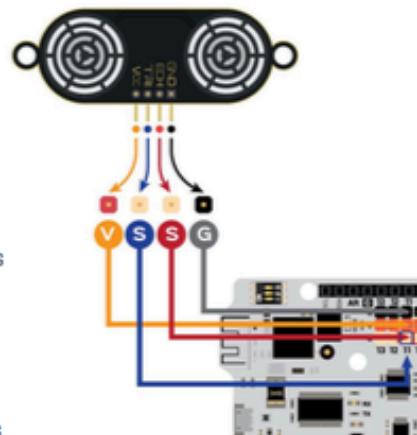
Como el sensor de ultrasonidos nos permite saber muchos valores distintos, puedes pensar que será analógico...Pero no, en realidad el sensor no me dice la distancia, sólo manda una señal cuando ha detectado, y el programa calcula la distancia. Así que su estado es "no estoy midiendo" (0) o "sí estoy midiendo" (1). Por esto, lo conectamos a los pines digitales.

El componente tiene 4 cables, que son para:

- VCC: se conecta a la tensión (Voltage) de la placa (pines rojos de la placa).
- GND: se conecta a la masa (Ground) de la placa (pines negros).

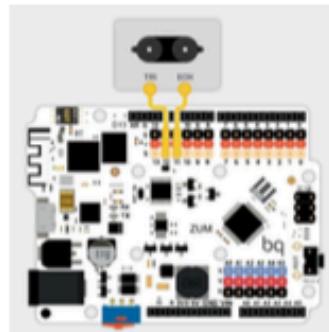
VCC y GND son como la pila del sensor, le dan energía.

- TRG y ECH: son señales, así que se conectan a los pines blancos o amarillos.



- **¿Y en Bitbloq?**

Como puedes ver en la imagen de la conexión en Bitbloq, lo que le importa a la placa son las señales (cables TRG y ECH), por lo que debemos hacer coincidir los pines donde hemos conectado estos cables a la placa física con los de la placa de bitbloq.



- **¿Cómo lo programo?**

Para leer el sensor con Bitbloq se usa el siguiente bloque, que nos dirá la distancia en centímetros hasta el objeto que haya delante.

Leer ultrasonidos_0 ▾

Pero para poder ver los valores que va leyendo, usaremos el puerto serie, que nos permitirá ver la distancia que mide el sensor de ultrasonidos en la pantalla del ordenador. Para ello, añadimos el componente a la placa y ponemos en la parte *Bucle principal (Loop)* los siguientes bloques:



Imagen de ejemplo del proyecto

2.1.4.- Campus tecnológicos de BQ

Desarrollados en verano de 2016 y en Navidades de 2016, estos campus se han realizado durante Verano y Navidad.

Video explicativo: https://www.youtube.com/watch?v=GYuNL_fw9yk

Web de los campus: <https://www.bq.com/es/campus-tecnologico>

Organizados para niños entre 8 y 14 años y alineados con la visión holística y sistemática planteada en el modelo CCR de la Comisión Europea, para integrar el potencial de la tecnología e innovar en la enseñanza, nos ha ayudado a validar la herramienta con proyectos tecnológicos robóticos durante los meses de Diciembre y Agosto. Han sido organizados en colaboración con colegios de la Comunidad de Madrid.