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

MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD

CDTI Centro para el Desarrollo Tecnológico Industrial

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

ENTREGABLE E.5.3.1. Módulo de programación de dispositivos móviles Android para kits genéricos de robótica.

RESUMEN DEL DOCUMENTO

Se han creado nuevos bloques en Bitbloq para interactuar con el móvil. Estos bloques se encargarán de generar el código necesario para que las placas puedan 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

Índice

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

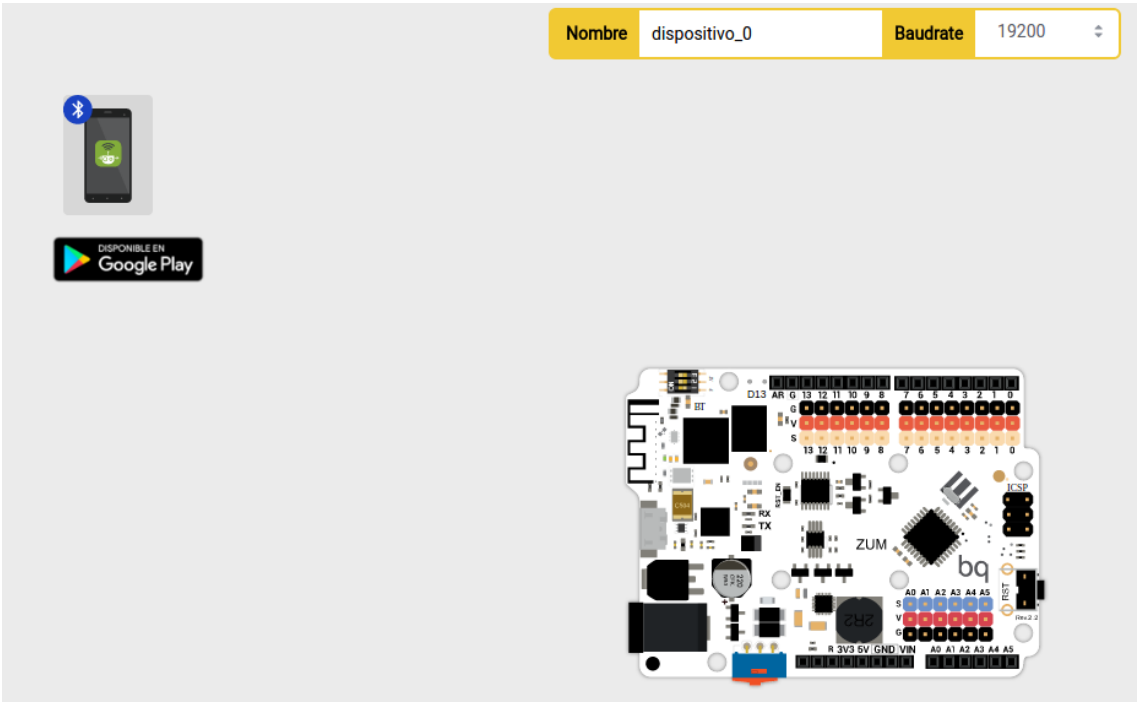
1.- Comunicación con la placa

La comunicación de la placa con la aplicación móvil se hará mediante Bluetooth.

Por tanto, el objetivo de la placa es mandar vía Bluetooth un comando y unos parámetros que indiquen a la aplicación qué debe hacer. Del mismo modo, en algunos casos la placa se quedará esperando a que el móvil le envíe los datos solicitados.

Las placas con las que se han realizado las pruebas son 3:

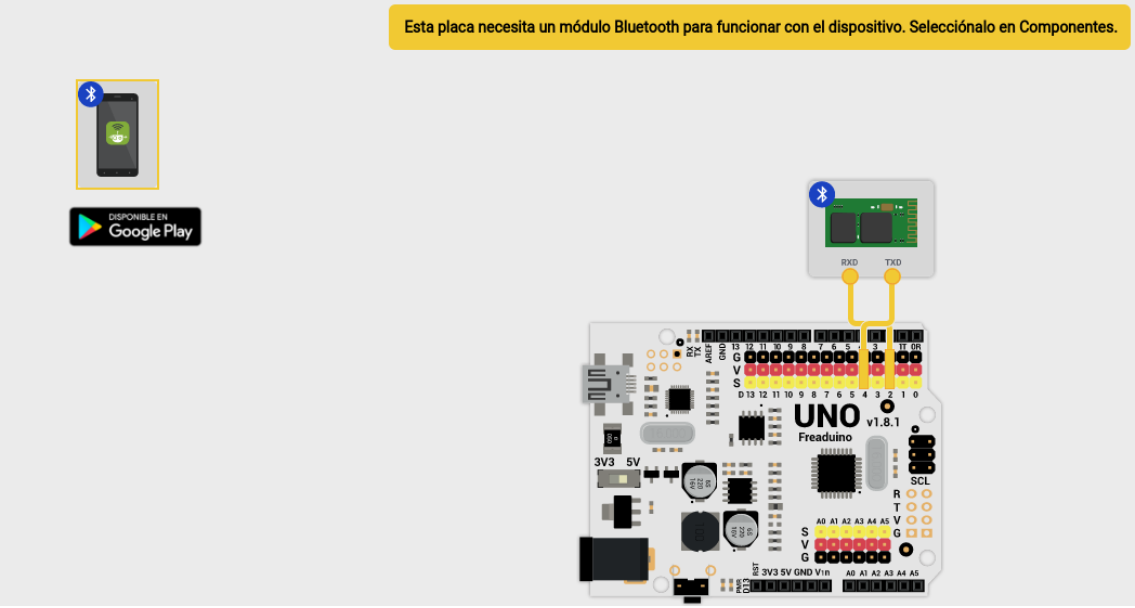
- **BQ ZUM** (https://www.bq.com/es/zum-kit)



La placa BQ ZUM posee el módulo de Bluetooth integrado, según se puede leer en las especificaciones. Para enviar y recibir de modo correcto con este módulo, se deberá configurar el baudrate a 19200 como se puede observar en la figura.

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

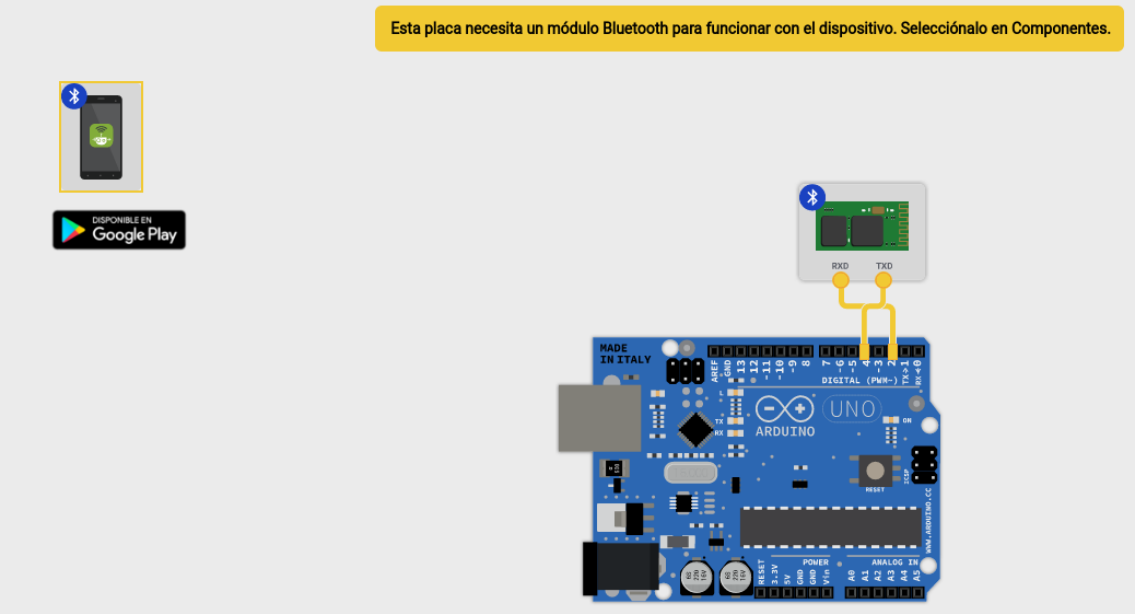
- **FreaduinoUno**
  (http://www.elecfreaks.com/wiki/index.php?title=Freaduino_UNO)



En este tipo de placas, el Bluetooth no viene integrado y es necesario conectar un módulo de Bluetooth. De cara a las pruebas, se utiliza el módulo HC-05 (http://www.prometec.net/bt-hc05/) y se configura el baudrate a 38400 para que se envíen y se reciban correctamente los datos vía Bluetooth.

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

- **ArduinoUNO** (https://www.arduino.cc/en/Main/ArduinoBoardUno)



De modo similar al caso anterior, no existe Bluetooth integrado, por lo que se deberá conectar el módulo de Bluetooth HC-05 y se configurará del mismo modo el baudrate a 38400 para un perfecto funcionamiento. Tanto en este caso como en el anterior, se informa al usuario de que tiene que conectar el módulo para que la placa pueda comunicarse con la aplicación instalada en el dispositivo móvil.

Como se puede observar, en ambos casos se presenta un icono (o *badge*) que permite consultar el enlace de la aplicación móvil Bitbloq Connect.

A continuación, se hablará del desarrollo de la funcionalidad incluyendo la definición e implementación de los bloques necesarios para la interacción dispositivo móvil - placa.

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.- Desarrollo

Para el desarrollo se crearon nuevos bloques en Bitbloq de cara a interactuar con el móvil mediante Bluetooth.

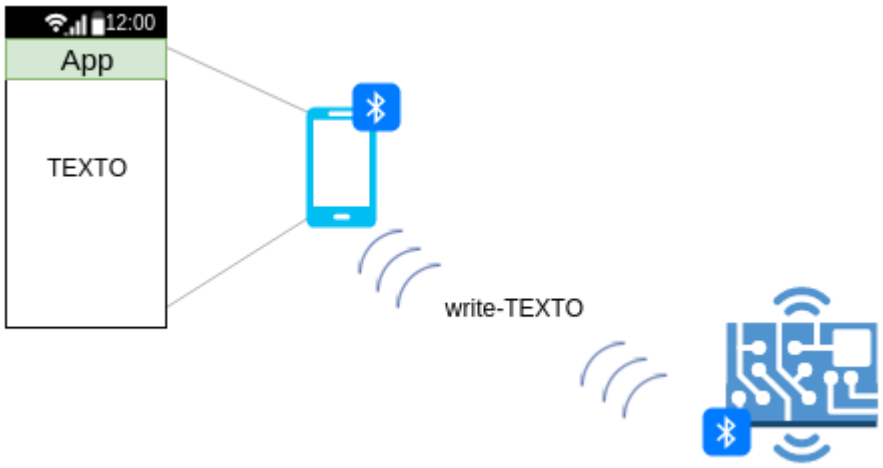
De forma general, el objetivo de los bloques se puede dividir en tres casos, dependiendo de cuál sea el bloque:

- La placa le mande al móvil un comando y el móvil realice una acción.
- El móvil envíe datos y la placa se quede esperando a recibirlos.
- Una combinación de los dos anteriores: la placa envíe un comando solicitando unos datos al dispositivo móvil y la placa le conteste con los mismos en el momento en el que estén disponibles.

A continuación se describirán los bloques creados para cumplir cada una de las necesidades.

2.1.- Móvil recibe texto de la placa.

El principal objetivo de esta funcionalidad es mostrar en la pantalla del móvil el texto enviado por la placa.



Para ello, se ha creado el bloque "`phoneSendText`".

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

En el hueco al lado de Mostrar en la pantalla se le añadirá el texto que queramos mandarle a la placa, ya sea mediante un bloque tipo String o mediante un bloque tipo variable asociado a una variable que posea el texto. Por ejemplo, si se quiere enviar el texto 'hola', se conectará un bloque String, obteniendo el siguiente resultado:

El bloque anterior, si es colocado en el loop, generará un código de este tipo:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >

/*** Global variables and function definition ***/
bqSoftwareSerial dispositivo_0(0, 1, 9600);
dispositivo_0.println(String("toggleFlashlight-") + String());
/*** Setup ***/
void setup() { }
/*** Loop ***/
void loop() {
dispositivo_0.println(String("write-") + String("hola"));
}
```

Como se puede observar, lo que hace el bloque es enviar por Bluetooth un comando "**write-**" seguido del texto que se quiere mostrar por pantalla.

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "phoneSendText",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    },
    {
      "type": "connector--input",
      "accept": "connector--output",
      "acceptType": [
        "all"
```


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

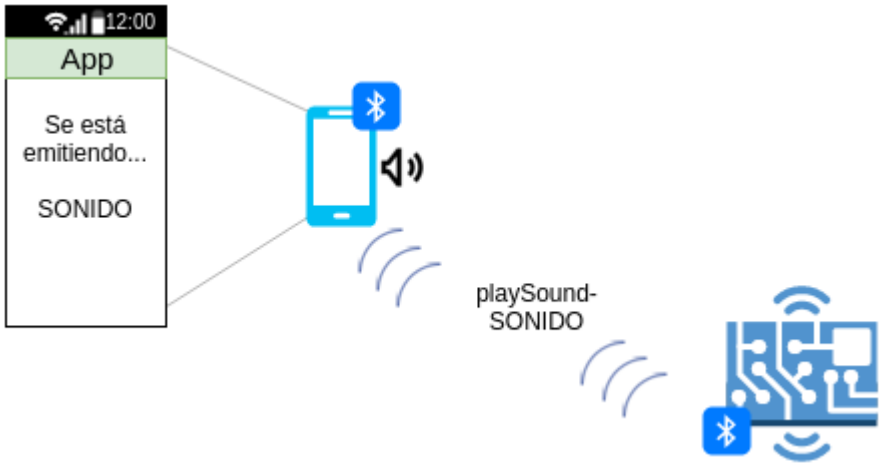
```
      ],
      "suggestedBloqs": [
        "string",
        "number",
        "selectVariable"
      ],
      "name": "94206051-51a7-446f-9da2-88b0fba3a5af"
    }
  ],
  "bloqClass": "bloq-phone-write",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-phone-write-show"
      },
      {
        "bloqInputId": "DATA",
        "alias": "bloqInput",
        "acceptType": [
          "all"
        ],
        "suggestedBloqs": [
          "string",
          "number",
          "selectVariable"
        ],
        "name": "94206051-51a7-446f-9da2-88b0fba3a5af"
      },
      {
        "alias": "text",
        "value": "bloq-phone-screen"
      },
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      }
    ]
  ],
  "code": "{PHONE}.println(\"write-\" + {DATA});",
  "arduino": {
    "includes": [
      "BitbloqSoftwareSerial.h"
    ],
    "code": "{PHONE}.println(String(\"write-\")+String({DATA}));"
  },
  "python": {
    "codeLines": [
      {
        "code": "Bloque \"phoneSendText\" no preparado para generar
código python"
      }
    ]
  }
}
```


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

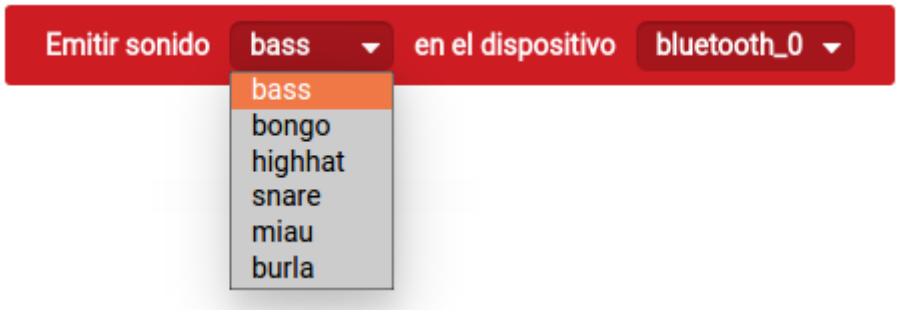
2.2.- Móvil emite sonido.

El principal objetivo de esta funcionalidad es que el móvil emita el sonido que le indica la placa.

Para ello, la placa le enviará el nombre del audio y será la aplicación móvil la que tenga en su interior los ficheros mp3 asociados al mismo.



Para ello, se ha creado el bloque "`phoneEmitSound`".



En el *dropdown* el usuario elegirá el sonido a emitir (bass, bongo, highhat, snare, miau, burla).

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

El bloque anterior, si es colocado en el loop, generará un código de este tipo:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >

/*** Global variables and function definition ***/
bqSoftwareSerial dispositivo_0(0, 1, 9600);

/*** Setup ***/
void setup() { }

/*** Loop ***/
void loop() {
dispositivo_0.println(String("playSound-") + String("meow"));
}
```

Como se puede observar, lo que hace el bloque es enviar por Bluetooth un comando "**playSound-**" seguido del sonido que se quiere emitir en el móvil.

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "phoneEmitSound",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-phone-sound",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-phone-sounds"
      },
      {
        "id": "SOUND",
        "alias": "staticDropdown",
        "options": [
          {
            "label": "bloq-phone-sounds-bass-v1",
            "value": "\"bass\""
```


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

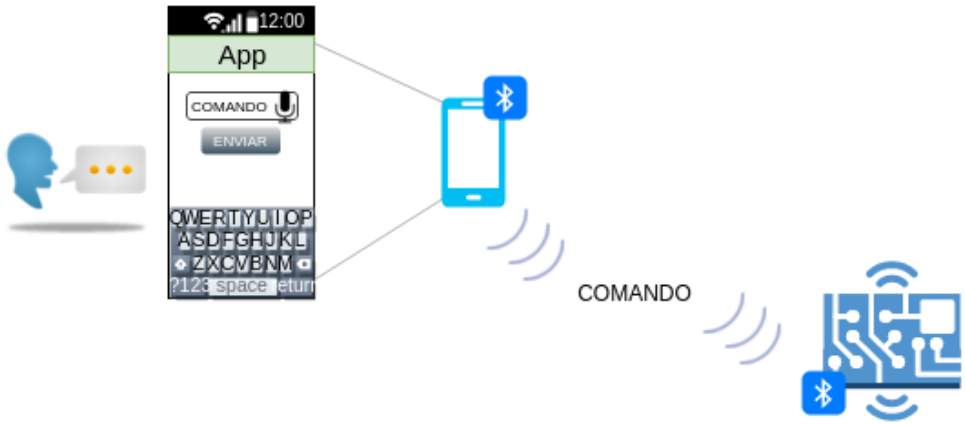
```
            },
            {
              "label": "bloq-phone-sounds-bongo-v1",
              "value": "\"bongo\""
            },
            {
              "label": "bloq-phone-sounds-highhat-v1",
              "value": "\"highhat\""
            },
            {
              "label": "bloq-phone-sounds-snare-v1",
              "value": "\"snare\""
            },
            {
              "label": "bloq-phone-sounds-meow-v1",
              "value": "\"meow\""
            },
            {
              "label": "bloq-phone-sounds-joke-v1",
              "value": "\"joke\""
            }
          ]
        },
        {
          "alias": "text",
          "value": "bloq-phone-sounds-device"
        },
        {
          "id": "PHONE",
          "alias": "dynamicDropdown",
          "options": "serialElements"
        }
      ]
    ],
    "code": "{PHONE}.println(\"playSound-\" +
{SOUND});delay(1000);",
    "arduino": {
      "includes": [
        "BitbloqSoftwareSerial.h"
      ],
      "code": "{PHONE}.println(String(\"playSound-
\")+String({SOUND}));delay(1000);"
    },
    "python": {
      "codeLines": [
        {
          "code": "Bloque \"phoneEmitSound\" no preparado para
generar código python"
        }
      ]
    }
}
```


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

2.3.- Móvil envía comando a la placa.

El principal objetivo de esta funcionalidad es recibir comandos desde el móvil.

Estos comandos llegarán a Bitbloq en formato texto, pero como se explicó en el módulo relacionado a la aplicación del móvil, serán introducidos tanto mediante el teclado como por voz.



Para ello, se ha creado el bloque "`phoneReceive`".

Si por ejemplo, asociamos ese bloque a una variable (de nombre comando) y la colocamos en el SETUP, el código generado sería el siguiente:

```
/*** Included libraries ***/
# include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >

/*** Global variables and function definition ***/
bqSoftwareSerial dispositivo_0(0, 1, 9600);
String comando = dispositivo_0.readString();

/*** Setup ***/
void setup() { }

/*** Loop ***/
void loop() {}
```


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

Como se puede observar en el código generado, lo que hace la placa es esperar a recibir datos del Bluetooth. Con esto conseguimos que sea muy fácil controlar la placa mediante comandos a voz (utilizando la funcionalidad del móvil de paso de voz a texto) de tal modo que podríamos, por ejemplo, controlar un led diciéndole ENCIENDE o APAGA de un modo muy sencillo.

Definición del bloque en formato JSON:

```
{
  "type": "output",
  "name": "phoneReceive",
  "connectors": [
    {
      "type": "connector--output",
      "accept": "connector--input"
    }
  ],
  "bloqClass": "bloq-phone-receive",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-phone-receive"
      },
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      }
    ]
  ],
  "code": "{PHONE}.readString()",
  "arduino": {
    "includes": [
      "BitbloqSoftwareSerial.h"
    ],
    "code": "{PHONE}.readString()"
  },
  "returnType": {
    "type": "simple",
    "value": "String"
  },
  "python": {
    "codeLines": [
      {
        "code": "Bloque \"phoneReceive\" no preparado para
generar código python"
      }
    ]
  }
```


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

```
}
```

2.4.- Placa interactúa con la linterna del móvil.

El principal objetivo de esta funcionalidad es interactuar con linterna (o flash) del móvil.

Cabe destacar que se ha añadido un delay de 500 ms debido a las especificaciones hardware habituales del flash en los dispositivos Android (no pueden encender/apagar la linterna a una frecuencia igual o superior a 500 ms).

Para ello se han creado 2 bloques que se explican a continuación:

2.4.1.- Encender la linterna del móvil

El objetivo de esta funcionalidad es encender la linterna (o flash) del móvil.

Para ello, se ha creado el bloque "`phoneTurnOnLight`".

Si por ejemplo, usamos el bloque con una intensidad 60% dentro del SETUP, el código generado sería el siguiente:

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

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >

/*** Global variables and function definition ***/
bqSoftwareSerial dispositivo_0(0, 1, 9600);

/*** Setup ***/
void setup() {
dispositivo_0.println(String("turnonFlashlight-"));
delay(500);
}
 /*** Loop ***/
void loop() {}
```

Como se puede apreciar, se le envía al móvil el comando "**turnonFlashlight-**".

Definición del bloque en formato JSON:

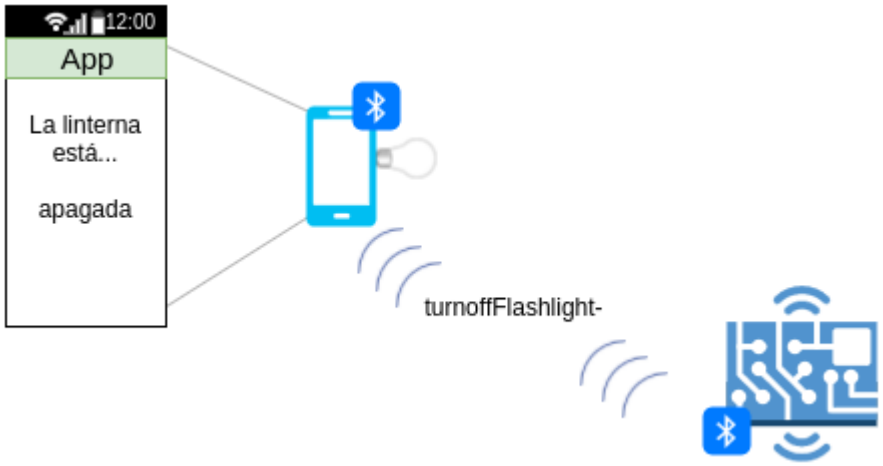
```
{
  "type": "statement",
  "name": "phoneTurnOnLight",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-phone-turnon-light",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-phone-turnon-light"
      },
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      }
    ]
  ],
  "code": "{PHONE}.println(\"turnonFlashlight-\");
delay(500);",
```


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

```
  "arduino": {
    "includes": [
      "BitbloqSoftwareSerial.h"
    ],
    "code": "{PHONE}.println(String(\"turnonFlashlight-\"));
delay(500);"
  },
  "python": {
    "codeLines": [
      {
        "code": "Bloque \"phoneTurnOnLight\" no preparado para
generar código python"
      }
    ]
  }
}
```

2.4.2.- Apagar la linterna del móvil

El objetivo de esta funcionalidad es apagar la linterna (o flash) del móvil.



Para ello, se ha creado el bloque "`phoneTurnOffLight`".

Si por ejemplo, usamos el bloque dentro del SETUP, el código generado sería el siguiente:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >
```


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

```
/*** Global variables and function definition ***/
bqSoftwareSerial dispositivo_0(0, 1, 9600);

/*** Setup ***/

void setup() {
dispositivo_0.println(String("turnoffFlashlight-"));
delay(500);
}

/*** Loop ***/
void loop() {}
```

Como se puede apreciar, se le envía al móvil el comando "**turnoffFlashlight-**".

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "phoneTurnOffLight",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-phone-turnoff-light",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-phone-turnoff-light"
      },
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      }
    ]
  ],
  "code": "{PHONE}.println(\"turnoffFlashlight-\");
delay(500);",
  "arduino": {
    "includes": [
      "BitbloqSoftwareSerial.h"
    ],
```


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": "{PHONE}.println(String(\"turnoffFlashlight-\"));
delay(500);"
  },
  "python": {
    "codeLines": [
      {
        "code": "Bloque \"phoneTurnOffLight\" no preparado
para generar código python"
      }
    ]
  }
}
```

2.5.- Placa publica un tweet.

El objetivo de este bloque es publicar un tweet (mensaje de menos de 140 caracteres) en una cuenta de Twitter.

En este caso se hace imprescindible definir dos bloques, uno de configuración y otro de publicación de Twitter, ya que el usuario tiene que identificarse para que el dispositivo móvil pueda publicar tweets en su nombre.

2.5.1. Configurar cuenta de Twitter

El objetivo de este bloque es enviar a la aplicación móvil la configuración de Twitter (las claves necesarias para identificar al usuario en Twitter).

Para ello, se ha creado el bloque "`phoneConfigTwitter`".

Es imprescindible colocar este bloque en el SETUP para configurar la aplicación antes de enviar datos. Si no se configura con las claves de aplicación de Twitter correspondientes, no funcionará el programa.

Si por ejemplo, usamos el bloque dentro del SETUP, el código generado sería el siguiente:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >

/*** Global variables and function definition ***/
```


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

```
bqSoftwareSerial dispositivo_0(0, 1, 9600);

/*** Setup ***/
void setup() { }

/*** Loop ***/
void loop() {
/*sendTwitterAppConfig*/
}
```

Como se puede observar, el código generado es un simple comentario en Arduino. Cuando el código se cargue a la placa, se sustituirá por el comando "**twitterConfig-**" y las claves de la aplicación de Twitter que el usuario haya configurado en el proyecto separadas por caracteres '/'.

El porqué de esta decisión es favorecer la compartición de proyectos y la publicación de los mismos (aumentando por tanto la reutilización del mismo y disminuyendo el tiempo de desarrollo). Si se configuraran las claves en el propio código, a la hora de compartir el proyecto o de hacerlo público serían accesibles por otros usuarios no propietarios del Twitter, y con ello podrían publicar mensajes en nombre del autor del proyecto.

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "phoneConfigTwitter",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    }
  ],
  "bloqClass": "bloq-twitter-config",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-twitter-config"
      }
    ]
  ],
  "code": "/*sendTwitterAppConfig*/",
  "arduino": {
```


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": "/*sendTwitterAppConfig*/"
  }
}
```

2.5.2. Publicar mensaje en Twitter configurado previamente

El objetivo de este bloque es publicar un mensaje de Twitter en una cuenta de Twitter previamente configurada.

Para ello, se ha creado el bloque "`phoneSendTweet`".

Si por ejemplo se quiere publicar el tweet "Hola mundo" el Twitter configurado previamente, se deberá colocar en SETUP tanto este bloque (con un bloque texto con 'Hola mundo' conectado) y previamente el bloque 2.5.1. Estos bloques generarán el siguiente código:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >


/*** Global variables and function definition ***/
bqSoftwareSerial dispositivo_0(0, 1, 9600);


/*** Setup ***/
void setup() {

/*sendTwitterAppConfig*/

dispositivo_0.println(String("twitterSend-") + String("hola
mundo"));
}


/*** Loop ***/
void loop() {}
```

Como se puede apreciar, se le envía al móvil el comando "**twitterSend-**" seguido del mensaje a publicar.

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

Definición del bloque en formato JSON:

```
{
  "type": "statement",
  "name": "phoneSendTweet",
  "connectors": [
    {
      "type": "connector--top",
      "accept": "connector--bottom"
    },
    {
      "type": "connector--bottom",
      "accept": "connector--top"
    },
    {
      "type": "connector--input",
      "accept": "connector--output",
      "acceptType": [
        "all"
      ],
      "suggestedBloqs": [
        "string",
        "selectVariable"
      ],
      "name": "c0f5da97-db3e-4469-a85e-2567fbce0174"
    }
  ],
  "bloqClass": "bloq-send-tweet",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-send-tweet"
      },
      {
        "bloqInputId": "TWEET",
        "alias": "bloqInput",
        "acceptType": [
          "all"
        ],
        "suggestedBloqs": [
          "string",
          "selectVariable"
        ],
        "name": "c0f5da97-db3e-4469-a85e-2567fbce0174"
      },
      {
        "alias": "text",
        "value": "bloq-from-device"
      },
```


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

```
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      }
    ]
  ],
  "code": "{PHONE}.println(\"twitterSend-\" + {TWEET});",
  "arduino": {
    "includes": [
      "BitbloqSoftwareSerial.h"
    ],
    "code": "{PHONE}.println(String(\"twitterSend-
\")+String({TWEET}));"
  },
  "python": {
    "codeLines": [
      {
        "code": "Bloque \"phoneSendTweet\" no preparado para
generar código python"
      }
    ]
  }
}
```


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

2.6.- Placa recibe datos de sensores

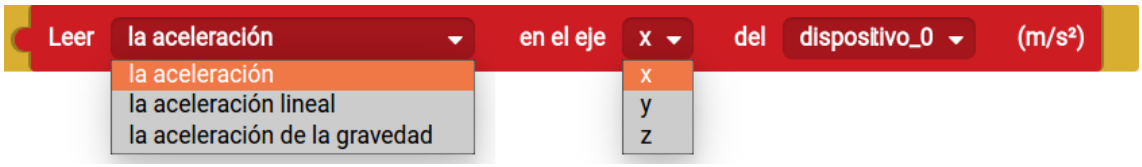
El objetivo de este conjunto de bloques es obtener los datos de los sensores disponibles en el dispositivo móvil.

Para ello, previamente se hizo un análisis de los sensores disponibles en el móvil, como se pudo observar en el documento E5.2.1. Una vez obtenidos los sensores compatibles con el dispositivo móvil, se analizó su utilidad, dando lugar a los siguientes bloques:

2.6.1. Placa recibe datos del sensor de aceleración

El objetivo de este bloque es obtener la aceleración del dispositivo, incluída la de la gravedad. El dispositivo móvil proporciona la aceleración en cada uno de los ejes.

Por ello, se ha creado el bloque "`phoneReadAccel`".



En el bloque se puede especificar de qué eje se quiere obtener la aceleración (x,y o z), así como la aceleración que se desea medir (aceleración global, lineal o de la gravedad).

Si por ejemplo, asociamos ese bloque a una variable (de nombre accel), seleccionamos la opción aceleración en el eje x y la colocamos en el bloque de variables, el código generado sería el siguiente:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >

/*** Global variables and function definition ***/
bqSoftwareSerial dispositivo_0(0, 1, 9600);

float getAcceleration(String axis, String message,
bqSoftwareSerial & phone) {
  phone.println(String(message) + String(axis));
  String data = "";
  while (data == "") {
    data = phone.readString();
  }
```


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

```
  return data.toFloat();
}

float accel = getAcceleration("x", "readAccel-",
dispositivo_0);

/*** Setup ***/
void setup() { }

/*** Loop ***/
void loop() {}
```

Como se puede observar en el código, el bloque genera una función `getAcceleration` que recibe como parámetros el eje en el cuál se desea obtener la aceleración ("x" en este caso) y el dispositivo del que se quiere medir ("dispositivo_0", de tipo `bqSoftwareSerial`).

La función `getAcceleration` envía al teléfono el comando pasado como segundo parámetro, unido al eje del que desea obtener la aceleración. Una vez hecho esto, se queda esperando a que el móvil le conteste con el dato y una vez esto ocurre lo devuelve convertido a `float`. Estos valores vendrán expresados en m/s^2.

Definición del bloque en formato JSON:

```
{
  "type": "output",
  "name": "phoneReadAccel",
  "connectors": [
    {
      "type": "connector--output",
      "accept": "connector--input"
    }
  ],
  "bloqClass": "bloq-phone-read-accel",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-read-read"
      },
      {
        "id": "MESSAGE",
        "alias": "staticDropdown",
        "options": [
          {
            "label": "bloq-phone-acceleration",
            "value": "\"readAccel-\""
          },
```


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

```
          {
            "label": "bloq-phone-lacceleration",
            "value": "\"readLAccel-\""
          },
          {
            "label": "bloq-phone-gravity",
            "value": "\"readGravity-\""
          }
        ]
      },
      {
        "alias": "text",
        "value": "bloq-phone-axis"
      },
      {
        "id": "AXIS",
        "alias": "staticDropdown",
        "options": [
          {
            "label": "x",
            "value": "\"x\""
          },
          {
            "label": "y",
            "value": "\"y\""
          },
          {
            "label": "z",
            "value": "\"z\""
          }
        ]
      },
      {
        "alias": "text",
        "value": "bloq-phone-of"
      },
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      },
      {
        "alias": "text",
        "value": "(m/s²)"
      }
    ]
  ],
  "code": "{PHONE}.readString()",
  "arduino": {
    "includes": [
      "BitbloqSoftwareSerial.h"
    ],
```


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

```
    "extraFunctionCode": "float getAcceleration(String axis,
String message, bqSoftwareSerial &
phone){phone.println(String(message)+String(axis));String
data=\"\";while(data==\"\"){data=phone.readString();}return
data.toFloat();}",
    "code": "getAcceleration({AXIS}, {MESSAGE}, {PHONE})"
  },
  "returnType": {
    "type": "simple",
    "value": "float"
  },
  "python": {
    "codeLines": [
      {
        "code": "Bloque \"phoneReadAccel\" no preparado para
generar código python"
      }
    ]
  }
}
```

2.6.2. Placa recibe datos del giroscopio.

El objetivo de este bloque es obtener el valor del giroscopio del dispositivo. El dispositivo móvil proporciona un valor por cada uno de los ejes.

Por ello, se ha creado el bloque "phoneReadGyroscope".

En el bloque se puede especificar de qué eje se quiere obtener el valor del giroscopio (x,y o z).

Si por ejemplo, asociamos ese bloque a una variable (de nombre gyros) y la colocamos en el bloque de variables, el código generado sería el siguiente:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >

/*** Global variables and function definition ***/

float getGyroscope(String axis, bqSoftwareSerial & phone) {
  phone.println(String("readGyros-") + String(axis));
  String data = "";
  while (data == "") {
    data = phone.readString();
```


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

```
  }
  return data.toFloat();
}

bqSoftwareSerial dispositivo_0(0, 1, 9600);

float gyros = getGyroscope("x", dispositivo_0);

/*** Setup ***/
void setup() { }

/*** Loop ***/
void loop() {}
```

Como se puede observar en el código, el bloque genera una función `getGyroscope` que recibe como parámetros el eje en el cuál se desea obtener el valor ("x" en este caso) y el dispositivo del que se quiere medir ("dispositivo_0", de tipo `bqSoftwareSerial`).

La función `getGyroscope` envía al teléfono el comando **"readGyros"** seguido del eje del que desea obtener el valor. Una vez hecho esto, se queda esperando a que el móvil le conteste con el dato y una vez esto ocurre lo devuelve convertido a `float`. Estos valores vendrán expresados en `rad/s`.

Definición del bloque en formato JSON:

```
{
  "type": "output",
  "name": "phoneReadGyroscope",
  "connectors": [
    {
      "type": "connector--output",
      "accept": "connector--input"
    }
  ],
  "bloqClass": "bloq-phone-read-gyros",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-phone-gyroscope"
      },
      {
        "id": "AXIS",
        "alias": "staticDropdown",
        "options": [
          {
            "label": "x",
```


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

```
            "value": "\"x\""
          },
          {
            "label": "y",
            "value": "\"y\""
          },
          {
            "label": "z",
            "value": "\"z\""
          }
        ]
      },
      {
        "alias": "text",
        "value": "bloq-phone-of"
      },
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      },
      {
        "alias": "text",
        "value": "(rad/s)"
      }
    ]
  ],
  "code": "",
  "arduino": {
    "includes": [
      "BitbloqSoftwareSerial.h"
    ],
    "extraFunctionCode": "float getGyroscope(String
axis,bqSoftwareSerial &
phone){phone.println(String(\"readGyros-
\")+String(axis));String
data=\"\";while(data==\"\"){data=phone.readString();}return
data.toFloat();}",
    "code": "getGyroscope({AXIS}, {PHONE})"
  },
  "returnType": {
    "type": "simple",
    "value": "float"
  },
  "python": {
    "codeLines": [
      {
        "code": "Bloque \"phoneReadGyroscope\" no preparado
para generar código python"
      }
    ]
  }
```


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

```
}
```

2.6.3. Placa recibe datos del sensor de campo magnético

El objetivo de este bloque es obtener el valor del campo magnético del dispositivo. El dispositivo móvil proporciona este valor en cada uno de los ejes.

Por ello, se ha creado el bloque "`phoneReadMagnetic`".

En el bloque se puede especificar de qué eje se quiere obtener el valor del campo magnético (x,y o z).

Si por ejemplo, asociamos ese bloque a una variable (de nombre magn) y la colocamos en el bloque de variables, el código generado sería el siguiente:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >

/*** Global variables and function definition ***/
float getMagneticField(String axis, bqSoftwareSerial & phone)
{
  phone.println(String("readMagnetic-") + String(axis));
String data = "";
  while (data == "") {
    data = phone.readString();
  }
  return data.toFloat();
}

bqSoftwareSerial dispositivo_0(0, 1, 9600);

float magn = getMagneticField("x", dispositivo_0);

/*** Setup ***/
void setup() { }

/*** Loop ***/
void loop() {}
```


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

Como se puede observar en el código, el bloque genera una función `getMagneticField` que recibe como parámetros el eje en el cuál se desea obtener el valor ("x" en este caso) y el dispositivo del que se quiere medir ("dispositivo_0", de tipo `bqSoftwareSerial`).

La función `getMagneticField` envía al teléfono el comando **"readMagnetic"** seguido del eje del que desea obtener el valor. Una vez hecho esto, se queda esperando a que el móvil le conteste con el dato y una vez esto ocurre lo devuelve convertido a `float`. Estos valores vendrán expresados en µT .

Definición del bloque en formato JSON:

```
{
  "type": "output",
  "name": "phoneReadMagnetic",
  "connectors": [
    {
      "type": "connector--output",
      "accept": "connector--input"
    }
  ],
  "bloqClass": "bloq-phone-read-magnetic",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-phone-magnetic"
      },
      {
        "id": "AXIS",
        "alias": "staticDropdown",
        "options": [
          {
            "label": "x",
            "value": "\"x\""
          },
          {
            "label": "y",
            "value": "\"y\""
          },
          {
            "label": "z",
            "value": "\"z\""
          }
        ]
      },
```


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

```
      {
        "alias": "text",
        "value": "bloq-phone-of"
      },
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      },
      {
        "alias": "text",
        "value": "(µT)"
      }
    ]
  ],
  "code": "{PHONE}.readString()",
  "arduino": {
    "includes": [
      "BitbloqSoftwareSerial.h"
    ],
    "extraFunctionCode": "float getMagneticField(String
axis,bqSoftwareSerial &
phone){phone.println(String(\"readMagnetic-
\")+String(axis));String
data=\"\";while(data==\"\"){data=phone.readString();}return
data.toFloat();}",
    "code": "getMagneticField({AXIS}, {PHONE})"
  },
  "returnType": {
    "type": "simple",
    "value": "float"
  },
  "python": {
    "codeLines": [
      {
        "code": "Bloque \"phoneReadMagnetic\" no preparado
para generar código python"
      }
    ]
  }
}
```


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

2.6.4. Placa recibe datos del sensor de luz

El objetivo de este bloque es obtener el valor del sensor de luz del dispositivo.

Por ello, se ha creado el bloque "`phoneReadLight`".

Si por ejemplo, asociamos ese bloque a una variable (de nombre light) y la colocamos en el bloque de variables, el código generado sería el siguiente:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >

/*** Global variables and function definition ***/
float getLight(bqSoftwareSerial & phone) {
phone.println(String("readLight-"));
  String data = "";
  while (data == "") {
    data = phone.readString();
  }
  return data.toFloat();
}

bqSoftwareSerial dispositivo_0(0, 1, 9600);

float light = getLight(dispositivo_0);

 /*** Setup ***/
void setup() { }

/*** Loop ***/
void loop() {}
```

Como se puede observar en el código, el bloque genera una función `getLight` que recibe como parámetros el dispositivo del que se quiere medir ("dispositivo_0", de tipo `bqSoftwareSerial`).

La función `getLight` envía al teléfono el comando **"readLight"**. Una vez hecho esto, se queda esperando a que el móvil le conteste con el dato y una vez esto ocurre lo devuelve convertido a `float`. Estos valores vendrán expresados en lx.

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

Definición del bloque en formato JSON:

```
{
  "type": "output",
  "name": "phoneReadLight",
  "connectors": [
    {
      "type": "connector--output",
      "accept": "connector--input"
    }
  ],
  "bloqClass": "bloq-phone-read-light",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-phone-light"
      },
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      },
      {
        "alias": "text",
        "value": "(lx)"
      }
    ]
  ],
  "code": "{PHONE}.readString()",
  "arduino": {
    "includes": [
      "BitbloqSoftwareSerial.h"
    ],
    "extraFunctionCode": "float getLight(bqSoftwareSerial &
phone){phone.println(String(\"readLight-\"));String
data=\"\";while(data==\"\"){data=phone.readString();}return
data.toFloat();}",
    "code": "getLight({PHONE})"
  },
  "returnType": {
    "type": "simple",
    "value": "float"
  },
  "python": {
    "codeLines": [
      {
        "code": "Bloque \"phoneReadLight\" no preparado para
generar código python"
      }
    ]
```


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

```
  }
}
```

2.6.5. Placa recibe valores del sensor de proximidad

El objetivo de este bloque es obtener el valor del sensor de proximidad del dispositivo. Dependiendo de este valor podremos saber si el dispositivo está tapado o no.

Por ello, se ha creado el bloque "`phoneisCovered`".

En el bloque se puede especificar si se desea saber si el dispositivo está tapado (opción "tapado") o no está tapado (opción "no tapado").

Si por ejemplo, asociamos ese bloque a una variable (de nombre covered) y la colocamos en el bloque de variables, el código generado sería el siguiente:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >


/*** Global variables and function definition ***/
boolean getProx(String cv, bqSoftwareSerial & phone) {
phone.println(String("readProx-") + String(cv));
  String data = "";
  boolean result = false;
  while (data == "") {
    data = phone.readString();
  }
  if (data.indexOf("true") >= 0) {
    result = true;
  } else {
    result = false;
  }
  return result;
}

bqSoftwareSerial dispositivo_0(0, 1, 9600);

boolean covered = getProx("covered", dispositivo_0);

/*** Setup ***/
void setup() { }
```


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

```
/*** Loop ***/
void loop() {}
```

Como se puede observar en el código, el bloque genera una función `getProx` que recibe como parámetro el dispositivo del que se quiere medir ("dispositivo_0", de tipo `bqSoftwareSerial`) y un String indicando si se quiere saber si está tapado el dispositivo (parámetro con valor `"covered"`) o destapado (parámetro con valor `"ncovered"`).

La función `getProx` envía al teléfono el comando **"readProx"** seguido de la información que se desea conocer (si se desea conocer si está tapado, se le envía `"covered"`, y en caso contrario `"ncovered"`). Una vez hecho esto, se queda esperando a que el móvil le conteste con el dato. Cuando devuelve el dato, como será un `String`, se evalúa y se transforma a `boolean`, que es el tipo que devolverá la función.

Definición del bloque en formato JSON:

```
{
  "type": "output",
  "name": "phoneisCovered",
  "connectors": [
    {
      "type": "connector--output",
      "accept": "connector--input"
    }
  ],
  "bloqClass": "bloq-phone-isCovered",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-the"
      },
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      },
      {
        "alias": "text",
        "value": "bloq-phone-is"
      },
      {
        "id": "COVERED",
        "alias": "staticDropdown",
        "options": [
```


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

```
            {
              "label": "bloq-phone-covered",
              "value": "\"covered\""
            },
            {
              "label": "bloq-phone-not-covered",
              "value": "\"ncovered\""
            }
          ]
        }
      ]
    ],
    "code": "{PHONE}.readString()",
    "arduino": {
      "includes": [
        "BitbloqSoftwareSerial.h"
      ],
      "extraFunctionCode": "boolean getProx(String
cv,bqSoftwareSerial & phone){phone.println(String(\"readProx-
\")+String(cv));String data=\"\";boolean
result=false;while(data==\"\"){data=phone.readString();}if(data.
indexOf(\"true\")>=0){result=true;}else{result=false;}return
result;}",
      "code": "getProx({COVERED}, {PHONE})"
    },
    "returnType": {
      "type": "simple",
      "value": "boolean"
    },
    "python": {
      "codeLines": [
        {
          "code": "Bloque \"phoneisCovered\" no preparado para
generar código python"
        }
      ]
    }
}
```


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

2.6.6. Placa recibe valores de la orientación del móvil

El objetivo de este bloque es obtener la posición del dispositivo relativa al marco de referencia de la Tierra (más específicamente respecto al polo magnético norte).

El dispositivo móvil proporciona el valor del azimut (ángulo alrededor del eje z), roll (ángulo alrededor del eje y) y pitch (ángulo alrededor del eje x).

Por ello, se ha creado el bloque "`phoneReadOrientation`".

En el bloque se puede especificar qué valor se quiere obtener de la orientación (azimut, roll o pitch).

Si por ejemplo, asociamos ese bloque a una variable (de nombre azim) y la colocamos en el bloque de variables, el código generado sería el siguiente:

```
/*** Included libraries ***/
#include < SoftwareSerial.h >
#include < BitbloqSoftwareSerial.h >

/*** Global variables and function definition ***/
float getOrientation(String axis, bqSoftwareSerial & phone) {
  phone.println(String("readOrientation-") +   String(axis));
  String data = "";
  while (data == "") {
    data = phone.readString();
  }
  return data.toFloat();
}

bqSoftwareSerial dispositivo_0(0, 1, 9600);

float azim = getOrientation("azimuth", dispositivo_0);


/*** Setup ***/
void setup() { }

/*** Loop ***/
void loop() {}
```


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

Como se puede observar en el código, el bloque genera una función `getOrientation` que recibe como parámetros el valor de la orientación que se desea obtener ("azimuth" en este caso) y el dispositivo del que se quiere medir ("dispositivo_0" de tipo `bqSoftwareSerial`).

La función `getOrientation` envía al teléfono el comando "**readOrientation**" seguido del valor a medir. Una vez hecho esto, se queda esperando a que el móvil le conteste con el dato y una vez esto ocurre lo devuelve convertido a `float`. Estos valores vendrán expresados en º.

Definición del bloque en formato JSON:

```
{
  "type": "output",
  "name": "phoneReadOrientation",
  "connectors": [
    {
      "type": "connector--output",
      "accept": "connector--input"
    }
  ],
  "bloqClass": "bloq-phone-read-orientation",
  "content": [
    [
      {
        "alias": "text",
        "value": "bloq-value"
      },
      {
        "id": "AXIS",
        "alias": "staticDropdown",
        "options": [
          {
            "label": "bloq-phone-orientation-azimuth",
            "value": "\"azimuth\""
          },
          {
            "label": "bloq-phone-orientation-roll",
            "value": "\"roll\""
          },
          {
            "label": "bloq-phone-orientation-pitch",
            "value": "\"pitch\""
          }
        ]
      },
```


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

```
      {
        "alias": "text",
        "value": "bloq-phone-of"
      },
      {
        "id": "PHONE",
        "alias": "dynamicDropdown",
        "options": "serialElements"
      },
      {
        "alias": "text",
        "value": "(°)"
      }
    ]
  ],
  "code": "{PHONE}.readString()",
  "arduino": {
    "includes": [
      "BitbloqSoftwareSerial.h"
    ],
    "extraFunctionCode": "float getOrientation(String
axis,bqSoftwareSerial &
phone){phone.println(String(\"readOrientation-
\")+String(axis));String
data=\"\";while(data==\"\"){data=phone.readString();}return
data.toFloat();}",
    "code": "getOrientation({AXIS}, {PHONE})"
  },
  "returnType": {
    "type": "simple",
    "value": "float"
  },
  "python": {
    "codeLines": [
      {
        "code": "Bloque \"phoneReadOrientation\" no preparado
para generar código python"
      }
    ]
  }
}
```


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