

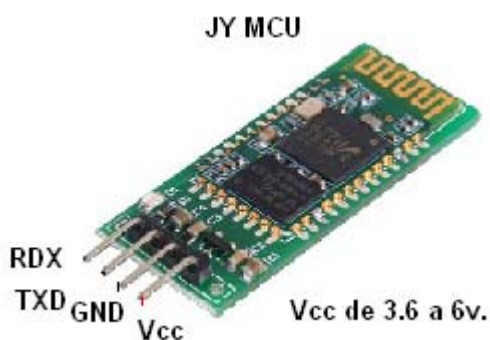
## Aplicación Android para controlar arduino mediante bluetooth

Vamos a crear una aplicación para dispositivos Android con la que podemos controlar la placa arduino mediante un módulo bluetooth (JY-MCU).

### Módulo Bluetooth:

El módulo JY-MCU podemos utilizarlo para comunicarse con arduino mediante bluetooth con ayuda de un dispositivo con sistema Android.

Conectar a la placa arduino RDX a D1(TX), TXD a D0(RX), GND y Vcc a los pines correspondientes. La información se recibe vía serie.



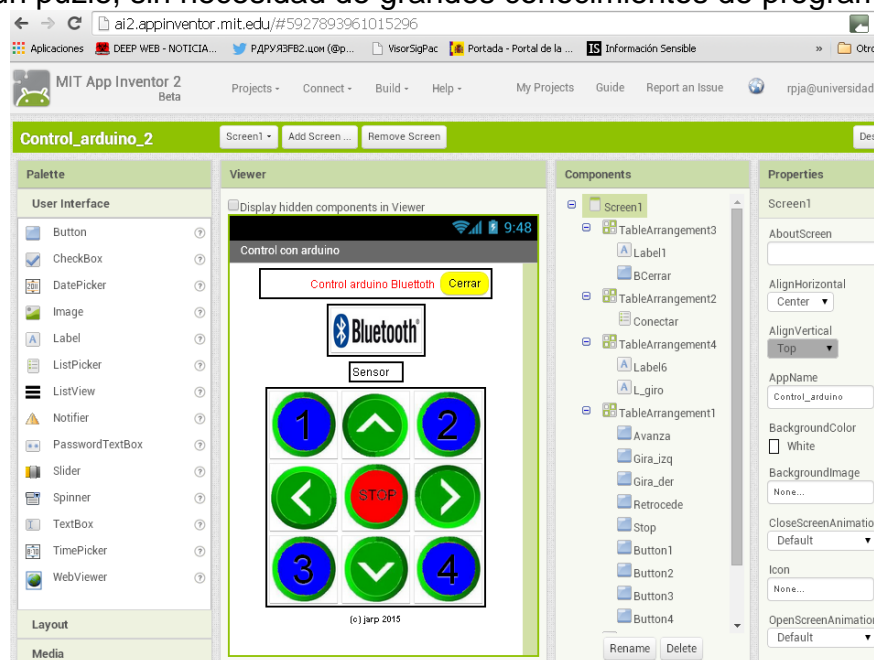
Para utilizar el dispositivo móvil como elemento de control, vincular el módulo como **livor** (contraseña **1234**).

Hay aplicaciones en Android para descargar como puede ser Motbot (<http://www.mobot.es/>)

También podemos configurar los valores del módulo mediante comandos AT (ver <http://www.extremadura-web.es/Blog/2012/10/29/comunicacion-bluetooth-serie-arduino-y-basic4android/>)

(Con ayuda del teclado y el terminal serie de arduino, podemos depurar el scrip).

Para realizar aplicaciones en Android podemos utilizar diferentes programas (Eclipse, BASIC4Android, etc.) programando mediante código de instrucciones o utilizar una herramienta gratuita **App inventor2** programando mediante bloques de instrucciones a semejanza de un puzle, sin necesidad de grandes conocimientos de programación..



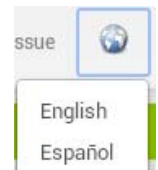
**App inventor**, necesita para funcionar, tener una cuenta de Google y conexión a internet (aunque ya no es necesario, existe una aplicación de forma portable para trabajar sin conexión <http://sourceforge.net/p/ailivecomplete/wiki/Summary/>).

Se puede trabajar sin tener móvil o tablet, instalando el emulador, pero la forma más cómoda de trabajar es tener un dispositivo Android con la aplicación **MIT AI2 Companion** instalada (se puede descargar desde Google Play <https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3>).



Con esta aplicación podemos estar conectados, ver los cambios y descargar la aplicación realizada.

**App inventor2** se puede configurar para trabajar en español pulsando en el icono de la bola →



Unos cursos muy buenos con ejemplos en castellano lo podemos ver en: <http://www.iesromerovargas.es/android/> o <https://sites.google.com/site/appinventormegusta/>

Vamos a realizar una aplicación para controlar arduino con ayuda de **App Inventor2** (<http://ai2.appinventor.mit.edu/>).

Trabajaremos en inglés ya que las instrucciones se parecen más a otros lenguajes de programación y es bueno para los alumnos que se familiaricen con los bloques en inglés.

En los siguientes enlaces se puede descargar la aplicación y el proyecto completo para que los alumnos puedan modificarlo y adaptarlo a sus aplicaciones.

Descargar aplicación:

[https://dl.dropboxusercontent.com/u/384051/appinventor/app/Control\\_arduino.apk](https://dl.dropboxusercontent.com/u/384051/appinventor/app/Control_arduino.apk)



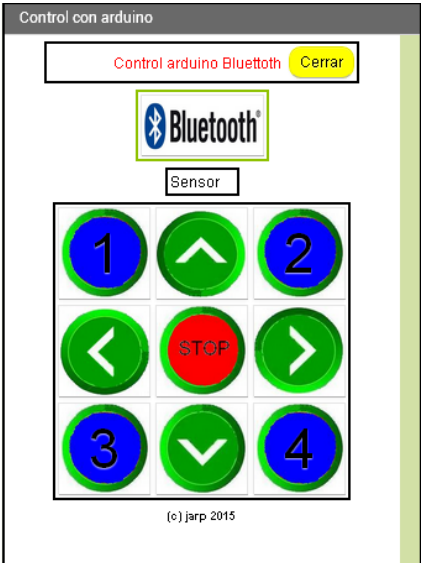

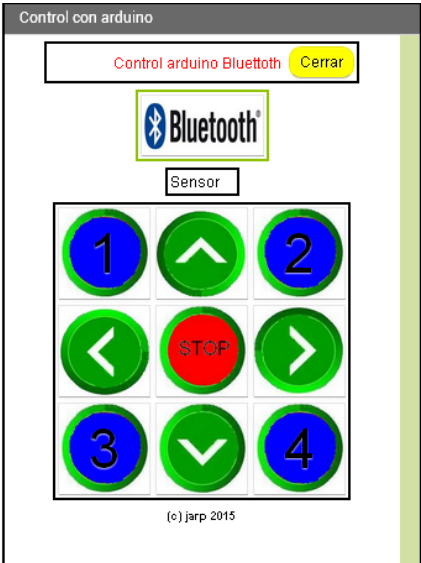

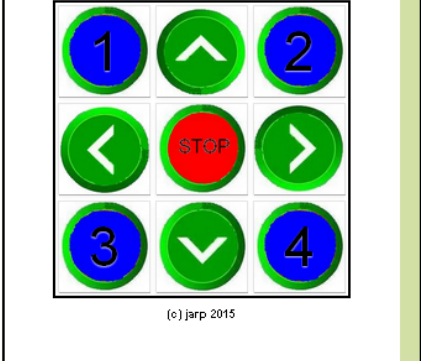

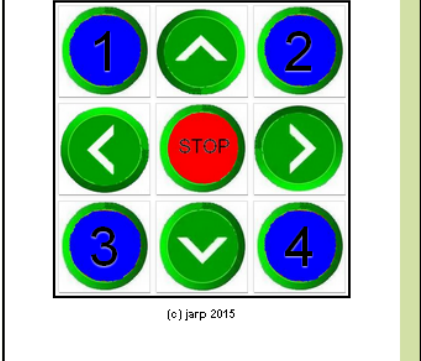

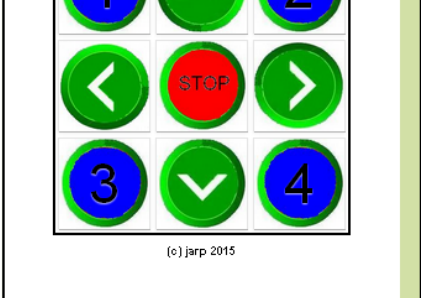







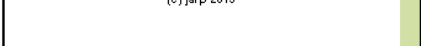




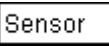
Descargar proyecto completo:

[https://dl.dropboxusercontent.com/u/384051/appinventor/app/Control\\_arduino.aia](https://dl.dropboxusercontent.com/u/384051/appinventor/app/Control_arduino.aia)

## Diseño de la aplicación Android:

La aplicación que llamaremos **Control\_arduino**, dispone de varios botones para enviar (mediante bluetooth) cuando se pulsa cada botón un carácter diferente. Utiliza el sensor de posición para enviar un carácter distinto cuando detectar un determinado giro.

En la siguiente tabla podemos ver los caracteres que envía al pulsar el correspondiente botón.

	Botones	Carácter enviado
		Cierra la aplicación
		Muestra los dispositivos bluetooth vinculados y permite conectarnos ( <b>livor</b> con <b>1234</b> )
		1
		2
		3
		4
		A
		I
		D
		R
		S
		Muestra y envía al inclinar el dispositivo: 20° (> 20° derecha): <b>E</b> -20° (>20° izquierda): <b>W</b> Entre -20° y 20° : <b>0</b>

Estos caracteres se pueden cambiar al diseñar el programa:

Los pasos a seguir para crear una nueva aplicación:

1. Una vez conectados a <http://ai2.appinventor.mit.edu/> creamos un nuevo proyecto desde **My projects-> strat new project**.
2. Desde la pestaña **Designer** diseñamos la pantalla de la aplicación colocando los componentes, subiendo las imágenes de los botones y configurándolos.
3. Desde la pestaña **Blocks** creamos el código de la aplicación, componiendo los bloques de código.
4. Una vez terminada la aplicación desde **Buils ->App(provide QR code for .apk)** compila, crea la aplicación y genera un código QR para descargarla en nuestro dispositivo (desde la aplicación **MIT AI2 Companion**).

5. También podemos guardar la aplicación en nuestro ordenador con **Buils ->App(save .apk to my computer..)**.
6. Desde **Project->Export selected project (.aia) to my computer..** podemos guardar el proyecto completo en nuestro ordenador.

Durante el desarrollo de la aplicación podemos ver el resultado en nuestro dispositivo o en el simulador desde **Connect-> Al Companin** o **Connect->Emulador**.

Para facilitar el trabajo podemos importar el proyecto descargado desde el enlace anterior ([https://dl.dropboxusercontent.com/u/384051/appinventor/app/Control\\_arduino.aia](https://dl.dropboxusercontent.com/u/384051/appinventor/app/Control_arduino.aia)) a nuestro ordenador e importarlo a **App Inventor2** desde **Project->Import project (.aia) from my computer..** y realizar las modificaciones oportunas.

### Código de la aplicación:

- Definimos variables con los valores a enviar:



The image shows a vertical stack of ten 'initialize global' code blocks in App Inventor. Each block is configured to initialize a specific variable to a string value. The variables and their values are: v\_1 to '1', v\_2 to '2', v\_3 to '3', v\_4 to '4', v\_avanza to 'A', v\_retrocede to 'R', v\_izq to 'I', v\_der to 'D', v\_stop to 'S', and v\_sensor to '0'.

```
initialize global v_1 to " 1 "  
initialize global v_2 to " 2 "  
initialize global v_3 to " 3 "  
initialize global v_4 to " 4 "  
initialize global v_avanza to " A "  
initialize global v_retrocede to " R "  
initialize global v_izq to " I "  
initialize global v_der to " D "  
initialize global v_stop to " S "  
initialize global v_sensor to " 0 "
```


- Acción del botón cerrar aplicación:



The image shows a single code block for a button click event. It is configured with the event 'when BCerrar .Click' and the action 'do close application'.

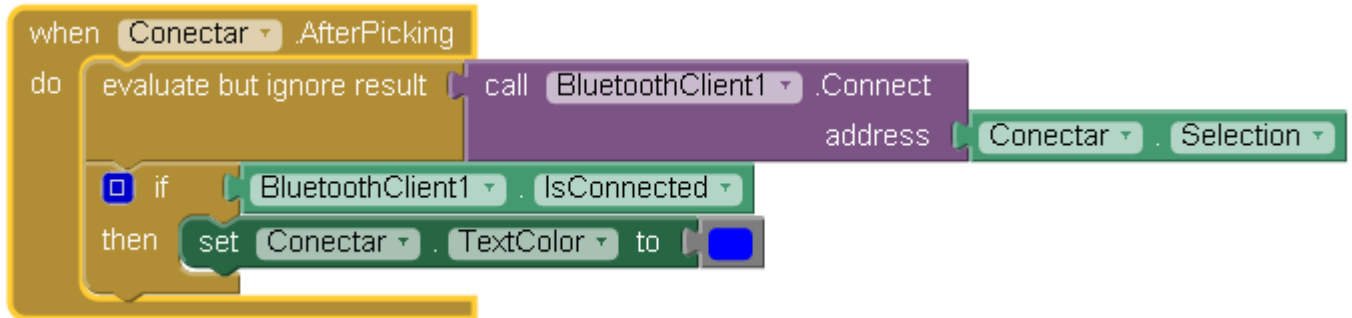
```
when BCerrar .Click  
do close application
```



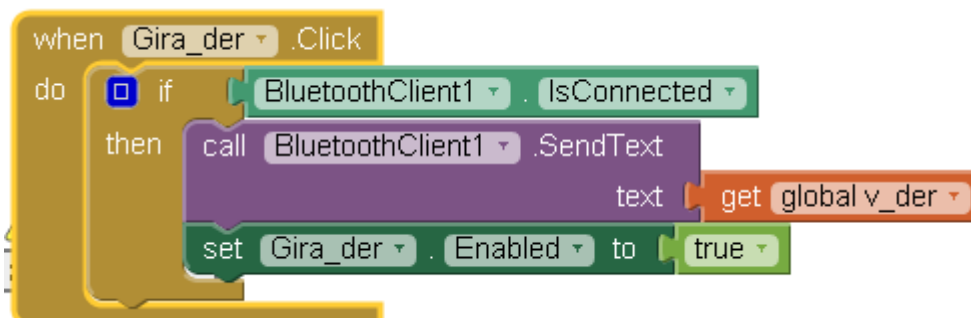
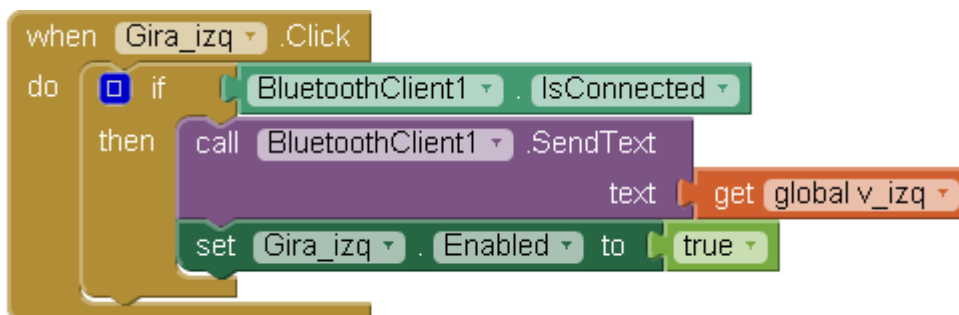
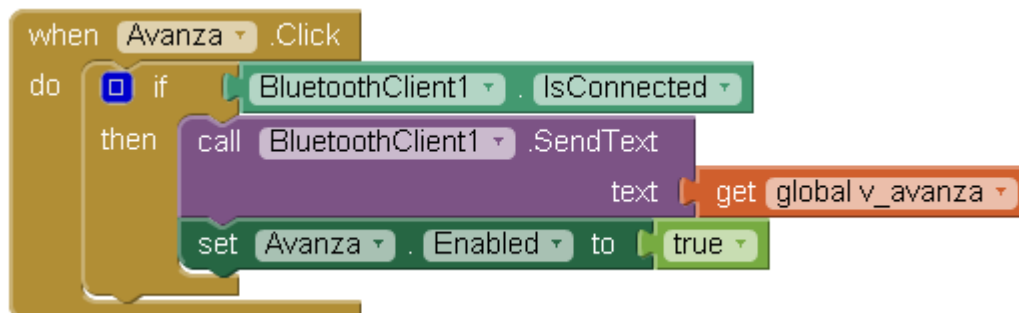
- Al pulsar  muestra la lista de dispositivos Bluetooth



- Al seleccionar el dispositivo Bluetooth vinculado, se conecta con arduino.



- Al pulsar los distintos botones, comprueba si hay conexión con el dispositivo Bluetooth y envía el carácter definido en la variable correspondiente. La orden **set .. Enabled = true** mantiene el botón activo después de pulsarlo, se podría suprimir o utilizar para ocultar botones una vez pulsados.



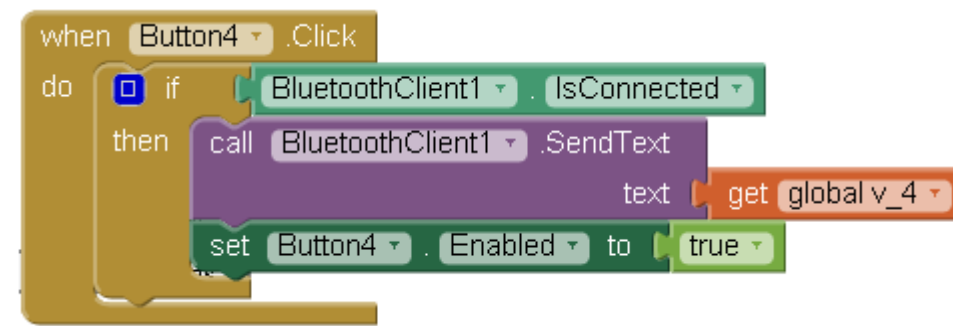
```
when Retrocede .Click
do
  if BluetoothClient1 .IsConnected
  then
    call BluetoothClient1 .SendText
    text get global v_retrocede
    set Retrocede .Enabled to true
```

```
when Stop .Click
do
  if BluetoothClient1 .IsConnected
  then
    call BluetoothClient1 .SendText
    text get global v_stop
    set Stop .Enabled to true
```

```
when Button1 .Click
do
  if BluetoothClient1 .IsConnected
  then
    call BluetoothClient1 .SendText
    text get global v_1
    set Button1 .Enabled to true
```

```
when Button2 .Click
do
  if BluetoothClient1 .IsConnected
  then
    call BluetoothClient1 .SendText
    text get global v_2
    set Button2 .Enabled to true
```

```
when Button3 .Click
do
  if BluetoothClient1 .IsConnected
  then
    call BluetoothClient1 .SendText
    text get global v_3
    set Button3 .Enabled to true
```

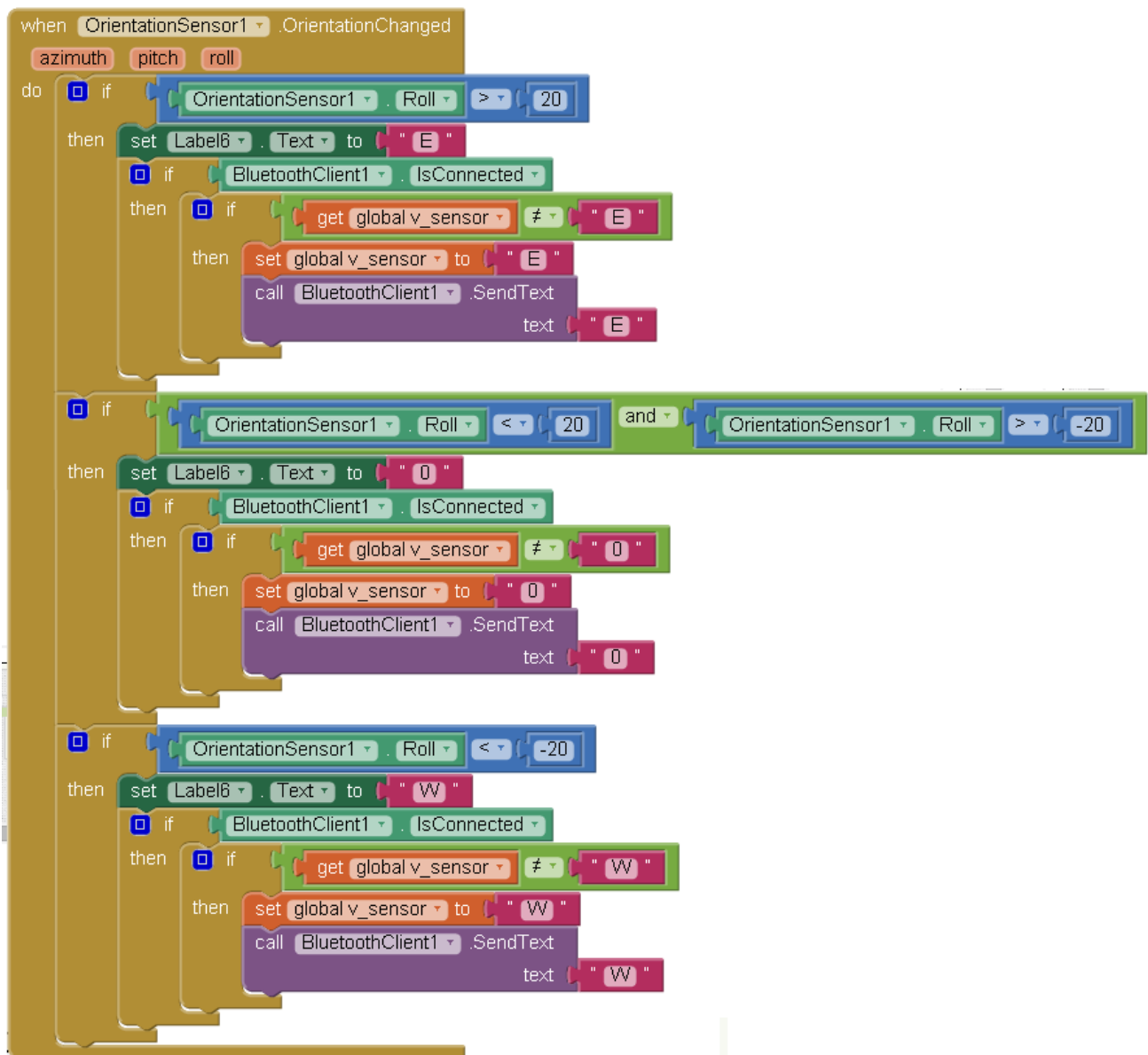


- **Sensor de orientación:** este sensor cada vez que se mueve el dispositivo, según el sentido, da los valores de **azimuth** (orientación con respecto al norte entre 0° y 360°), **pitch** (inclinación horizontal entre -90° 0° +90°), **roll** (giro de la pantalla entre -90° 0° +90°).

Vamos a utilizar la propiedad **roll**, para enviar a arduino 3 caracteres diferentes, según giremos el dispositivo (podría utilizarse por ejemplo para controlar los giros de un vehículo). Comprobamos (if ) que el dispositivo gira un ángulo mayor de 20° a la derecha y -20° a la izquierda para enviar respectivamente los caracteres **E** y **W** . Cuando el giro en uno u otro sentido sea menor de 20° envía el carácter **0**

Para evitar que el sensor este enviado continuamente el carácter **E, W, 0** , utilizamos la variable **global\_sensor** para almacenar el carácter enviado y no vuelve a enviar otro hasta que detecte un cambio de posición diferente (izquierda, centro, derecha) y cambie el valor a enviar.

Los valores enviados y a partir de que ° detecta el giro se pueden modificar.



Una mejora de la aplicación sería crear otra pantalla para poder configurar los valores enviados desde la aplicación, almacenándolos en un archivo de texto en el dispositivo (ver **Storage File**).



## Ejemplo scrip arduino para probar:

```
/*programa para pruebas de control de salidas mediante bluetooth
(c) jarp Universidad Laboral Albacete 2015
Utilizamos 8 leds
Enciende leds según el botón pulsado. Con S apaga todos los led.
*/
int digTotal = 8; // Numero de pines a usar
//int vdig[] = {0,0,0,0,0,0,0,0}; // valores iniciales (Tantos como pines usemos)
int dig[] = {13,12,11,10,9,8,7,6}; //define el nº de pin de cada dig
char dato; //almacena caracter leido del bluetooth
void setup() {
//Velocidad del modulo bluetooth, 9600 por defecto
Serial.begin(9600);
    for (int i=0; i < digTotal; i++) {
        pinMode(dig[i], OUTPUT); // inicializa los pines digitales como salida
        digitalWrite(dig[i],LOW); // pone a nivel bajo LOW=0
    } //for
    Serial.println("Pulsar 0,1,2,3,A,R,D,I,E,W para encender y S para apagar led");
} //setup
void apagaTodos(){ //apaga todos los led
    for (int i=0; i < digTotal; i++) {
        digitalWrite(dig[i], LOW);
    } //for
} //apagaTodos
void loop(){
    if(Serial.available()>0){
        dato=Serial.read();
        Serial.println(dato);
        switch (dato) {
            case 'S':
                apagaTodos();
                break;
            case '1':
                digitalWrite(dig[1],HIGH);
                break;
            case '2':
                digitalWrite(dig[2],HIGH);
                break;
            case '3':
                digitalWrite(dig[3],HIGH);
                break;
            case '4':
                digitalWrite(dig[4],HIGH);
                break;
            case 'A':
                digitalWrite(dig[0],HIGH);
                digitalWrite(dig[1],HIGH);
                break;
            case 'R':
                digitalWrite(dig[2],HIGH);
                digitalWrite(dig[3],HIGH);
                break;
            case 'D':
```

```
digitalWrite(dig[4],HIGH);
digitalWrite(dig[5],HIGH);
break;
case 'I':
digitalWrite(dig[6],HIGH);
digitalWrite(dig[7],HIGH);
break;
case 'E':          //sensor posición derecha
digitalWrite(dig[0],HIGH);
digitalWrite(dig[7],HIGH);
break;
case 'W':          //sensor posición izquierda
digitalWrite(dig[1],HIGH);
digitalWrite(dig[6],HIGH);
break;
case 'O':          //sensor posición centro
digitalWrite(dig[2],HIGH);
digitalWrite(dig[3],HIGH);
break;
} // switch
} //serial
delay(100);
} //loop
```