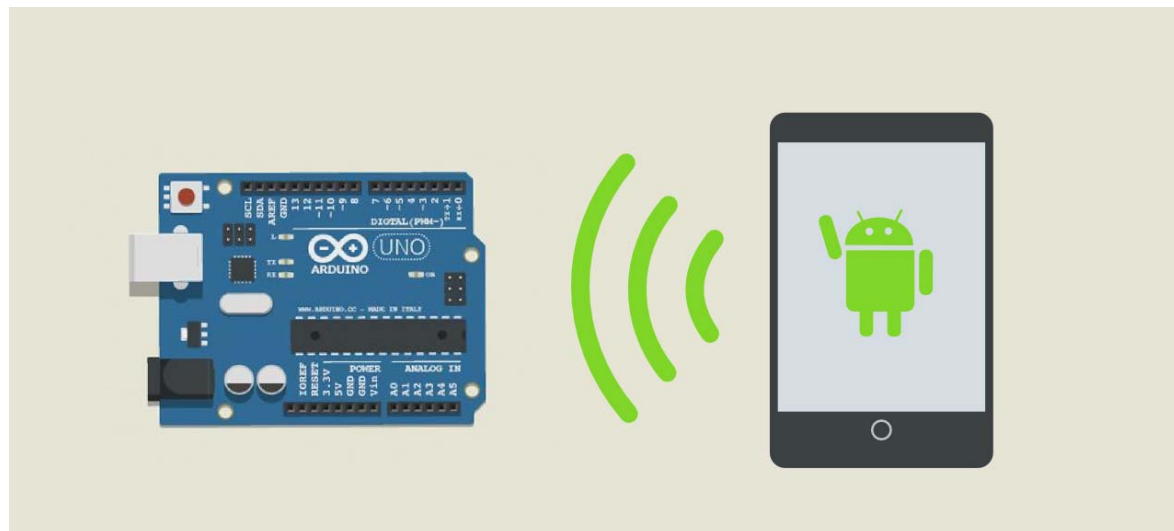


# Creación de aplicaciones con App Inventor para prototipos con Arduino vía BT



Nuria Fernández Enríquez  
nfernandezen@educa.jcyl.es

## Contenido

1. ¿QUÉ ES BLUETOOTH? .....	2
2. MÓDULOS BLUETOOTH PARA ARDUINO .....	3
3. PRACTICA 1: CONFIGURANDO EL MODULO HC-06.....	5
4. PRACTICA 2: ENCENDIDO Y APAGADO DE UN LED .....	8
APP INVENTOR .....	8
ARDUINO .....	10
5. CONTROL DE UN SERVO por BT .....	11
6. CONTROL DE UN RGB POR BLUETOOTH .....	14
CONEXIONADO.....	15
OPCIÓN 2: MEDIANTE SLIDERS .....	15
APP INVENTOR .....	15
ARDUINO .....	18
OPCIÓN 3: MEDIANTE UNA MATRIZ DE COLOR.....	19
7. PRÁCTICA 4: ENVIANDO DATOS MEDIANTE EL SENSOR DE DISTANCIA (ultrasónico).....	21
8. WEBGRAFÍA.....	24

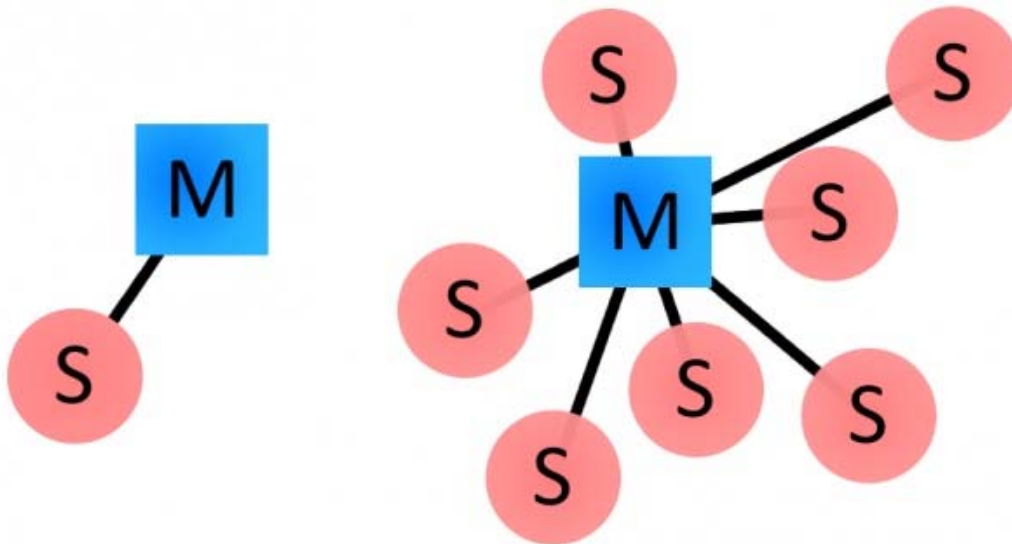
## 1. ¿QUÉ ES BLUETOOTH?

Es una tecnología de corto alcance que permite la comunicación inalámbrica entre dos dispositivos digitales, con un rango de 100m (en la teoría)

**No hay necesidad de utilizar ningún cable** para realizar la conexión como tampoco se requiere colocar los dispositivos frente a frente como sucede con la tecnología de infrarrojos.

Los dispositivos BlueTooth pueden actuar como Masters o como Slaves (Maestros o esclavos).

La diferencia es que un BlueTooth Slave solo puede conectarse a un master y a nadie más, en cambio un master BlueTooth, puede conectarse a varios Slaves o permitir que ellos se conecten y recibir y solicitar información de todos ellos, arbitrando las transferencias de información ( Hasta un máximo de 7 Slaves)





Cada uno de los dispositivos que se identifican vía BlueTooth presentan una dirección única de 48 bits y además un nombre de dispositivo que nos sirva para identificarlos. Por eso cuando configuras tu móvil puedes especificar un nombre propio que será el que mostraras a los demás cuando busquen tu teléfono en las inmediaciones.

Cuando vinculas dos dispositivos BT, se inicia un proceso en el que ellos se identifican por nombre y dirección interna y se solicitan la clave PIN para autorizar la conexión.

Si el emparejamiento se realiza con éxito, ambos nodos suelen guardar la identificación del otro y cuando se encuentran cerca se vuelven a vincular sin necesidad de intervención manual. Por eso el CD de tu coche reconoce el móvil de tu bolsillo en cuanto te subes y puedes reproducir la música que tienes en tu Smartphone.

## 2. MÓDULOS BLUETOOTH PARA ARDUINO

En el mercado existen diversos dispositivos que permiten la comunicación Bluetooth, pero nos centraremos en los módulos HC-05 y HC-06.

	HC-05	HC-06
Aspecto		
Pines	6 TX, RX, VCC, GND, KEY , STATE	4 TX, RX, VCC, GND

Nos centraremos en el HC-06.

Para poder configurar el HC-06 es necesario que este **NO este emparejado ni siendo usado por ningún dispositivo.**

El módulo HC-06 acepta un set muy básico de comandos, que permite pocas configuraciones, pero que sin duda será útil para personalizar este económico módulo y configurarlo para satisfacer las necesidades de la aplicación.

Los comandos que soporta son:

Prueba de funcionamiento:

- Envía: AT
- Recibe: OK

Configurar el Baudrate:

- Envía: AT+BAUD<Numero>

El parámetro número es un caracter hexadecimal de '1' a 'c' que corresponden a los siguientes Baud Rates: 1=1200, 2=2400, 3=4800, 4=9600, 5=19200, 6=38400, 7=57600, 8=115200, 9=230400, A=460800, B=921600, C=1382400

- Recibe: OK<baudrate>

Configurar el Nombre de dispositivo Bluetooth:

- Envía: AT+NAME<Nombre>
- Recibe: OKsetname

Configurar el código PIN de emparejamiento:

- Enviar: AT+PIN<pin de 4 digitos>
- Recibe: OK<pin de 4 digitos>

### Obtener la version del firmware:

- Enviar: AT+VERSION
- Recibe: Linvor1.8

### Recordar que para trabajar con los comandos AT con este módulo:

No es necesario finalizar el comando con `\r\n`, pero si es necesario ingresar los comandos con todos los caracteres seguidos sin pausas. NO hay necesidad de dar "enter" para finalizar un comando. El modulo tiene un Temporizador que hace necesario introducir el comando de una sola vez, sin pausas entre los caracteres.

Por lo anterior, si utilizamos un emulador de terminal hay que pegarlos en leste y no escribirlos uno a uno con el teclado. También podemos usar el "monitor serial" de Arduino configurado como se muestra en la imagen más arriba en este artículo.

Hay que tener cuidado de introducir TODAS LAS LETRAS DEL COMANDO en MAYUSCULAS, ya que de lo contrario, no funcionarán.

Las respuestas no parecen respuestas estándar a comandos AT.

Muy importante. Cómo conectar el módulo BT

[http://kio4.com/appinventor/9A\\_bluetooth\\_conectar\\_rxtx.htm](http://kio4.com/appinventor/9A_bluetooth_conectar_rxtx.htm)

Unas veces veremos el **módulo Bluetooth** conectados a los terminales **RX-0 y TX-1**, que son los terminales "oficiales" del Serial en el Arduino y otras veces lo veremos conectado a los **terminales 10 y 11**.

- Vamos a ver porqué utilizamos un par de conexiones o el otro.

- Los terminales "oficiales" del puerto Serie o Serial, en el Arduino son el 0 y el 1. Donde está indicado mediante **RX y TX** respectivamente. Estos terminales lo utiliza el Arduino para **cargar la aplicación** y también lo utiliza para el **Serial Monitor**. Están conectado a unos pequeños LED que hay sobre la placa del Arduino, por eso cuando cargamos una aplicación o estamos observando datos en el Serial Monitor, vemos parpadear esos pequeños LED.

- El módulo Bluetooth también funciona con **un puerto Serie**, de tal manera que lo podemos conectar a los terminales anteriormente nombrado RX y TX del Arduino. Pero debemos tener en cuenta que, como **el Arduino utiliza esos terminales para cargar la aplicación**, **debemos quitar el módulo Bluetooth** (o quitarle la alimentación) de esos terminales cada vez que vayamos a cargar una aplicación. Cuando la aplicación esté cargada podemos volver a conectar completamente los terminales del módulo Bluetooth.

- Además **no podremos utilizar el Serial Monitor(?)** para ver los resultados que está obteniendo la aplicación, ya que el Serial Monitor también utilizaría los terminales RX y TX, por lo cual entran en conflicto y avisa de que no puede ser utilizado. Así que no podemos tener conectado el módulo Bluetooth y utilizar **Serial.print**(temperatura);

- **¿Qué podemos hacer para no tener que quitar el módulo Bluetooth cada vez que cargamos una aplicación y poder utilizar el Serial Monitor?**

- Establecer además de la RX-0 y TX-1 **otro par de terminales de puerto Serie** para el módulo Bluetooth y dejar esos terminales "oficiales" para cargar la aplicación y poder utilizar el **Serial Monitor**.

- **¿Cómo lo hacemos?**

- En el código hay que incluir la librería:

```
#include <SoftwareSerial.h>
```

e indicar los terminales Serie que vamos a utilizar, además de ponerle un nuevo nombre a este puerto Serie, en este caso se ha puesto **I2CBT**.

```
SoftwareSerial I2CBT(10,11);  
// El TX del módulo BT va al pin 10 del Arduino  
// El RX del módulo BT va al pin 11 del Arduino
```

También debemos iniciar este puerto y asignarle una velocidad (baudios):

```
I2CBT.begin(9600);
```

Luego a lo largo del código utilizamos ese puerto llamándolo así, según queramos leer o escribir...

```
serialA=I2CBT.read();
```

```
I2CBT.write(Datos[0]);
```

Al mismo tiempo podemos utilizar el **Serial** "oficial" para ver cómo van evolucionando los datos de la aplicación, pero eso utilizamos el comando **Serial**.

```
Serial.begin(9600);
```

```
Serial.print("Temperatura = ");
```

```
Serial.println(temperatura);
```

De esta manera pueden convivir los dos puertos Serial, el "oficial" y el creado en los terminales 10 y 11.

- Si solo queremos utilizar el Serial "oficial", no es necesario incluir la librería <SoftwareSerial.h> y utilizaríamos **Serial.print** o **Serial.println**

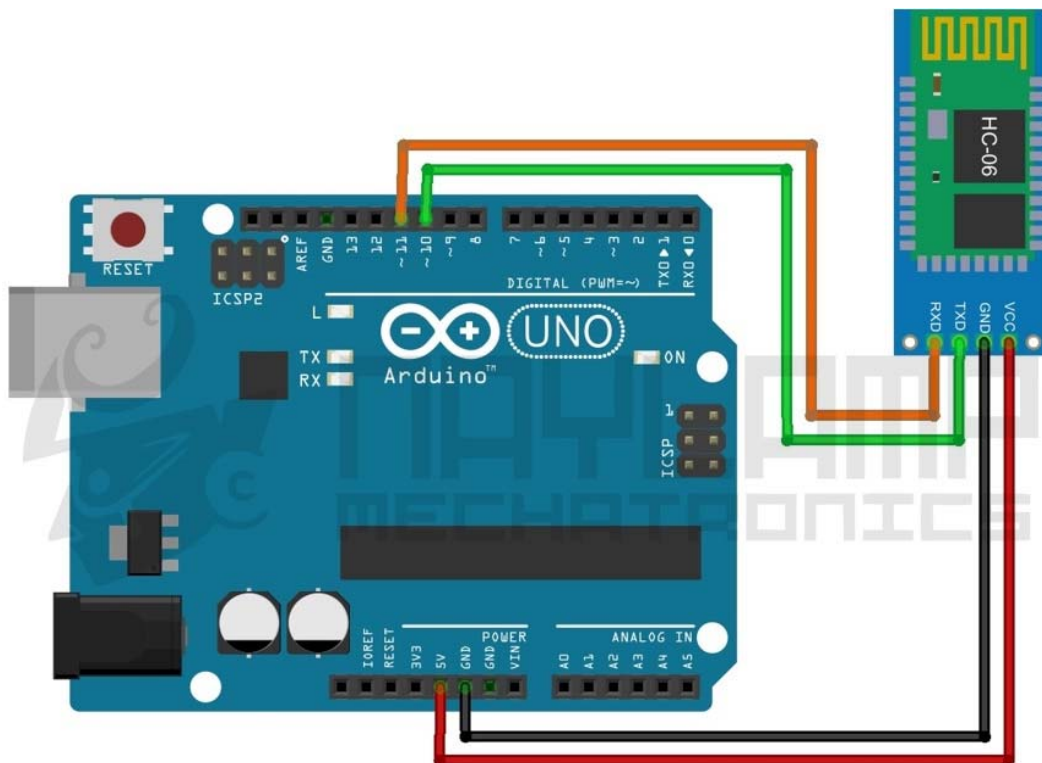
- En estos tutoriales una vez conectaremos el módulo Bluetooth a los terminales "oficiales" RX/TX y otras veces incluiremos la librería SoftwareSerial y lo conectaremos a los terminales 10 y 11.

### 3. PRACTICA 1: CONFIGURANDO EL MODULO HC-06

Materiales:

- Placa Arduino UNO R3
- Módulo HC-06
- Cables de conexión

Lo primero que vamos a realizar es la siguiente conexión con Arduino y el módulo HC-06. Observarse que la conexión se realiza en cruzado, el pin TX del BT irá al RX (pin 10) de Arduino y el RX del BT irá al TX de Arduino (pin 11)



Para enviar los comandos AT, nuestro HC-06 debe estar en Modo AT, esto significa sin conexión bluetooth con otro dispositivo o verificar que LED del HC-06 esté parpadeando.

Copiaremos el siguiente Sketch en Arduino

```
#include <SoftwareSerial.h> // Incluimos la librería SoftwareSerial
SoftwareSerial BT(10,11); // Definimos los pines RX y TX del Arduino conectados al Bluetooth. Ojo,
las conexiones están cruzadas entre la placa de arduino y el modulo BT

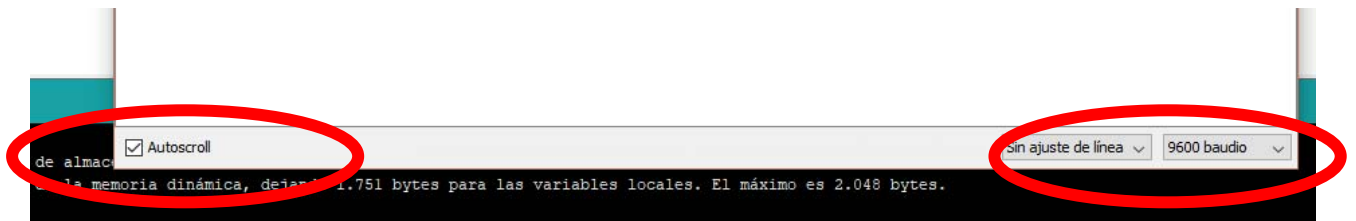
void setup()
{
  BT.begin(9600); // Inicializamos el puerto serie BT que hemos creado
  Serial.begin(9600); // Inicializamos el puerto serie
}

void loop()
```

```
{  
  if(BT.available()) // Si llega un dato por el puerto BT se envía al monitor serial  
  {  
    Serial.write(BT.read());  
  }  
  
  if(Serial.available()) // Si llega un dato por el monitor serial se envía al puerto BT  
  {  
    BT.write(Serial.read());  
  }  
}
```

Una vez hecho las configuraciones y conexión correspondientes, abrimos el Monitor serial del IDE de Arduino, .

En la parte inferior del monitor debemos escoger “Sin ajuste de línea” y la velocidad “9600 baud” (la velocidad por defecto de nuestro HC-06, si se lo ha cambiado poner la velocidad correspondiente)



Ahora emplearemos los siguientes comandos, para configurar el módulo.

Test de comunicación:

**Enviar:** AT

**Recibe:** OK

Si recibimos como respuesta un OK entonces podemos continuar, sino verificar las conexiones o los pasos anteriores.

Cambiar nombre de nuestro módulo HC-06:

Por defecto nuestro módulo bluetooth se llama “HC-06” o “Linvor” esto se puede cambiar con el siguiente comando AT

**Enviar:** AT+NAME<Nombre> Ejm: AT+NAMERobot

**Respuesta:** OKsetname

El nombre puede ser de hasta 20 caracteres como máximo



Cambiar Código de Vinculación:

Por defecto viene con el código de vinculación (Pin) "1234", para cambiarlo hay que enviar el siguiente comando AT

**Enviar:** AT+PIN<Pin> Ejm: AT+PIN1465

**Respuesta:** OKsetPIN

Necesitaremos instalar en nuestro dispositivo móvil una App que nos permita comprobar la conexión. Una muy sencilla y que funciona es BLUETHOOTH TERMINAL.

Desde el móvil localizamos nos conectamos al BT, al que le hemos cambiado nombre y contraseña.

Lanzamos un mensaje desde el terminal del móvil y debemos verlo en el monitor serie de Arduino.

Lanzamos un mensaje desde el monitor serie de Arduino y deberemos verlo en el móvil.

## 4. PRACTICA 2: ENCENDIDO Y APAGADO DE UN LED

Para esta práctica necesitaremos:

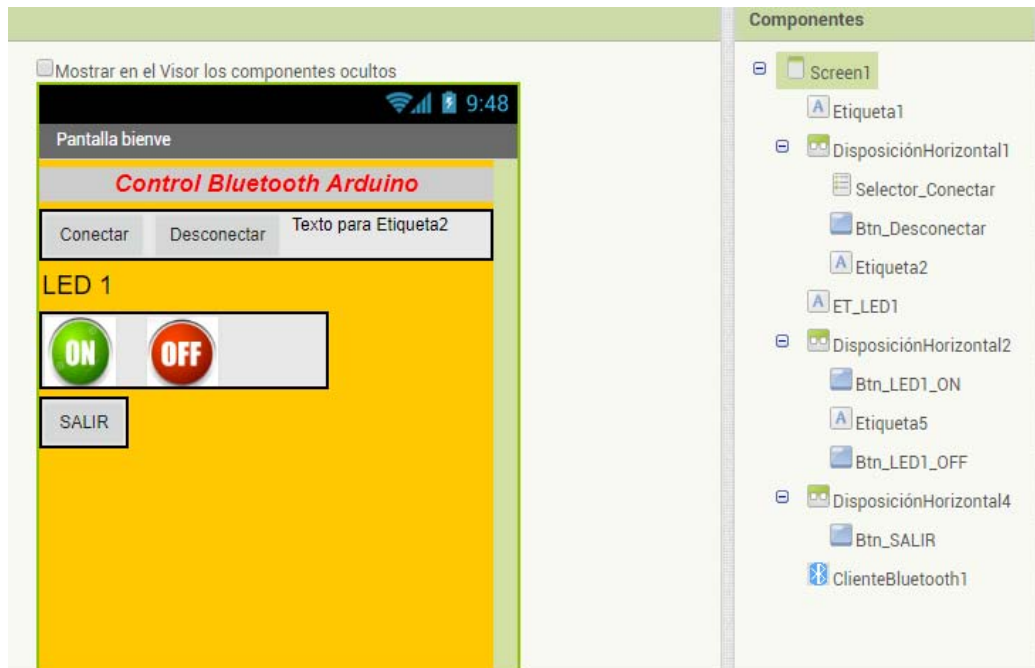
- App inventor (usuario y contraseña)
- Placa arduino UNO R3
- Diodo led
- Resistencia 330 ohm
- Módulo Bluetooth HC-05 o 06
- Cables de conexión
- Placa protoboard

### APP INVENTOR

Empezaremos creando la aplicación en app inventor. Va a tener:

- Un botón de selección de dispositivos bluetooth (Selector de lista)
- Un botón de desconexión
- Un botón de encendido del LED
- Un botón de apagado del LED
- Un botón para cerrar la aplicación.

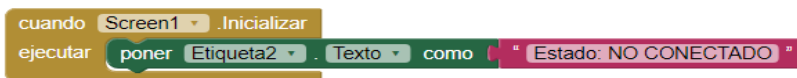
La disposición puede ser cualquiera, pero a modo de ejemplo crearemos la siguiente pantalla:



Muy importante: debemos añadir el componente no visible de “**ClienteBluetooth**”, que encontraremos en el menú CONECTIVIDAD.

Pasemos a dar funcionalidad a los bloques, para ello nos vamos al modo *Bloques*.

- Al iniciar la pantalla queremos que en ETIQUETA 2 ponga NO CONECTADO



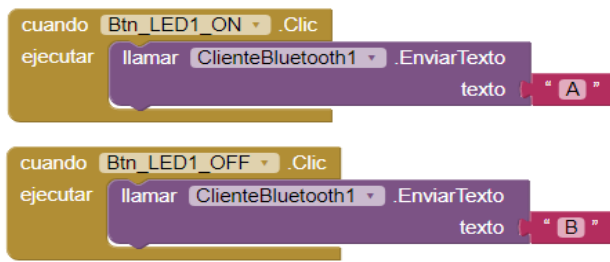
- Cargamos la lista de dispositivos BT disponibles en nuestro teléfono



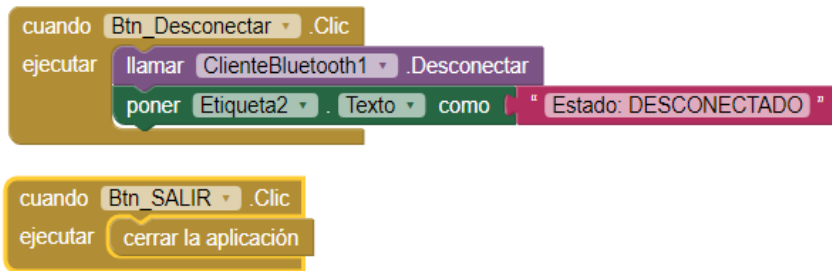
- Vamos a darle funcionalidad al botón conectar, para que una vez seleccionado el dispositivo BT se quede “almacenado” y la etiqueta2 cambie su estado a CONECTADO



- Damos funcionalidad a los botones de encendido y apagado de los led. Lo único que vamos a hacer es que cuando sean presionados manden una señal, en este caso de tipo TEXTO, que luego Arduino recibirá y ejecutará la acción de encender o apagar, en función de lo que hayamos pulsado.

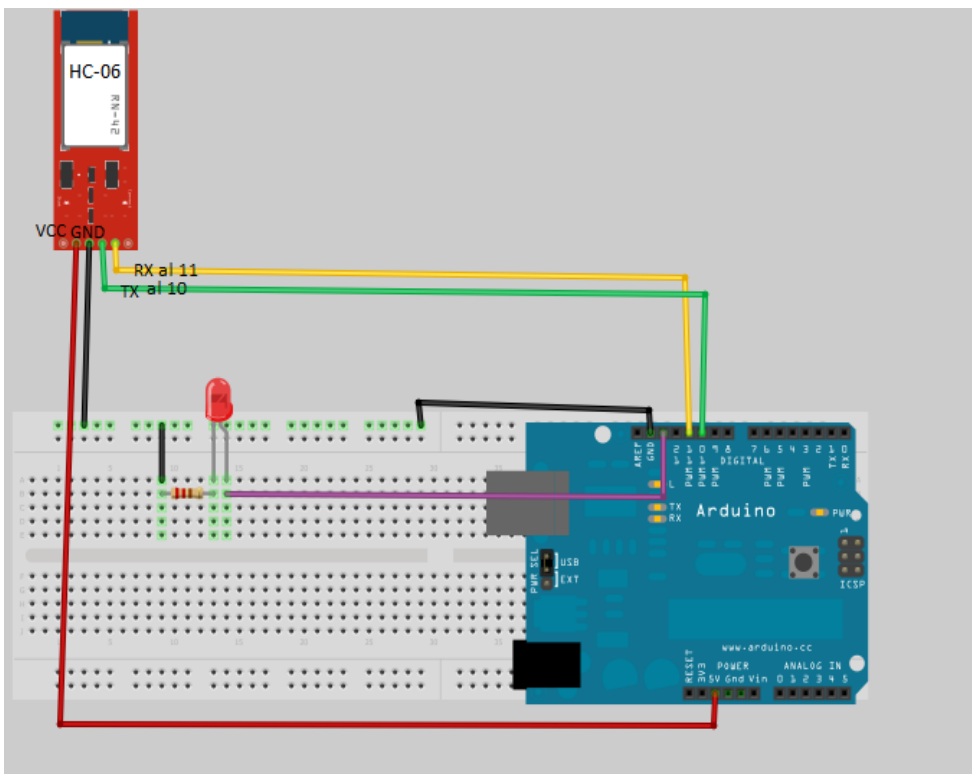


- Damos funcionalidad a los botones de DESCONECTAR y SALIR. El primero desconecta el BT y el segundo cierra la aplicación.



## ARDUINO

El montaje que tenemos que realizar el siguiente:



Y la programación de arduino:

```
Encender-apagar1-led$
#include <SoftwareSerial.h> // Incluimos la librería SoftwareSerial
SoftwareSerial BT(10,11); // Definimos los pines RX y TX del Arduino conectados al Bluetooth. Ojo, las conexiones están cruzadas entre la placa de arduino y el modulo BT
//el TX del BT va al 10
//el RX del BT va al 11

void setup()
{
  BT.begin(9600); // Inicializamos el puerto serie BT que hemos creado
  Serial.begin(9600); // Inicializamos el puerto serie
  pinMode(13, OUTPUT); // Definimos el pin 8 del LED
}
char dato; //definimos una variable de tipo caracter llamada dato, que nos sirve para leer y almacenar el valor que recibimos
void loop()
{
  if(BT.available()) // Si llega un dato por el puerto BT se envía al monitor serial
  {
    dato=(BT.read());//leo el valor del BT y lo guardo en led;
    Serial.println(dato);
    if (dato=='A'){
      digitalWrite(13,HIGH);
    }
    if (dato=='B'){
      digitalWrite(13,LOW);
    }
  }
}
```

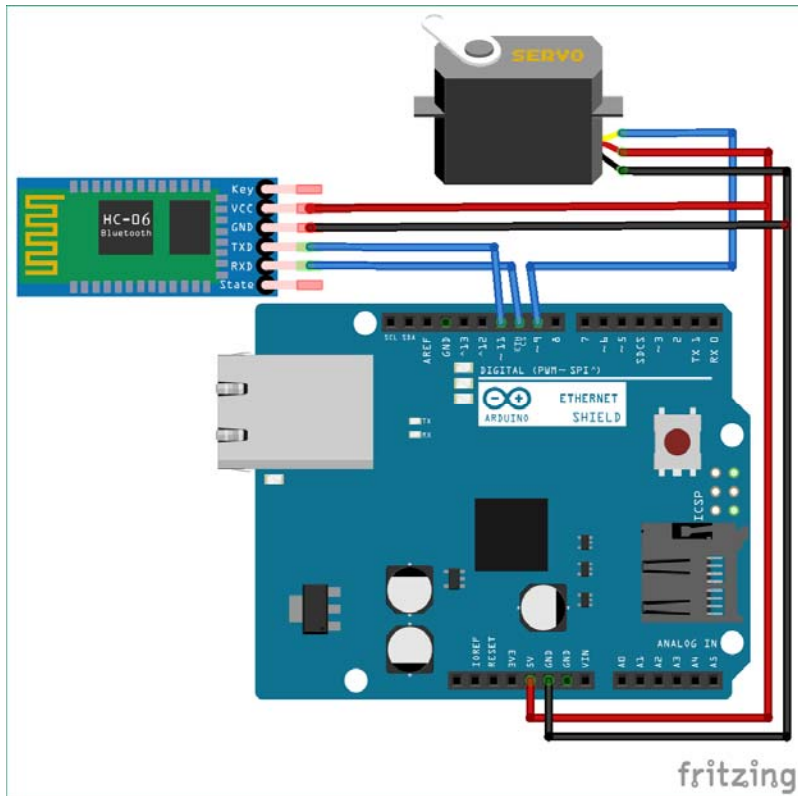
### ACTIVIDADES DE AMPLIACIÓN:

- Modifica la APP para saber el estado del LED en cada momento.
- Modifica los programas añadiendo un segundo LED que se encienda y apague al mismo tiempo que el primero.
- Modifica los programas para añadir un segundo LED que funcione de forma independiente al primero. *La solución a esta actividad está en la carpeta adjunta. Se ha modificado el programa para en lugar de trabajar con variables tipo char (caracteres) se trabaje con tipo int, para ello empleamos la codificación ASCII.*

## 5. CONTROL DE UN SERVO por BT

El segundo ejemplo consiste en el uso de PWM, para variar el ángulo de un servomotor y el. En este caso en particular, utilizamos deslizadores para enviar datos variables los cuales modificarán el ángulo del servo, pero también pondremos botones para posicionar el servo en una posición dada.

### El montaje en Arduino



El código de arduino sería el siguiente

p2-control\_servo Arduino 1.6.9

Archivo Editar Programa Herramientas Ayuda

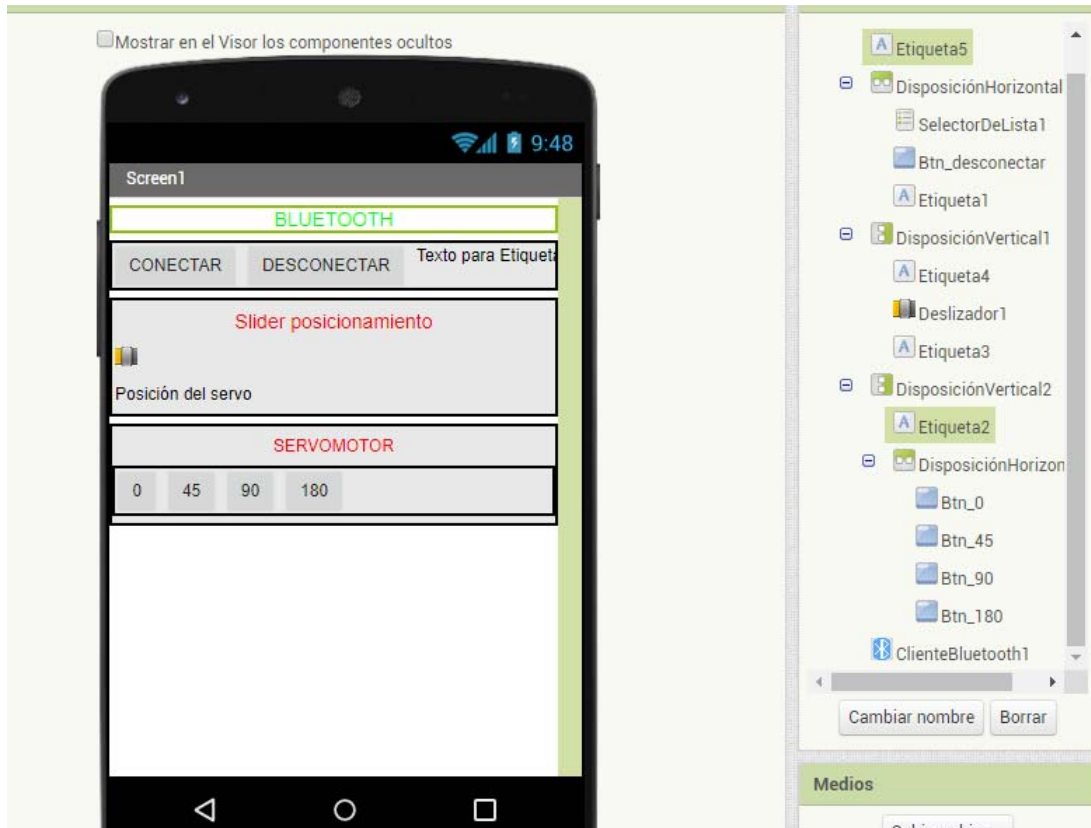
```

p2-control_servo $
#include <SoftwareSerial.h> // Incluimos la librería SoftwareSerial
SoftwareSerial BT(10,11); // Definimos los pines RX y TX del Arduino conectados al Bluetooth. Ojo,
#include <Servo.h>
Servo myservo;

void setup()
{
  BT.begin(9600); // Inicializamos el puerto serie BT que hemos creado
  Serial.begin(9600); // Inicializamos el puerto serie
  myservo.attach(9); //definimos el pin 9 al que hemos conectado el servo
}

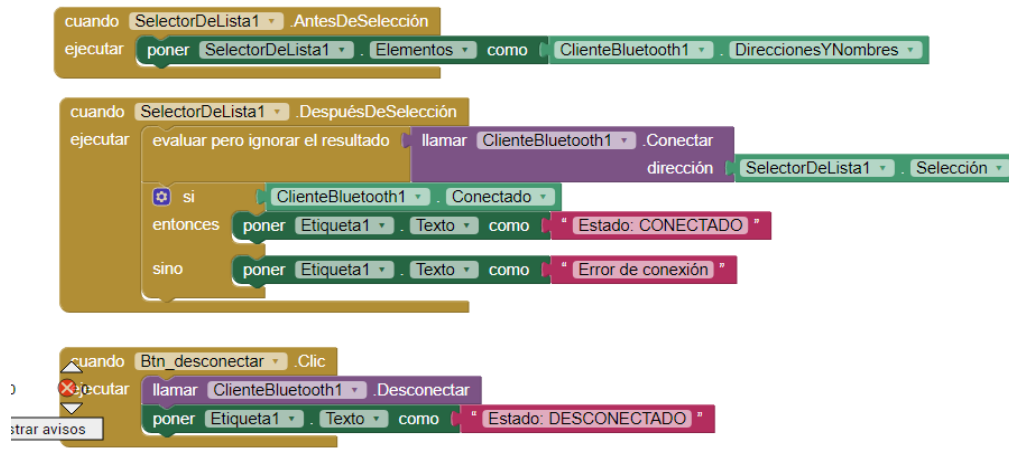
void loop() {
  if (BT.available()>0){
    int dato=BT.read();//leemos el dato que transmitimos vía BT, que es tipo int porque es un número
    Serial.println(dato);//comprobamos que el dato es el correcto
    myservo.write(dato);} //pasamos el dato al servo para que se posicione
}
    
```

Podemos realizar una interfaz a través de App inventor con la siguiente estructura:

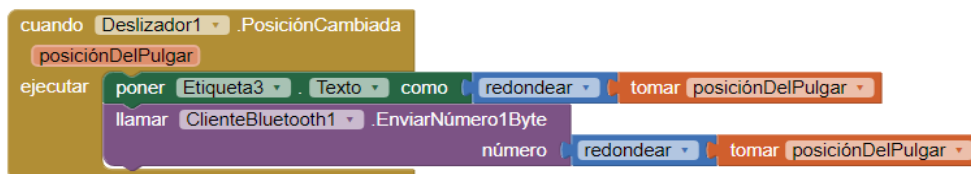


Y damos funcionalidad a los botones:

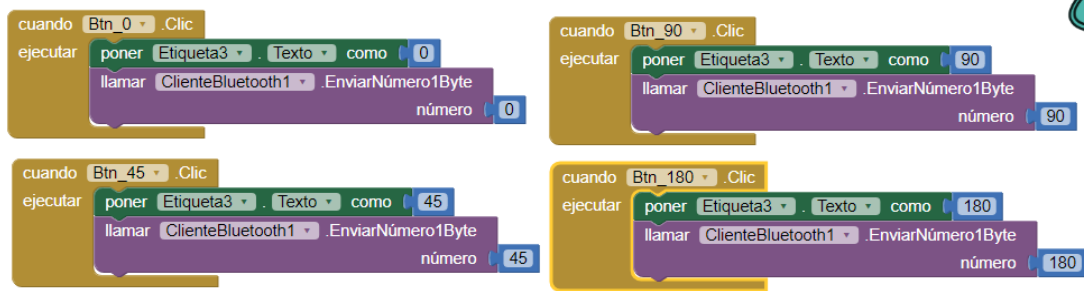
- a) La conexión con el BT se realiza del mismo modo que en el ejercicio anterior



- b) El posicionamiento del servo mediante el slider



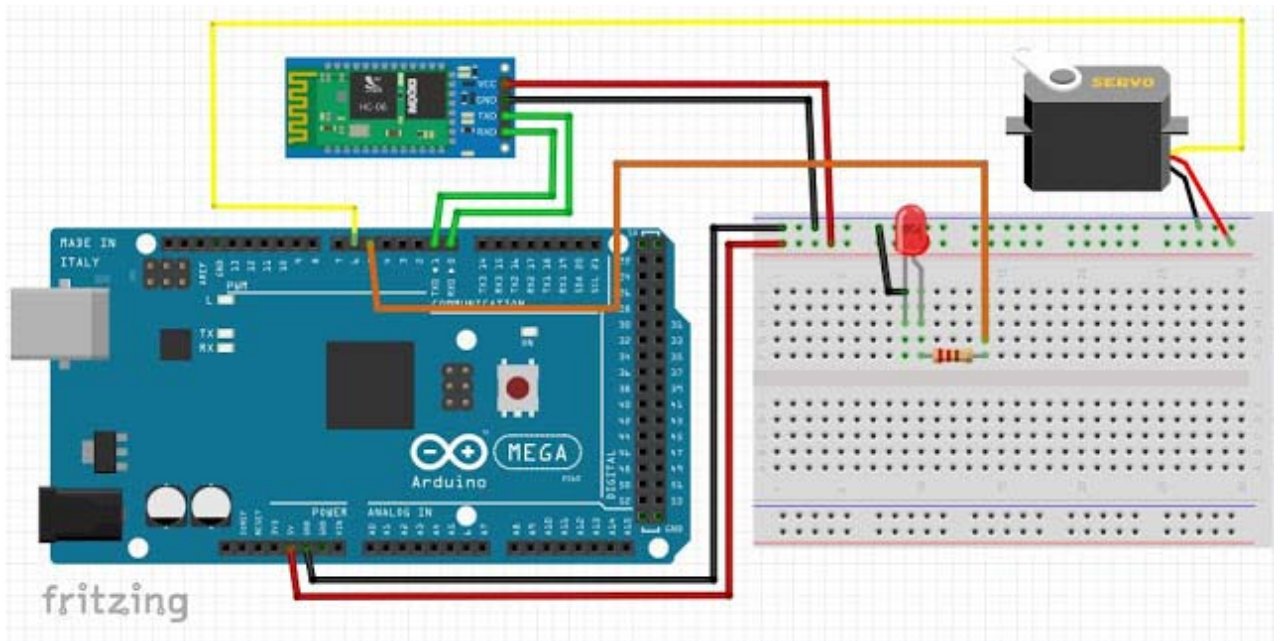
- c) El posicionamiento del servo mediante botones



### ACTIVIDAD DE AMPLIACIÓN

Sobre este programa podemos hacer una modificación que consiste en añadir un LED y modificar la intensidad de brillo con un slider.

EL montaje sería el siguiente, y se deja el link a la solución de la actividad que plantean en el blog "Ardu para todos"



<https://arduparatodos.blogspot.com/2017/06/modulo-bluetooth-hc-06-con-arduino-y.html>

## 6. CONTROL DE UN RGB POR BLUETOOTH

En este caso vamos controlar un diodo RGB desde el teléfono móvil. Esta práctica se puede plantear de diversas formas, de más sencilla a más compleja. Algunas de las actividades que se pueden desarrollar son:

- 1- Encendido de diodo ROJO, VERDE Y/O AZUL. Cada color tendrá dos botones de estado (ON/OFF). Es la más sencilla y similar a la anterior. En la carpeta de esta práctica podemos encontrar un PDF con su desarrollo.



- 2- Emplear SLIDERS, uno para color, de tal forma que, según la posición de los deslizadores así lucirá el diodo.
- 3- Emplear una matriz de color y que automáticamente el diodo se encienda en función del color elegido.

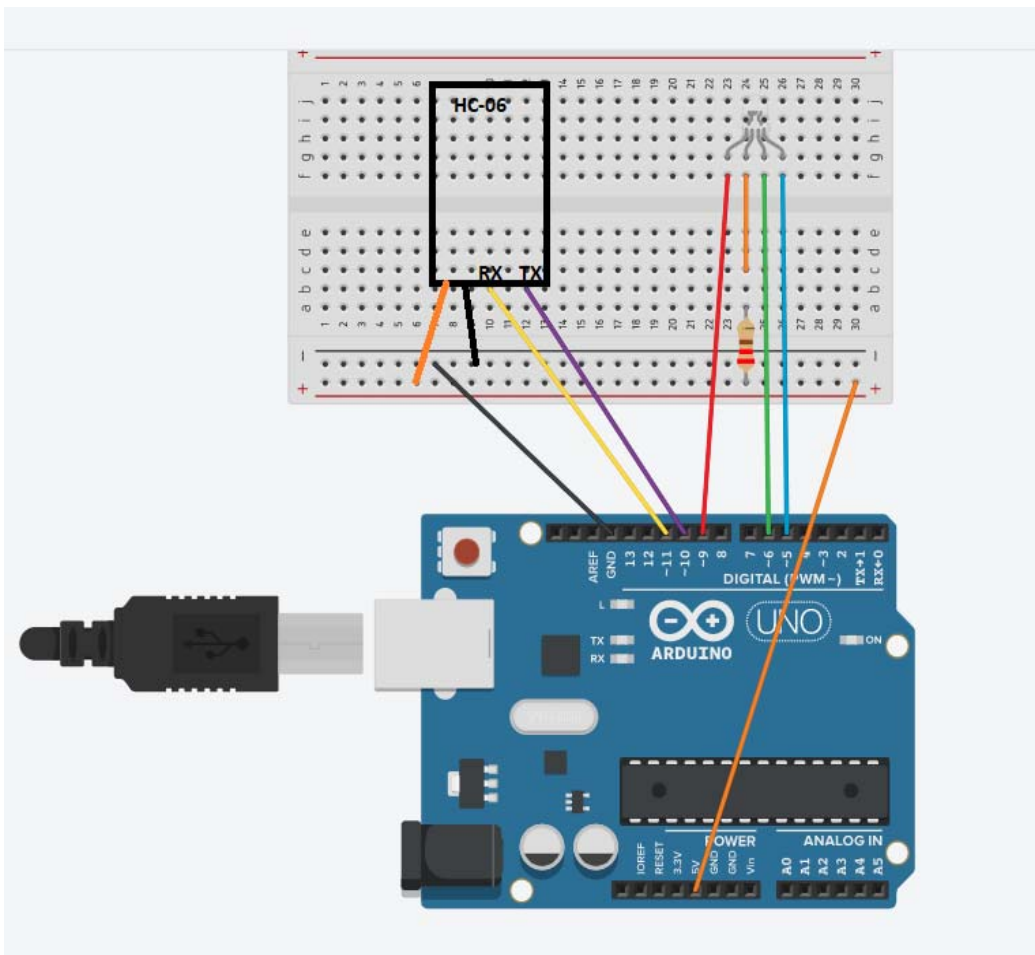
En este caso vamos a desarrollar la opción 2 y 3.

Los materiales que necesitaremos son:

- Placa arduino UNO r3
- Diodo RGB (ojo si es de ánodo o cátodo común, que la programación y conexión cambia). En este caso hemos usado uno de ÁNODO COMÚN.
- Resistencias 220  $\Omega$
- Modulo Bluetooth HC-06
- Cables de conexión.

En este caso vamos a empezar por la parte de conexionado (el módulo HC-06 mantiene la conexión de las prácticas anteriores) y que es el mismo para las 3 actividades planteadas.

### CONEXIONADO



### OPCIÓN 2: MEDIANTE SLIDERS

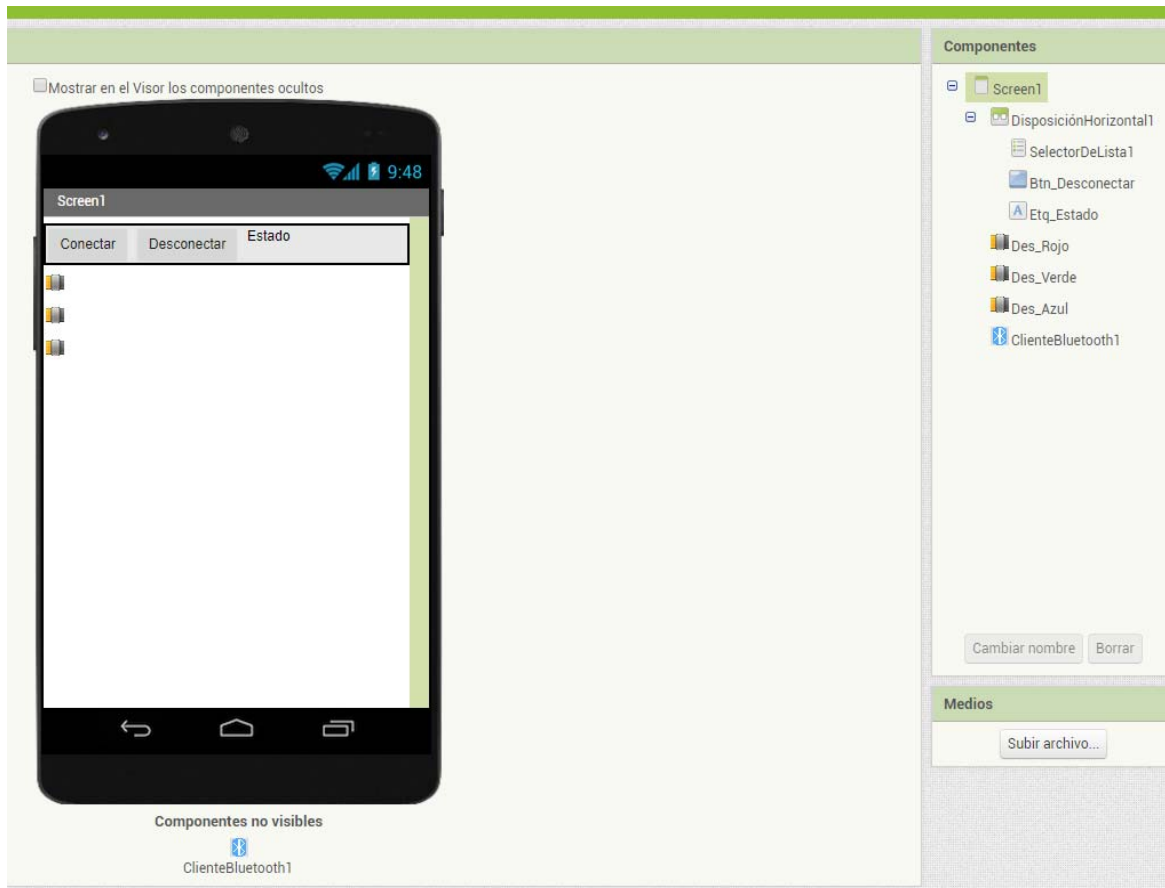
#### APP INVENTOR

La interfaz sería similar a la mostrada en la imagen:

Tiene:



- Disposición horizontal:
  - Selector de lista (CONECTAR), botón de desconectar, y etiqueta de estado
- 3 Deslizadores, valor min 0 valor máximo 255
- Componente no visible de Cliente BT



Y la programación:

-El BT es idéntico que en el caso anterior

```
cuando Screen1 . BotónAtrás
ejecutar poner Etq_Estado . Texto como " Estado: NO CONECTADO "

cuando SelectorDeLista1 . AntesDeSelección
ejecutar poner SelectorDeLista1 . Elementos como ClienteBluetooth1 . DireccionesYNombres

cuando SelectorDeLista1 . DespuésDeSelección
ejecutar evaluar pero ignorar el resultado llamar ClienteBluetooth1 . Conectar
dirección SelectorDeLista1 . Selección
si ClienteBluetooth1 . Conectado
entonces poner Etq_Estado . Texto como " Estado: CONECTADO "
sino poner Etq_Estado . Texto como " Error de conexión "
```

-Los deslizadores:



Lo que estamos haciendo es lo siguiente:

-La barra del deslizador cambiará de color, por la parte izquierda, a un color RGB de valor : la posición del pulgar, 0,0 (en el caso del rojo.)

- Se comprueba que el BT esté conectado

-Se envía un numero de 2 bytes por cada color (con un byte solo podemos tener 255 números, y en este caso nos hacen falta más), de tal forma que:

-Rojo irá de 1000 a 1255

-Verde: 2000 a 2255

-Azul: 3000 a 3255

Se puede tomar cualquier rango de valores, se ha tomado este por considerarse más sencillo, pero podría ser 000-255, 256 a 512 y 513 a 768.

## ARDUINO

RGB-Slider Arduino 1.8.7

Archivo Editar Programa Herramientas Ayuda

```
RGB-Slider$
#include <SoftwareSerial.h>

int bluetoothTx = 10;
int bluetoothRx = 11;

SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);

void setup()
{
  pinMode(5,OUTPUT); // blue pin of RGB LED
  pinMode(6,OUTPUT); // Green pin of RGB LED
  pinMode(9,OUTPUT); // Red pin of RGB LED

  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(9,LOW);
  //Iniciamos el puerto serie
  Serial.begin(9600);

  //Setup Bluetooth serial connection to android
  bluetooth.begin(9600);
}

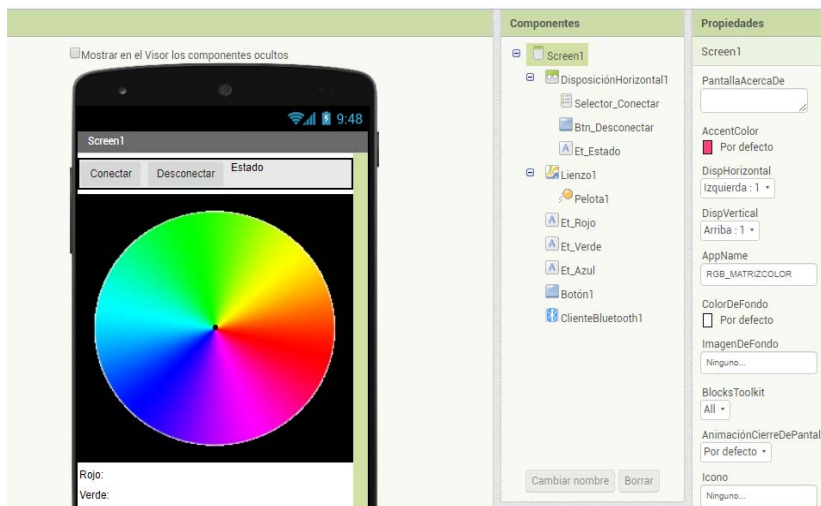
void loop()
{
  //Leemos el bluetooth y escribimos en el puerto serie, para comprobar los datos
  if(bluetooth.available()>= 2)//es el numero de bytes que tiene que recibir, 2, porque así lo hemos especificado en el APPINVENTOR
  {
    int color1 = bluetooth.read();
    Serial.println(color1);
    int color2 = bluetooth.read();
    Serial.println(color2);
    int color = (color2 *256) + color1; //Realizamos este calculo para obtener el valor de los 2bytes
    Serial.println(color);

    if (color >= 1000 && color <1255){
      int red = color;
      red = map(red, 1000,1255,0,255);
      red=255-red;//al usar un RGB de ánodo común debo "invertir" los colores
      analogWrite(9,red);
      Serial.print("red:");
      Serial.println(red);
      delay(10);
    }
  }
}
```

```
if (color >=2000 && color <2255){  
  int green = color;  
  green = map(green,2000,2255,0,255);  
  green=255-green;  
  analogWrite(6,green);  
  Serial.print("green:");  
  Serial.println(green);  
  delay(10);  
}  
  
if (color >=3000 && color < 3255){  
  int blue= color;  
  blue = map(blue, 3000, 3255,0,255);  
  blue=255-blue;  
  analogWrite(5,blue);  
  Serial.print("blue:");  
  Serial.println(blue);  
  delay(10);  
}  
  
}
```

## OPCIÓN 3: MEDIANTE UNA MATRIZ DE COLOR

El conexionado es el mismo y el programa de Arduino también. Tan sólo tenemos que modificar la App en appinventor, tal. y como se muestra en la figura:



Y vamos programando los botones:

Conexión del bluetooth:

```

cuando Btn_Desconectar .Clic
ejecutar
  llamar ClienteBluetooth1 .Desconectar
  poner Et_Estado .Texto como " Estado: DESCONECTADO "

cuando Screen1 .BotónAtrás
ejecutar
  poner Et_Estado .Texto como " Estado: NO CONECTADO "
  poner Et_Estado .ColorDeFondo como #000000
  poner Et_Estado .ColorDeTexto como #FF0000

cuando Selector_Conectar .AntesDeSelección
ejecutar
  poner Selector_Conectar .Elementos como ClienteBluetooth1 .DireccionesYNombres

cuando Selector_Conectar .DespuésDeSelección
ejecutar
  evaluar pero ignorar el resultado llamar ClienteBluetooth1 .Conectar
  dirección Selector_Conectar .Selección
  si ClienteBluetooth1 .Conectado
  entonces
    poner Et_Estado .Texto como " Estado: CONECTADO "
    poner Et_Estado .ColorDeFondo como #CCCCCC
    poner Et_Estado .ColorDeTexto como #00FF00
  sino
    poner Et_Estado .Texto como " Error de conexión "
    poner Et_Estado .ColorDeFondo como #FFCC00
  
```

Inicializamos 3 variables, una por cada color:

```

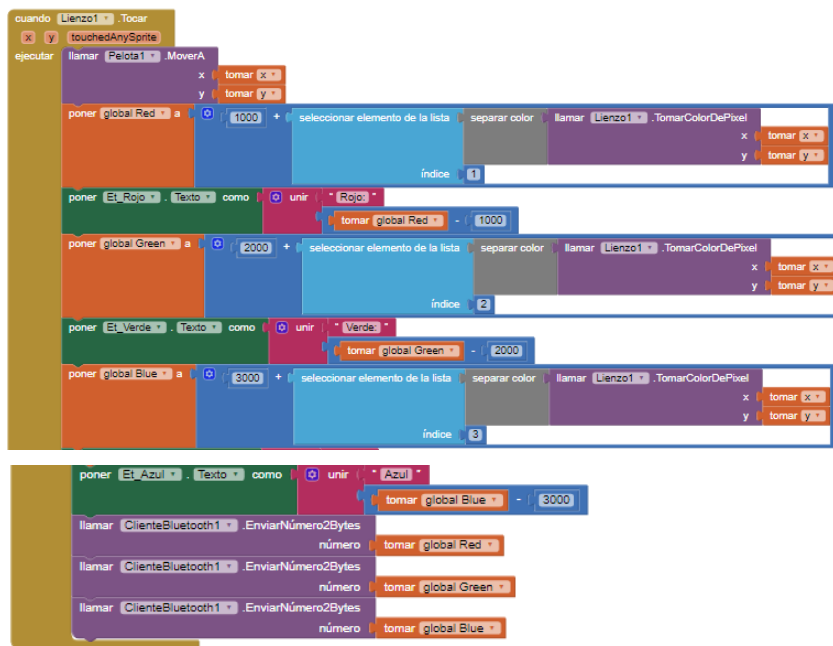
inicializar global Red como 0
inicializar global Green como 0
inicializar global Blue como 0
  
```

Al iniciar la aplicación vamos a situar la pelota en una zona negra del lienzo para que los LEDs estén apagados.

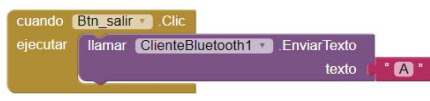
```

cuando Screen1 .Inicializar
ejecutar
  llamar Pelota1 .MoverA
  x 5
  y 100
  llamar ClienteBluetooth1 .EnviarNúmero2Bytes
  número tomar global Red
  llamar ClienteBluetooth1 .EnviarNúmero2Bytes
  número tomar global Green
  llamar ClienteBluetooth1 .EnviarNúmero2Bytes
  número tomar global Blue
  
```

Tomamos la posición de la pelota dentro del lienzo y tomamos una muestra del color, que convertimos al estandar RGB con una función propia del programa



Y por último programamos el botón de salir, como hemos hecho otras veces, pero apagando los LEDs primero.



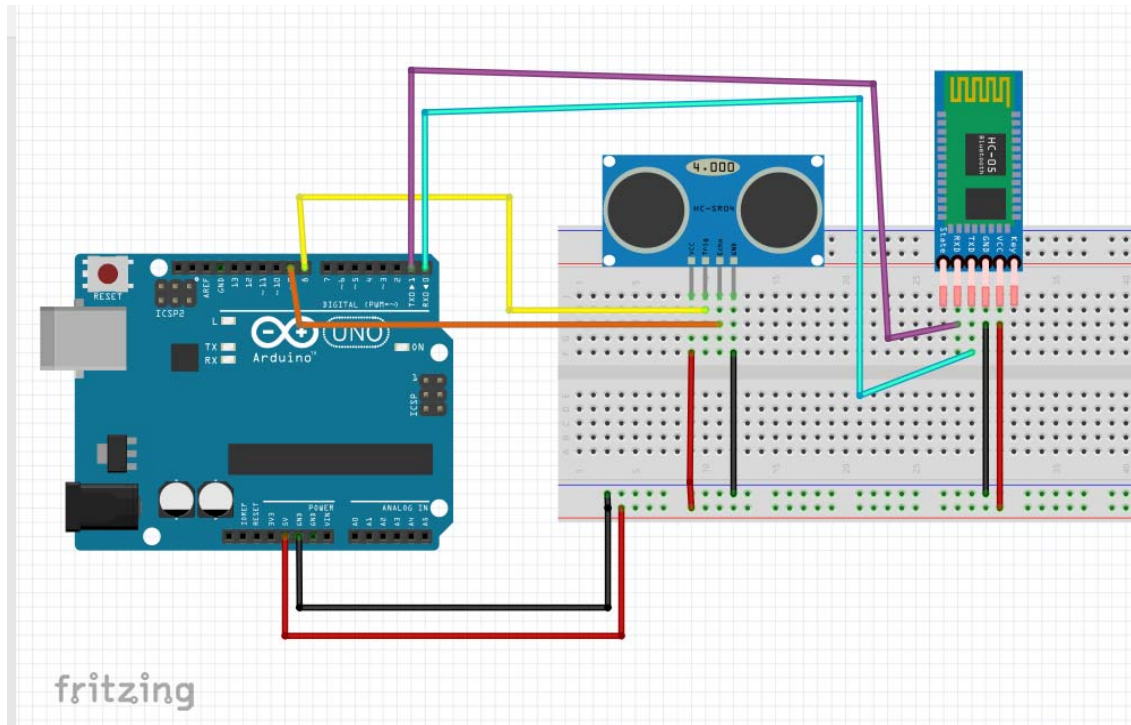
## 7. PRÁCTICA 4: ENVIANDO DATOS MEDIANTE EL SENSOR DE DISTANCIA (ultrasonido)

Mediante esta práctica vamos a ver cómo enviar datos desde Arduino a nuestra App. Este valor puede ser obtenido de una medición de una LDR, potenciómetro, sensor ultrasonido, temperatura....

En este caso vamos a usar el sensor de ultrasonido que puede medir distancias entre 2 y 700cm y consta de 4 pines: VCC, trigger, Echo y GND.

Para hacer más sencilla la programación, vamos a usar los pines **0** y **1** del Arduino para conectar el BT. Recordar que, **cuando hacemos la carga de programas, se usan estos pines, por lo que debemos DESCONECTAR el BT al subir el archivo a la placa.**

La conexión en Arduino se muestra en la imagen.



La programación ARDUINO sería

```
medidor_distancia
int Echo =9;//Declaramos el pin 5 como Echo
int Trigger= 8;//Declaramos el pin 6 como Trigger

double distancia;
long tiempo;
void setup() {
  Serial.begin (9600);

  pinMode(Trigger, OUTPUT);
  pinMode(Echo, INPUT);
}

void loop() {
  digitalWrite (Trigger, LOW);
  delayMicroseconds (5);
  digitalWrite(Trigger, HIGH);
  delayMicroseconds (10);
  digitalWrite (Trigger, LOW);

  tiempo=pulseIn(Echo, HIGH) ;//medimos el tiempo entre pulsos en microseg

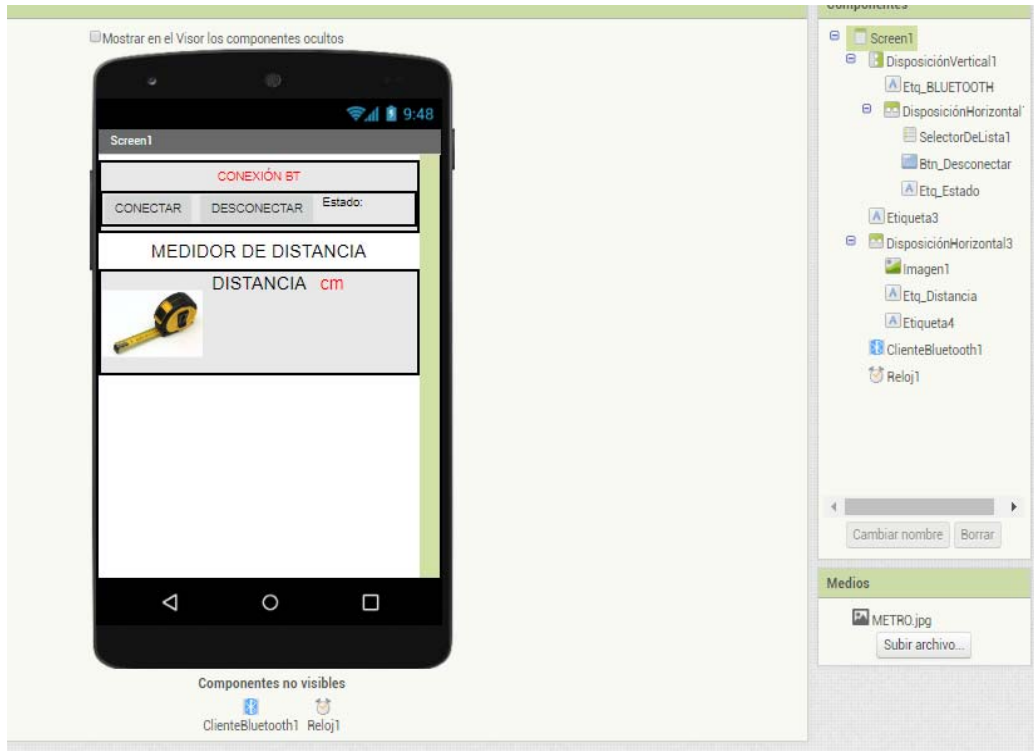
  distancia= tiempo / (29.2*2); //convertimos la distancia a cm, ya q esta en microseg, sabiendo que la velocidad es de 343m/s

  Serial.println(distancia); //Monitoreamos la distancia captada por el sensor y enviamos

  delay (250);
}
```

La interfaz en App inventor, tiene la siguiente estructura





Como características señalar:

- a) Hemos insertado y programado el cliente BT como en casos anteriores.
- b) Hemos insertado un RELOJ, que es un componente no visible, para que nos actualice los datos a modo de temporizador, con valor 1000 (1 segundo)

La programación es la siguiente:

- Programación del BT

```

cuando Screen1 . BotónAtrás
ejecutar poner Etq_Estado . Texto como " Estado: NO CONECTADO "

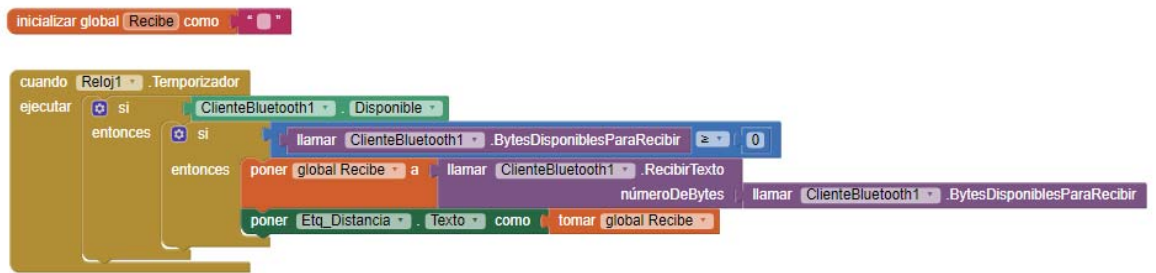
cuando SelectorDeLista1 . AntesDeSelección
ejecutar poner SelectorDeLista1 . Elementos como ClienteBluetooth1 . DireccionesYNombres

cuando SelectorDeLista1 . DespuésDeSelección
ejecutar evaluar pero ignorar el resultado llamar ClienteBluetooth1 . Conectar
                                     dirección SelectorDeLista1 . Selección
si ClienteBluetooth1 . Conectado
entonces poner Etq_Estado . Texto como " Estado: CONECTADO "
sino poner Etq_Estado . Texto como " Error de conexión "

cuando Btn_Desconectar . Clic
ejecutar llamar ClienteBluetooth1 . Desconectar
poner Etq_Estado . Texto como " Estado: DESCONECTADO "
    
```

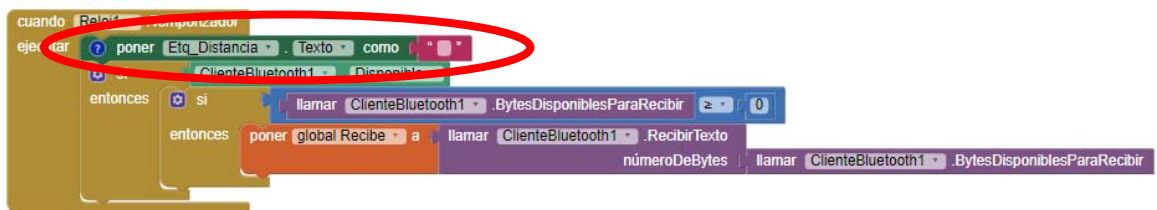
- Programación del reloj.





Si probamos la aplicación, podemos ver que en la ETQ\_DISTANCIa nos aparecen más distancias que la última. Esto se debe a que el tamaño de la ETQ es más grande que el de texto y nos “coge” más texto. Para solucionarlo tenemos dos opciones:

- A) Modificar el tamaño (pixeles) de la etiqueta, hasta ajustarlo
- B) Insertar la siguiente línea de código en el reloj, para que “borre” las mediciones anteriores



Aquí podemos ver un ejemplo de aplicación de este circuito ampliado, con una alarma cuando se sobrepasa una distancia

<https://www.youtube.com/watch?v=1JPN5sKMa0U>

## 8. WEBGRAFÍA

- <https://www.prometec.net/bt-hc06/>
- [https://naylorlampmechatronics.com/blog/15\\_Configuraci%C3%B3n--del-m%C3%B3dulo-bluetooth-HC-06-usa.html](https://naylorlampmechatronics.com/blog/15_Configuraci%C3%B3n--del-m%C3%B3dulo-bluetooth-HC-06-usa.html)
- [https://www.youtube.com/watch?v=LgDr\\_vWOQb0](https://www.youtube.com/watch?v=LgDr_vWOQb0)
- <http://kio4.com/appinventor/9bluetootharduino.htm>