

EDU-CIAA-NXP y BLE 4.0 HM10

Contenido

Licencia	1
Conectar EDU-CIAA-NXP y módulo HM10	1
Probar conexión entre la EDU-CIAA-NXP y el módulo HM10 mediante comandos AT	2
Utilizar el módulo HM10 desde un Smartphone Android con la aplicación “BLE Scanner”	6
Uso de la aplicación para Android “Bluetooth Serial Terminal”	10
Aplicaciones Android con MIT App Inventor 2	15
Programa HM10_LED	15
Descargar la aplicación “MIT AI2 Companion”	15
Ingresar en ApInventor	15
Crear una nueva aplicación	16
Construir la interfaz gráfica	17
Programar el comportamiento de los elementos de la interfaz gráfica mediante bloques encastrables	28
Comprobar su funcionamiento en el Smartphone	35
Programa HM10_LED_BOTON	40
Modificar la interfaz gráfica	40
Agregar el comportamiento de recepción	41
Comprobar su funcionamiento en el Smartphone	43
Bibliografía	44

Licencia

“Aplicación Android con MIT App Inventor 2 y BLE 4.0 HM10” por Ing. Eric Pernia, se distribuye bajo una Licencia Creative Commons Atribución-CompartirIgual 4.0 Internacional.



Conectar EDU-CIAA-NXP y módulo HM10

La conexión entre la placa EDU-CIAA-NXP y el módulo HM10 es muy sencilla. El HM10 se alimenta con 3.3V y los pines de UART Rx y Tx se conectan a los pines 232_RX y 232_TX de la EDU-CIAA (de forma cruzada, Tx a Rx):

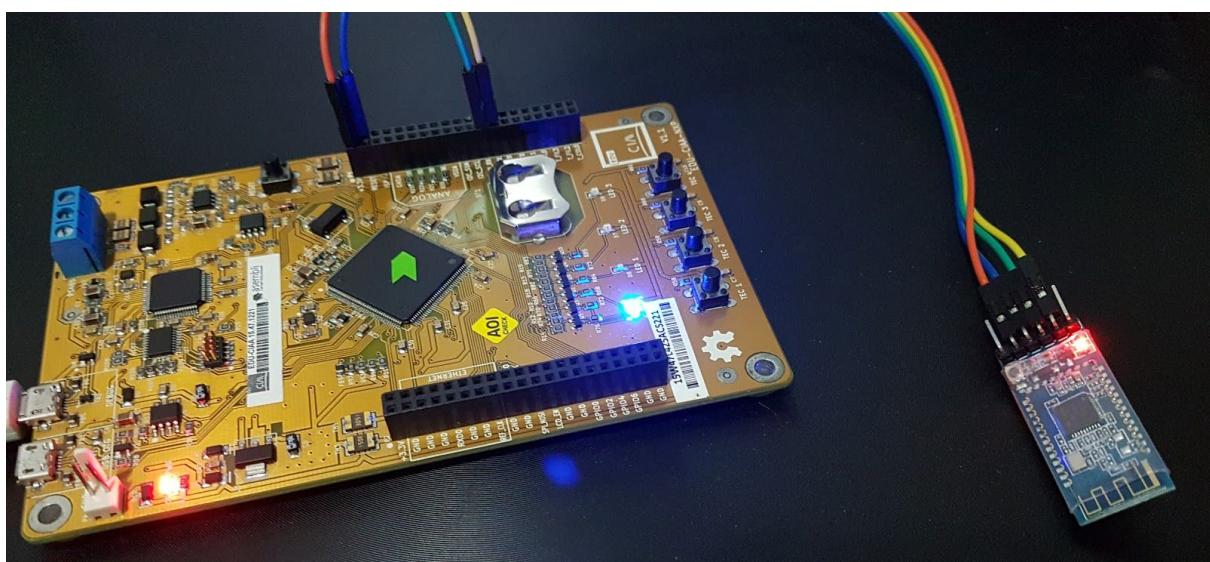
HM10 --- EDU-CIAA-NXP

VCC --- 3.3V

GND --- GND

TXD --- 232_RX

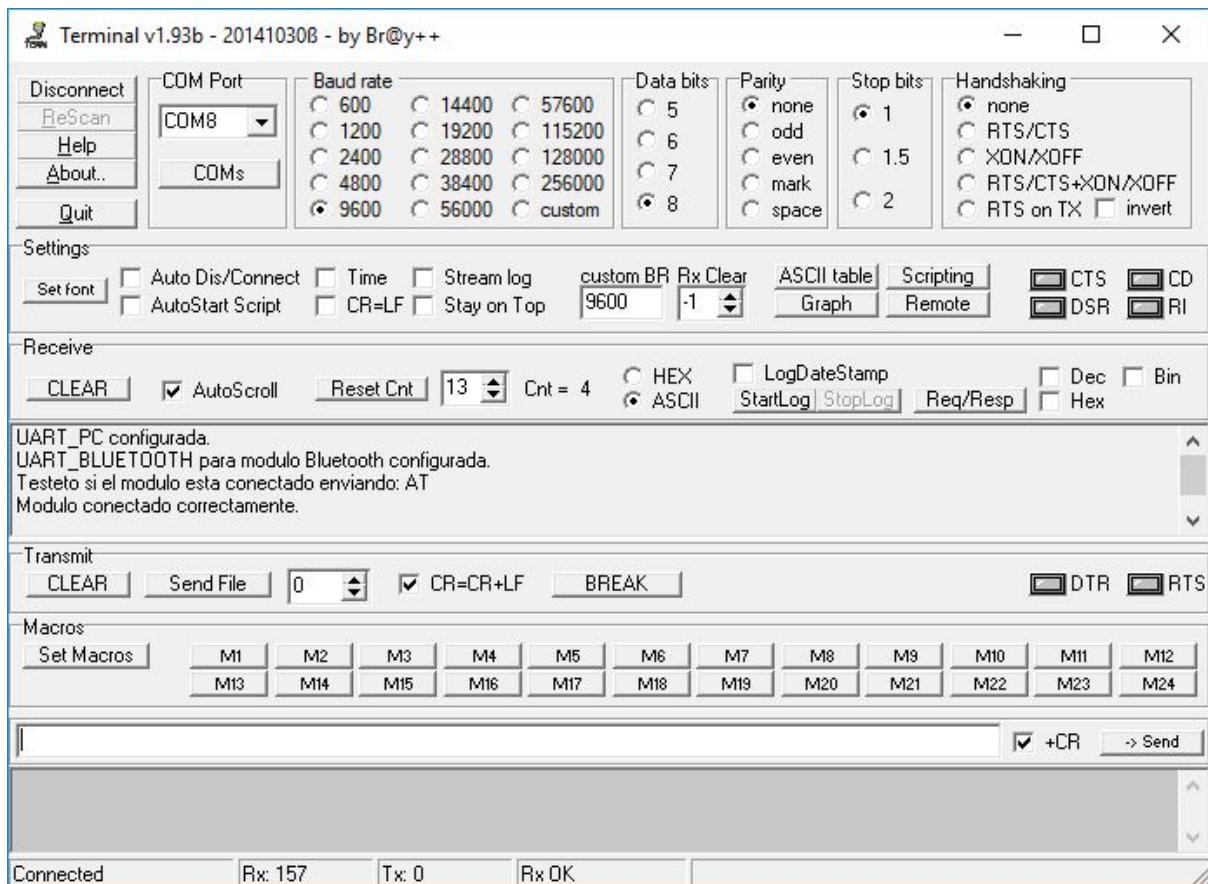
RXD --- 232_TX



Probar conexión entre la EDU-CIAA-NXP y el módulo HM10 mediante comandos AT

Para probar el módulo bluetooth se utilizará un programa que realizará un puente entre la UART de la EDU-CIAA-NXP que se conecta mediante el USB nombrado Debug (nombrada UART_USB en la biblioteca sAPI) y la UART que sale por los pines 232_TX y 232_RX del conector P1 de la EDU-CIAA-NXP (nombrada UART_232 en la biblioteca sAPI). De esta manera todo lo escrito desde la PC usando una Terminal Serie será enviado al módulo Bluetooth y todo lo que este responda llegará a la Terminal Serie también.

Al iniciar, el programa envía el comando AT y espera recibir un OK de parte del módulo HM10 para detectar si el mismo está conectado.



En el caso particular de recibir el carácter 'h' en la UART_232 (UART conectada al bluetooth) enciende el LEDB y si recibe 'l' lo apaga además de enviarlo a la UART conectada a la PC (esto se utilizará después).

Con TEC1 de la EDU-CIAA-NXP se le envía al módulo HM10 comando AT+HELP que retorna la lista de comandos AT soportados por el módulo.

Terminal v1.93b - 20141030B - by Br@y++

Disconnect **ReScan** **Help** **About..** **Quit**

COM Port: COM8 | **Baud rate**: 9600 | 600 | 1200 | 2400 | 4800 | 14400 | 19200 | 28800 | 38400 | 57600 | 115200 | 128000 | 256000 | custom

Data bits: 8 | **Parity**: none | odd | even | mark | space | **Stop bits**: 1 | 1.5 | 2 | **Handshaking**: none | RTS/CTS | XON/XOFF | RTS/CTS+XON/XOFF | RTS on TX | invert

Settings: Set font | Auto Dis/Connect | Time | Stream log | custom BR | Rx Clear | ASCII table | Scripting | CTS | CD | AutoStart Script | CR=LF | Stay on Top | 9600 | -1 | Graph | Remote | DSR | RI

Receive: **AutoScroll** | **Reset Cnt**: 13 | **Cnt =** 60 | **HEX** | **ASCII** | LogDateStamp | StartLog | StopLog | Req/Resp | Dec | Bin | Hex

UART_PC configurada.
UART_BLUETOOTH para modulo Bluetooth configurada.
Testeo si el modulo esta conectado enviando: AT
Modulo conectado correctamente.

* Command	Description
* AT	Check if the command terminal work normally
* AT+RESET	Software reboot
* AT+VERSION	Get firmware, bluetooth, HCI and LMP version
* AT+HELP	List all the commands
* AT+NAME	Get/Set local device name
* AT+PIN	Get/Set pin code for pairing
* AT+PASS	Get/Set pin code for pairing
* AT+BAUD	Get/Set baud rate
* AT+LADDR	Get local bluetooth address
* AT+ADDR	Get local bluetooth address
* AT+DEFAULT	Restore factory default
* AT+RENEW	Restore factory default
* AT+STATE	Get current state
* AT+PWRM	Get/Set power on mode(low power)
* AT+POWE	Get/Set RF transmit power
* AT+SLEEP	Sleep mode
* AT+ROLE	Get/Set current role.
* AT+PARI	Get/Set UART parity bit.
* AT+STOP	Get/Set UART stop bit.
* AT+START	System start working.
* AT+IMME	System wait for command when power on.
* AT+IBEAM	Switch iBeacon mode.
* AT+IBE0	Set iBeacon UUID 0.
* AT+IBE1	Set iBeacon UUID 1.
* AT+IBE2	Set iBeacon UUID 2.
* AT+IBE3	Set iBeacon UUID 3.
* AT+MARJ	Set iBeacon MARJ .
* AT+MINO	Set iBeacon MINO .
* AT+MEA	Set iBeacon MEA .
* AT+NOTI	Notify connection event .
* AT+UUID	Get/Set system SERVER_UUID .
* AT+CHAR	Get/Set system CHAR_UUID .

* Note: (M) = The command support slave mode only. *
* For more information, please visit <http://www.bolutek.com> *
* Copyright@2013 www.bolutek.com. All rights reserved. *

Transmit: CLEAR | Send File | 0 | CR=CR+LF | BREAK | DTR | RTS

Macros: Set Macros | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 | M9 | M10 | M11 | M12 | M13 | M14 | M15 | M16 | M17 | M18 | M19 | M20 | M21 | M22 | M23 | M24

+CR | → Send

Connected | Rx: 3340 | Tx: 0 | Rx OK

Con TEC3 se envía el texto “LED_ON\r\n” al módulo bluetooth y con TEC4 “LED_OFF\r\n” (esto se utilizará después).

Programa de ejemplo:

```
include "sapi.h"          // <= Biblioteca sAPI 0.5.1
#include <string.h>

#define UART_PC           UART_USB
#define UART_BLUETOOTH    UART_232

bool_t hm10bleTest( int32_t uart );
void hm10blePrintATCommands( int32_t uart );

int main( void )
{
    boardConfig();
    uartConfig( UART_PC, 9600 );
    uartWriteString( UART_PC, "UART_PC configurada.\r\n" );

    uartConfig( UART_BLUETOOTH, 9600 );
    uartWriteString( UART_PC, "UART_BLUETOOTH para modulo Bluetooth configurada\r\n." );

    uint8_t data = 0;

    uartWriteString( UART_PC, "Testeo si el modulo esta conectado enviando: AT\r\n");
    if( hm10bleTest( UART_BLUETOOTH ) ){
        debugPrintlnString( "Modulo conectado correctamente." );
    }

    while( TRUE ) {

        // Si leo un dato de una UART lo envio a al otra (bridge)
        if( uartReadByte( UART_PC, &data ) ) {
            uartWriteByte( UART_BLUETOOTH, data );
        }
        if( uartReadByte( UART_BLUETOOTH, &data ) ) {
            if( data == 'h' ) {
                gpioWrite( LEDB, ON );
            }
            if( data == 'l' ) {
                gpioWrite( LEDB, OFF );
            }
            uartWriteByte( UART_PC, data );
        }

        // Si presiono TEC1 imprime la lista de comandos AT
        if( !gpioRead( TEC1 ) ) {
            hm10blePrintATCommands( UART_BLUETOOTH );
        }

        // Si presiono TEC3 enciende el led de la pantalla de la app
        if( !gpioRead( TEC3 ) ) {
            uartWriteString( UART_BLUETOOTH, "LED_ON\r\n" );
            delay(500);
        }
        // Si presiono TEC4 apaga el led de la pantalla de la app
        if( !gpioRead( TEC4 ) ) {
            uartWriteString( UART_BLUETOOTH, "LED_OFF\r\n" );
        }
    }
}
```

```
        delay(500);
    }
}

return 0;
}

bool_t hm10bleTest( int32_t uart )
{
    uartWriteString( uart, "AT\r\n" );
    return waitForReceiveStringOrTimeoutBlocking(
        uart, "OK\r\n", strlen("OK\r\n"), 50 );
}

void hm10blePrintATCommands( int32_t uart )
{
    delay(500);
    uartWriteString( uart, "AT+HELP\r\n" );
}
```

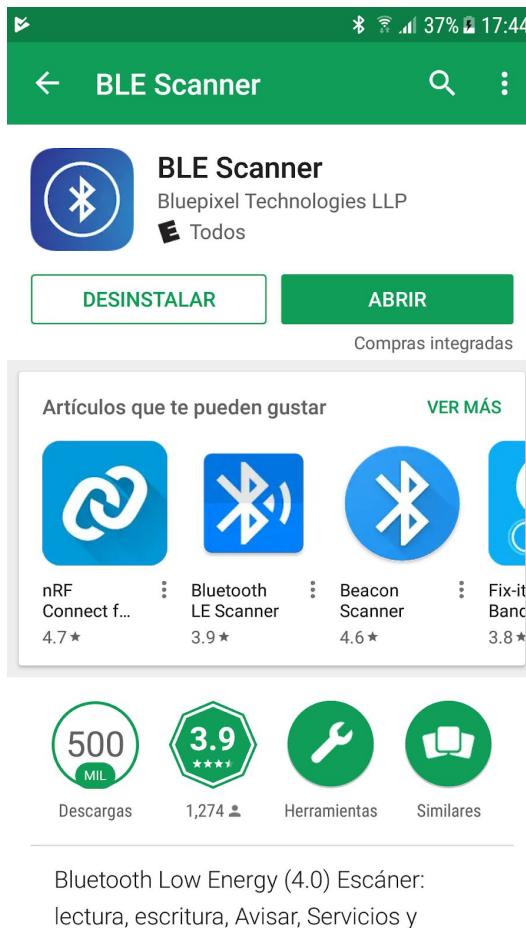
Descargar este programa y probar el funcionamiento de los comandos AT desde una Terminal Serie, los parámetros de conexión son 9600 baudios, 8 bits de datos, sin paridad y 1 bit de stop (9600, 8N1). Una cosa importante a realizar es cambiarle el nombre al dispositivo para luego no confundirse mediante el comando AT+NAMEnombre.

NOTA: *El módulo HM10 solamente responde ante comandos AT cuando NO está conectado a otro dispositivo (por ejemplo un celular).*

Utilizar el módulo HM10 desde un Smartphone Android con la aplicación “BLE Scanner”

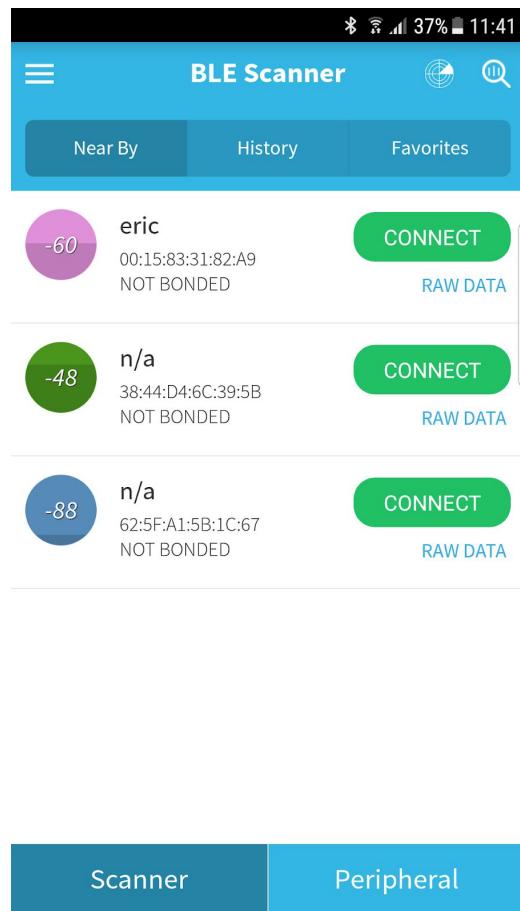
Esta aplicación permite detectar un módulo bluetooth, conectarse al mismo para observar sus Servicios y Características, utilizarlos, saber la potencia de transmisión MAC Address, entre otras cosas.

Se instala desde el Google Play Store:

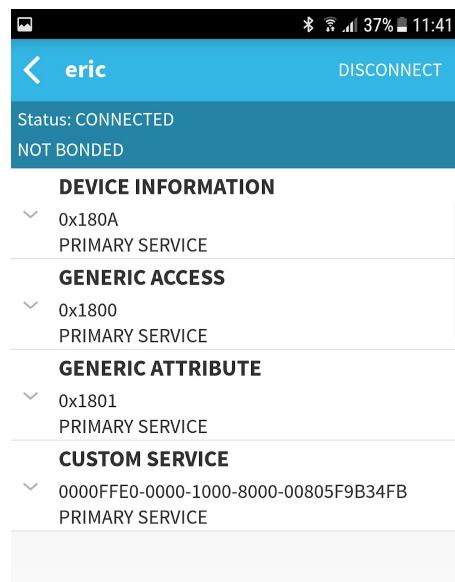


Bluetooth Low Energy (4.0) Escáner:
lectura, escritura, Avisar, Servicios y

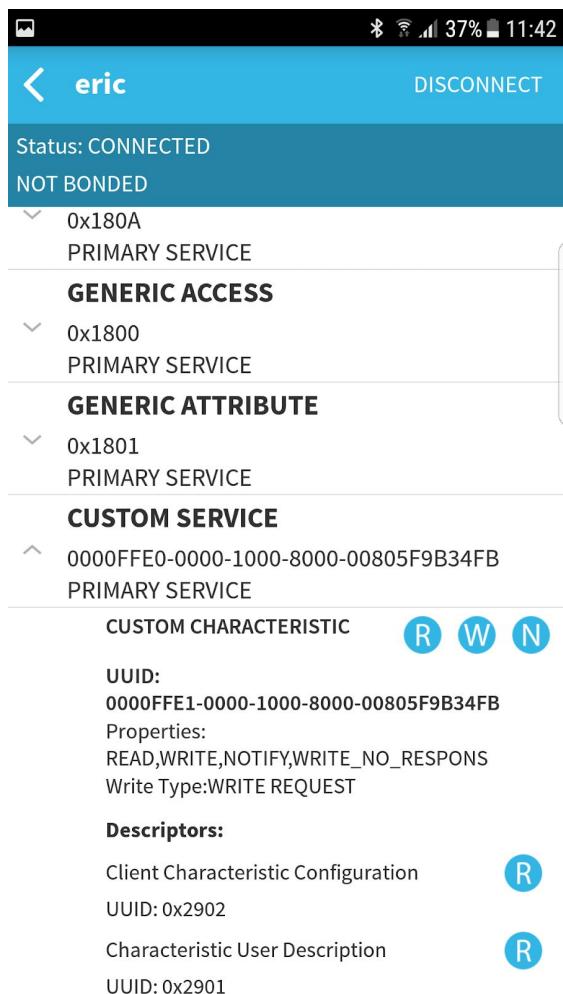
Para comprobar la conexión con el Smartphone, abrir la aplicación BLE Scanner y luego presionar el ícono de lupa para buscar nuevos dispositivos:



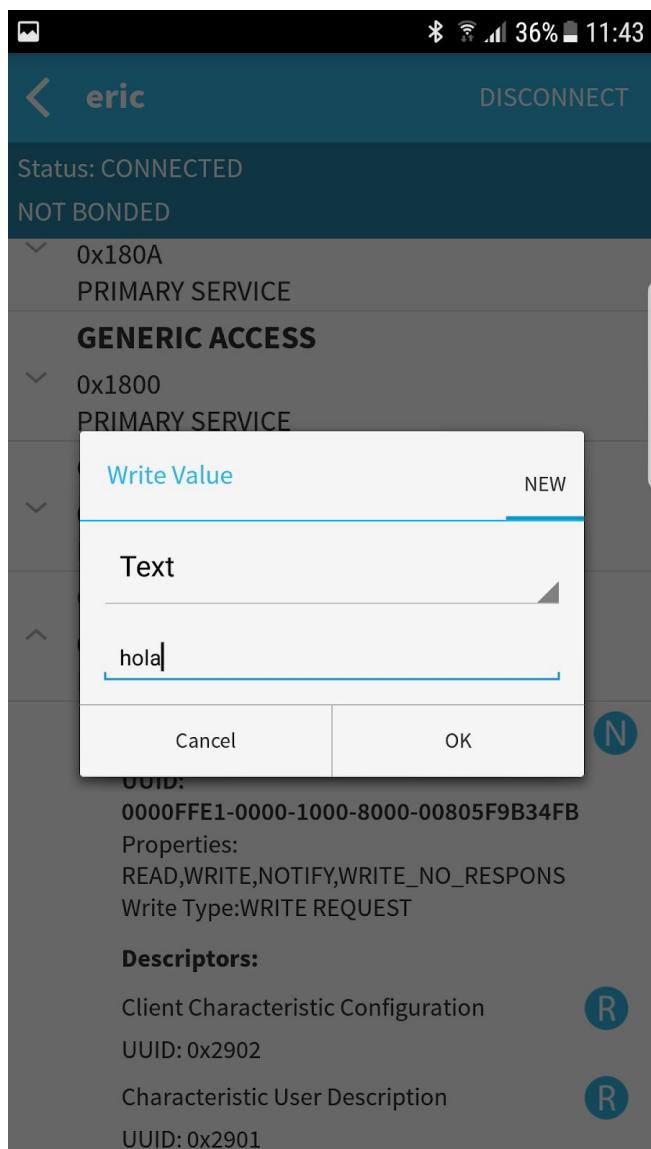
Presionar el botón CONNECT del dispositivo deseado, (en la pantalla capturada de ejemplo es “eric”, que es el nombre elegido al realizar el comando AT+NAME). De esta manera, se muestran los **Servicios** que brinda el módulo HM10:



Seleccionar el servicio personalizado “CUSTOM SERVICE” para mostrar sus Características:



Presionar el ícono con la “W” para escribir sobre la característica personalizada “CUSTOM CHARACTERISTIC”, escribir algún texto (por ejemplo “hola”) y presionar “OK”:



De esta manera llegará “holo” a la Terminal Serie en la PC.

Esto comprueba el mecanismo que utiliza el módulo bluetooth para enviar y recibir datos a través de la característica personalizada.

Uso de la aplicación para Android “Bluetooth Serial Terminal”

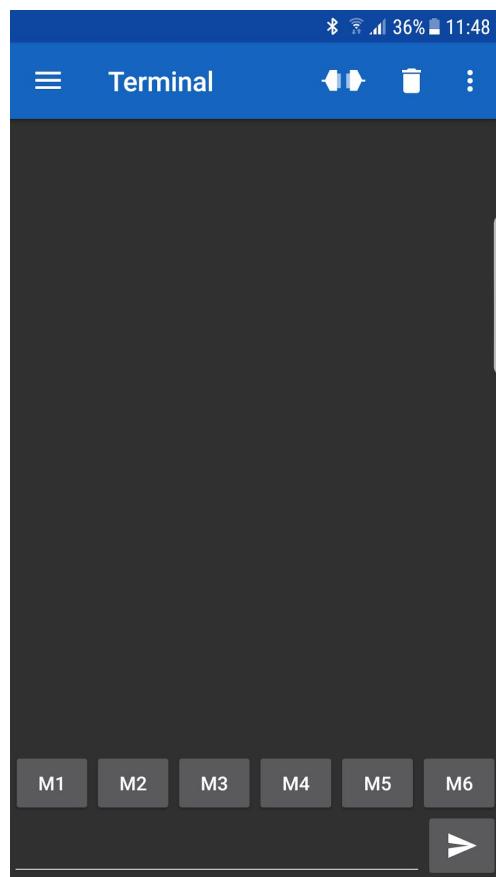
Esta aplicación permite conectarse tanto con módulos Bluetooth clásicos (como el HC05 y HC06), así como módulos Bluetooth Low Energy (BLE), como por ejemplo el módulo HM10.

Una vez seleccionado y conectado un dispositivo nos brinda una interfaz de Terminal serie para enviar y recibir datos entre la PC y el smartphone mediante utilizando bluetooth.

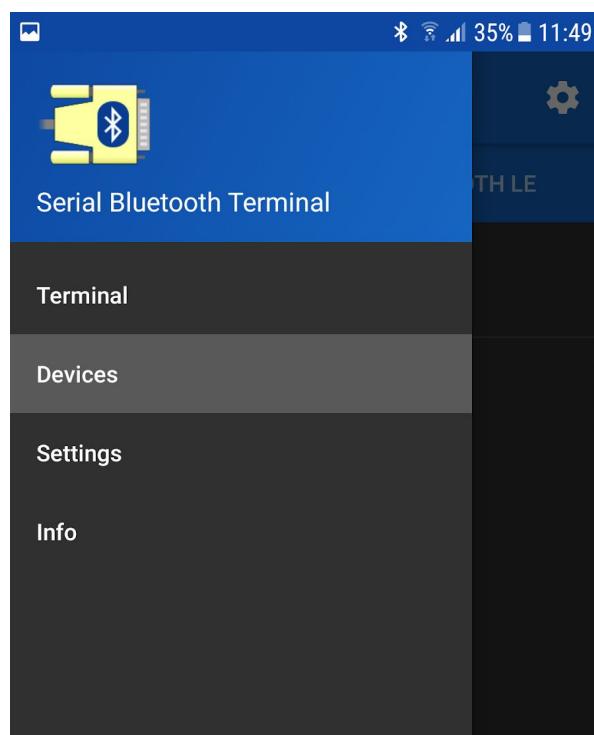
Se instala desde el Google Play Store:



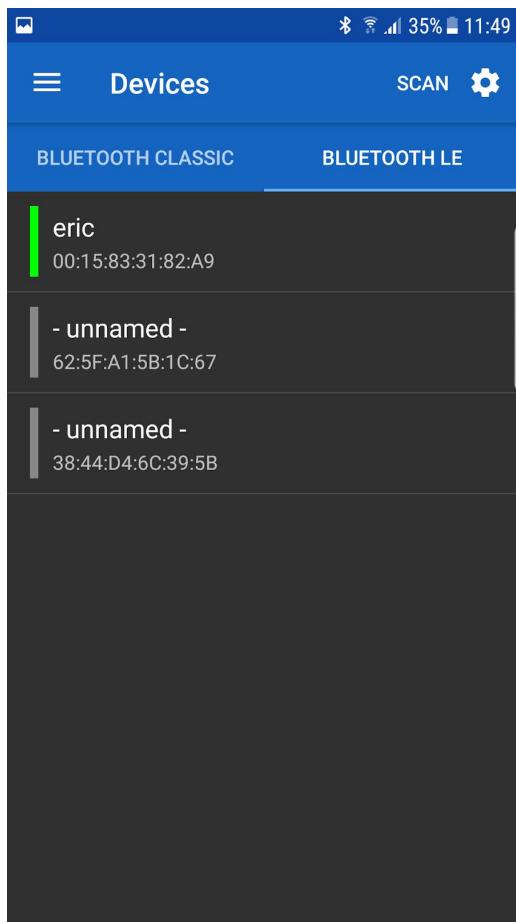
Para utilizarla abrir la misma y presionar el ícono de menú superior izquierdo:



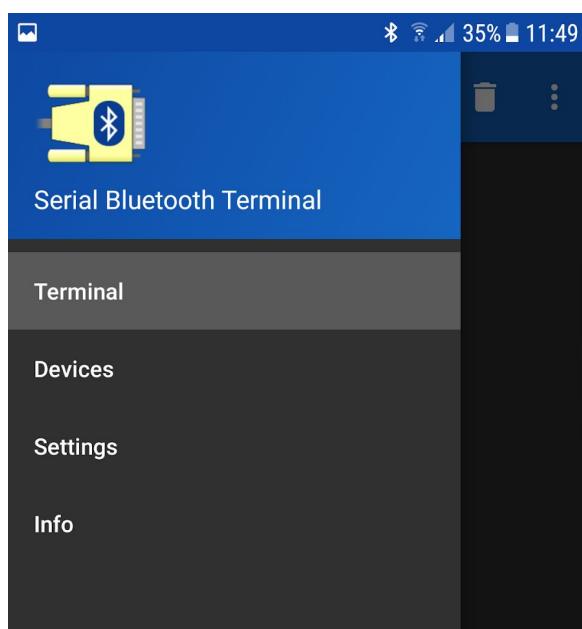
En el menú la elegir opción “Device” para buscar dispositivos:



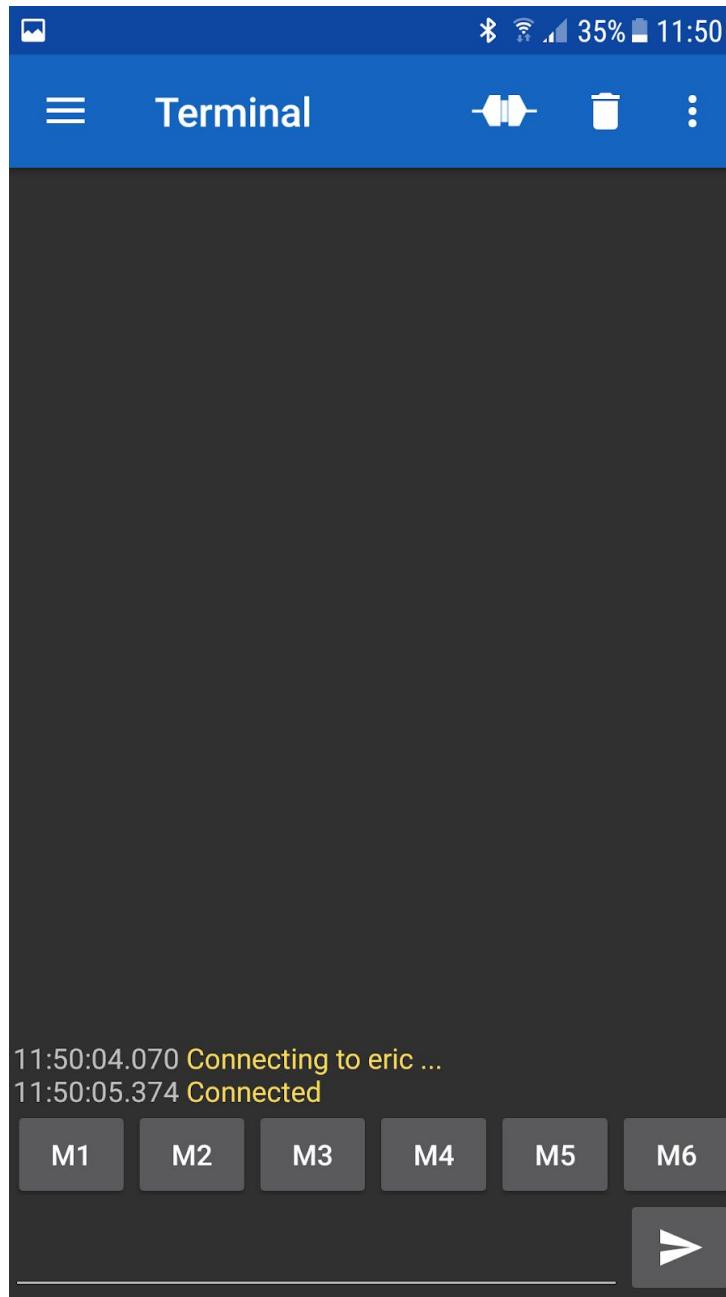
Elegir “BLUETOOTH LE” y presionar sobre la palabra “SCAN” arriba a la derecha. En este caso se puede observar los dispositivos detectados (con el dispositivo a conectar seleccionado):



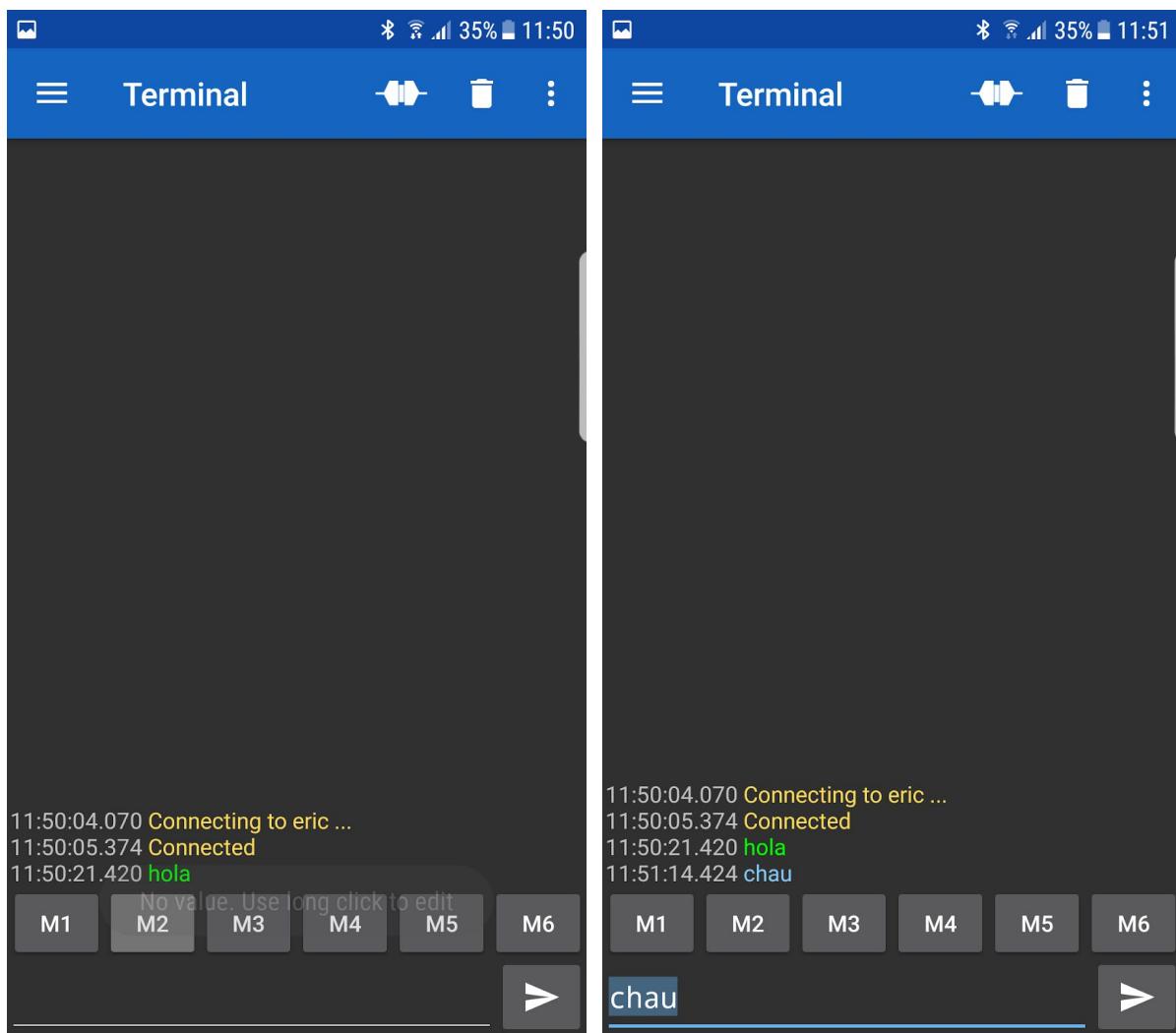
Volver a la terminal utilizando el menú de la aplicación:



Presionar el icono con los 2 cables para conectarse:



De esta manera el dispositivo queda conectado y se pueden enviar caracteres utilizando el campo de texto de la terminal y recibir caracteres enviados desde la PC. Por ejemplo, se envía “hola” desde la terminal serie del Smartphone hacia al terminal en la PC y se envía “chau” desde la terminal serie en la PC a la del Smartphone como se muestra en las siguientes imágenes (observar que lo enviado desde el Smartphone se muestra en verde, mientras que lo recibido desde la PC se muestra en azul en la terminal serie del Smartphone):



Observar que si se envía desde el Smartphone hacia la PC el carácter 'h' (High) se enciende el LEDB y con 'l' (Low) se apaga.

Aplicaciones Android con MIT App Inventor 2

Para realizar unas pequeñas aplicación de Android que interactúen con la EDU-CIAA-NXP utilizando el módulo BLE 4.0 HM10 se utiliza MIT App Inventor 2, el cual facilita la tarea de crear aplicaciones en Android permitiendo realizarlas de forma gráfica mediante un lenguaje de bloques encastrables.

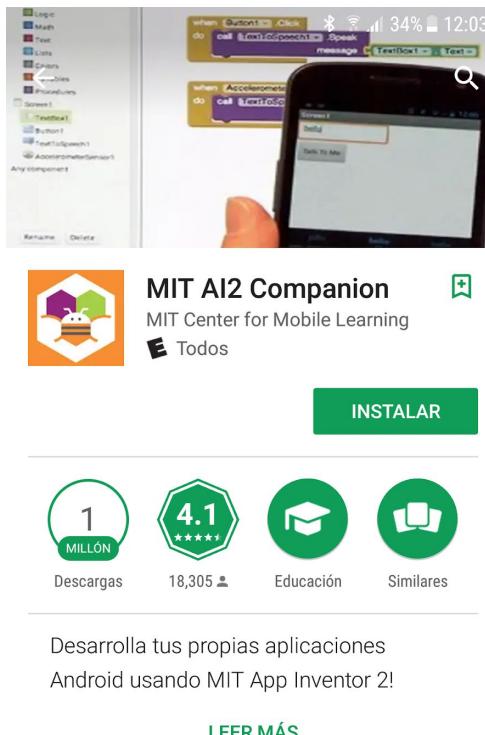
Se desarrollan 2 programas, el primero nombrado “HM10_LED” que permite solamente enviar datos desde el Smartphone hacia la PC, mientras que el segundo, “HM10_LED_BOTON”, permite comunicación bidireccional.

Programa HM10_LED

Este programa controla el LEDB de la EDU-CIAA-NXP desde una Aplicación en Android conectada por bluetooth utilizando el módulo HM10. En las siguientes secciones se describen los pasos para realizarla.

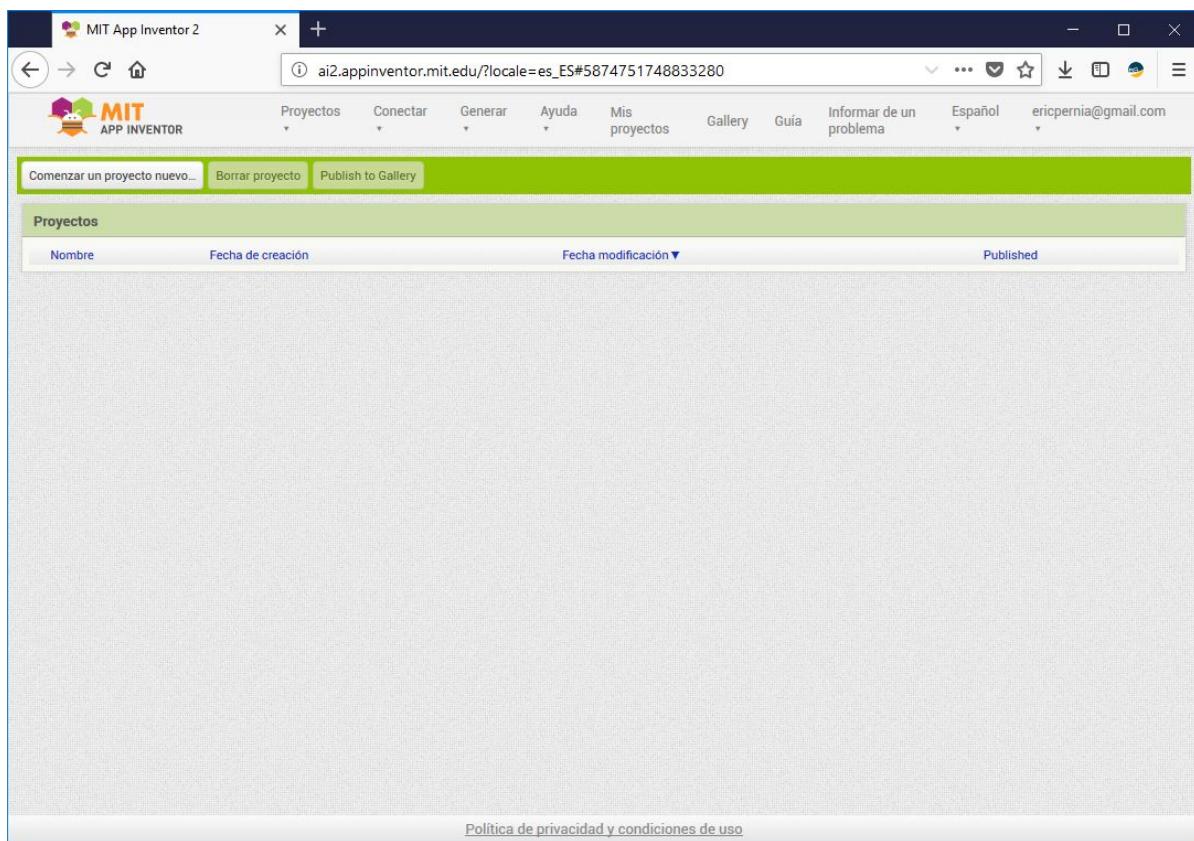
Descargar la aplicación “MIT AI2 Companion”

Además para poder comprobar el funcionamiento de la aplicación directamente desde el Smartphone de forma dinámica mientras la misma se va construyendo se utiliza la aplicación Android “MIT AI2 Companion”. Instalar la misma desde Google Play Store:



Ingresar en AppInventor

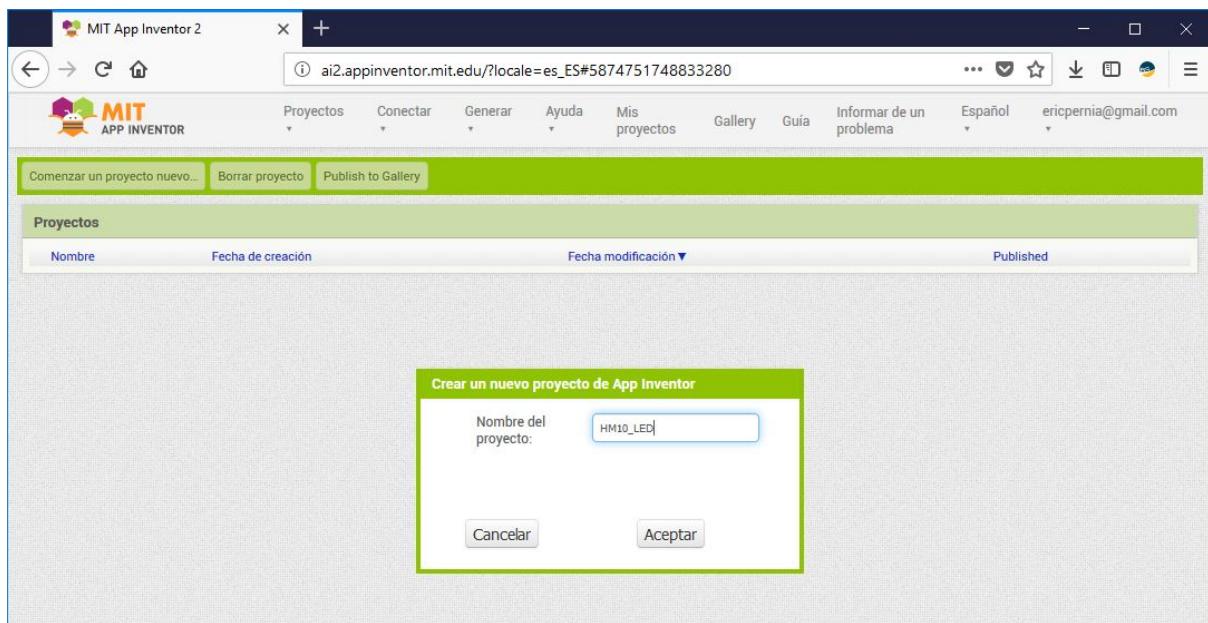
App inventor es una aplicación on line que se utiliza desde su browser, de esta manera se tiene que ingresar a su sitio web para utilizarla: ai2.appinventor.mit.edu/



Arriba a la derecha permite cambiar el lenguaje de la misma a español, idioma en el que se utilizará durante las siguientes secciones.

Crear una nueva aplicación

Se debe ir a la opción “Proyectos” del menú superior y luego eligiendo“Comenzar un nuevo proyecto...”. Poner como nombre de proyecto “” y luego “Aceptar” para crear el mismo.



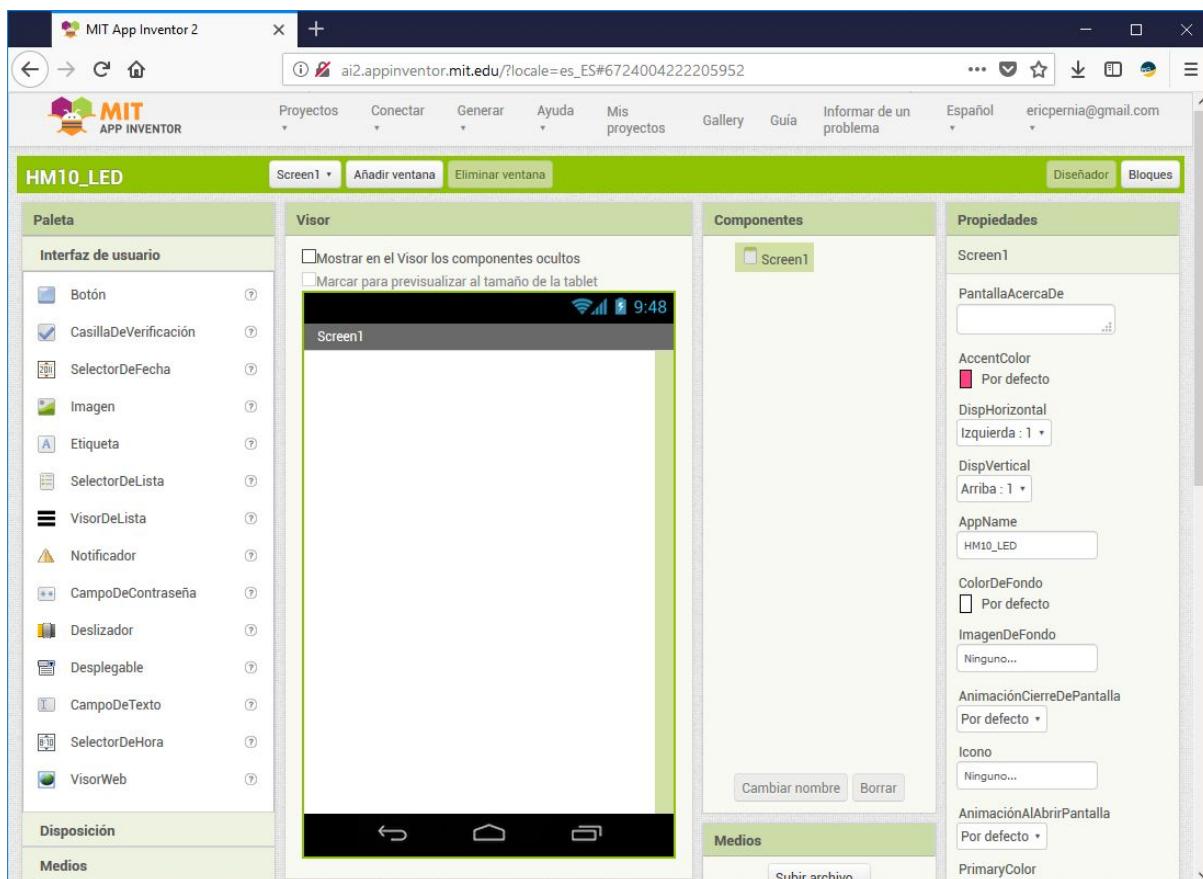
Construir la interfaz gráfica

El software App Inventor posee dos vistas bien distinguidas del proyecto, la vista “**Diseñador**” donde se construye la interfaz gráfica, y la vista “**Bloques**” donde se programa mediante bloques gráficos encastrables el funcionamiento de la aplicación. Estas vistas se cambian mediante los botones con estos nombres arriba a al derecha en la barra verde fluor.

Al crear una nueva aplicación la misma comienza con la vista “Diseñador” seleccionada por defecto. De esta forma primero se construye la interfaz gráfica de la aplicación, que consiste en una serie de componentes, los cuales son por ejemplo, elementos de layout en pantalla, botones, etiquetas, etc.

La vista “Diseñador” se divide en 4 paneles verticales, los cuales de izquierda a derecha son:

- **Paleta:** contiene la paleta de “componentes” a utilizar para construir la interfaz gráfica de la aplicación. Se deben elegir un componente y arrastrar a la pantalla del smartphone dibujada en el panel “Visor” para crear una nueva instancia del mismo.
- **Visor:** es la representación de como se verá nuestra aplicación en el Smartphone, permite ubicar los componentes de forma interactiva.
- **Componentes:** contiene la lista de instancias de componentes en uso en la interfaz gráfica.
- **Propiedades:** muestra las propiedades del componente instancia seleccionado actualmente.



Para realizar una aplicación BLE se debe descargar una “extensión” que consiste en un componente de App Inventor para el manejo de módulos BLE, la misma se descarga de:

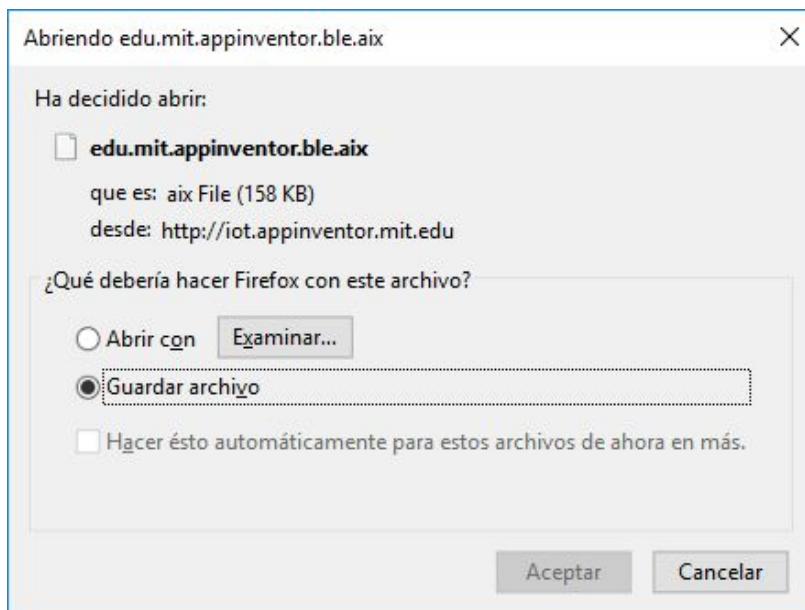
<https://puravidaapps.com/extensions.php>

The screenshot shows a web browser window with two tabs open: "MIT App Inventor 2" and "App Inventor Extensions | Pura Vida Apps". The main content area displays a list of extensions categorized by source:

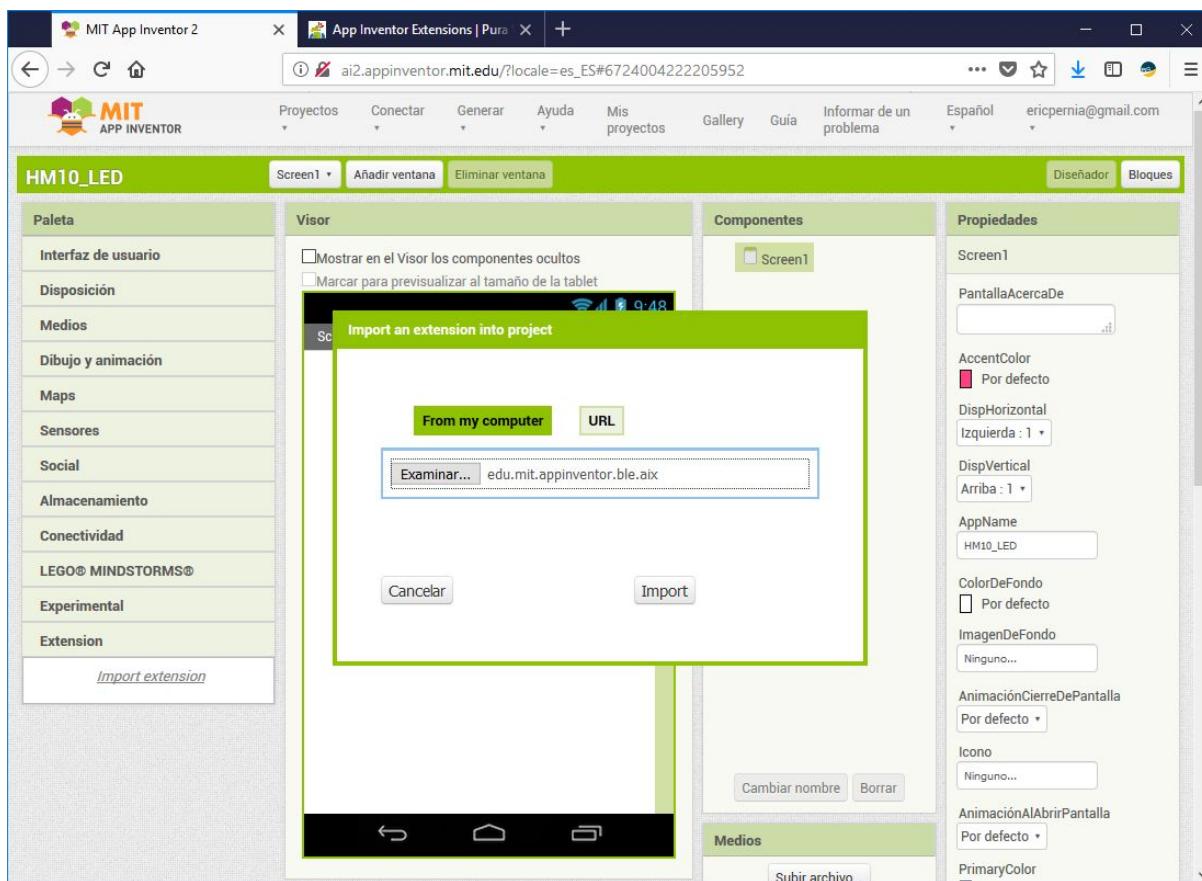
- 1. MIT**
 - Vector Arithmetic Extension by Ethan: Takes in two vectors and can add them to return a result vector.
 - Image Processor Extension by Justus: can do a weighted combine of two images, return the greyscale of an image.
 - Sound Analysis Extension by Mouhamadou: analyzes the pitch of a sound through the microphone and returns it.
 - Scale Detector Extension by Hal: Adds a multitouch scale gesture detector to a Canvas.
 - Bluetooth Low Energy Extension (old) by MIT
 - Bluetooth Low Energy Extension (new) by MIT
 - Microbit Extension by MIT
 - Rotation Detector Extension by Xinyue Deng, which reacts to two-finger rotation gestures.
 - ChartMaker Extension by Kate Manning and Emily Kager
 - Android Things Extension by Thilanka Munasinghe
- 2. Mad Robots**
 - Gyro Sensor Extension by Gareth
 - Sound Extension by Gareth: offers Pitch effects, Volume Control, Speaker Balance
- 3. Makeblock**
 - Computer Vision Extension by Makeblock: for color detection, face detection and feature analysis (using online API).
 - mBot Extension by Makeblock: to control Makeblock mBots.

NOTA: observar que sea la “new” en lugar de la “old”.

El archivo que descarga se llama “edu.mit.appinventor.ble.aix”:

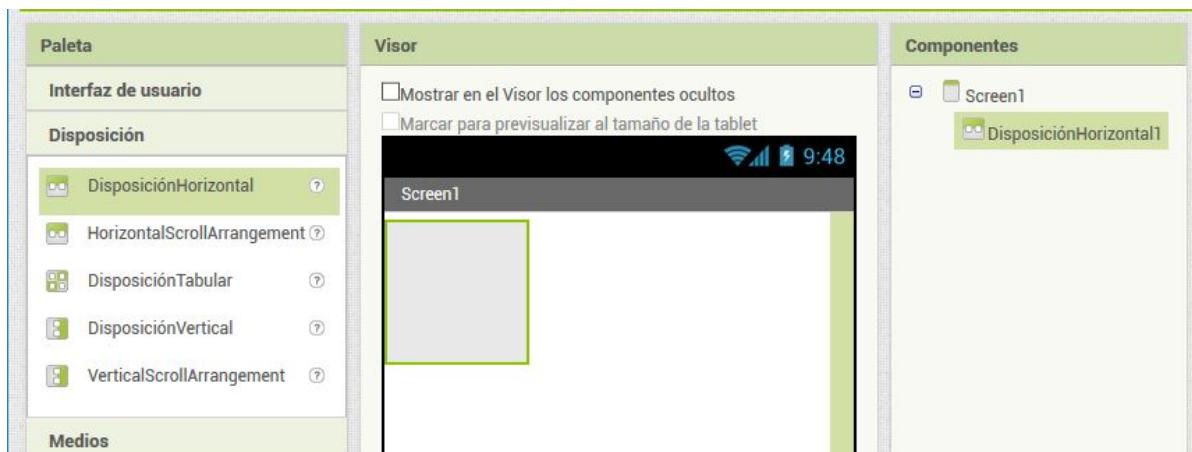


Este archivo se importa mediante la opción “import extension” en el menú “Extensión” del panel **Paleta**. Buscar la extensión en la carpeta de descargas, elegir y presionar el botón “Import”.

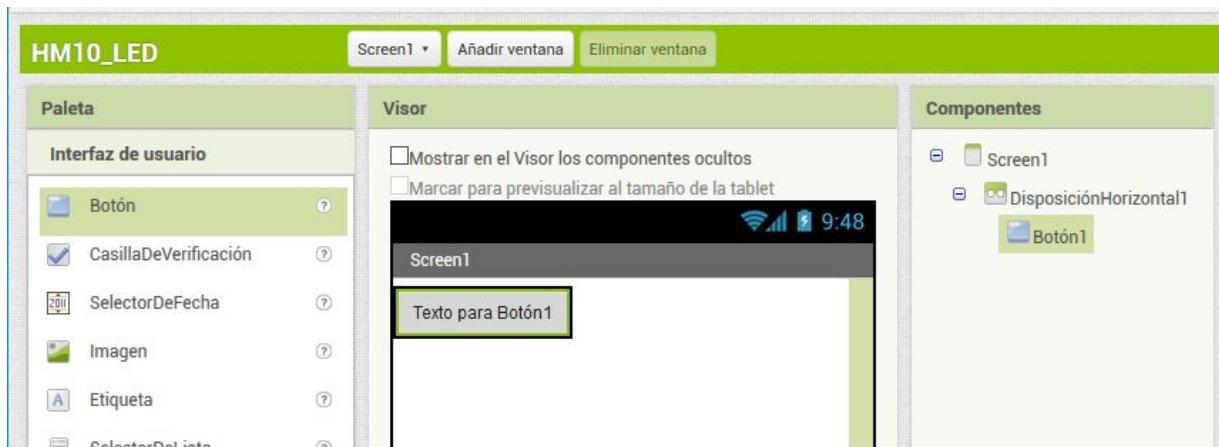


A continuación se eligen los componentes de interfaz gráfica de la aplicación.

Se comienza por agregar un componente **DisposiciónHorizontal** que es un componente contenedor para realizar un layout horizontal de otros componentes en su interior. Seleccionar el mismo del panel “Paleta”, dentro del menú “Disposición” y arrastrarlo al “Visor”:

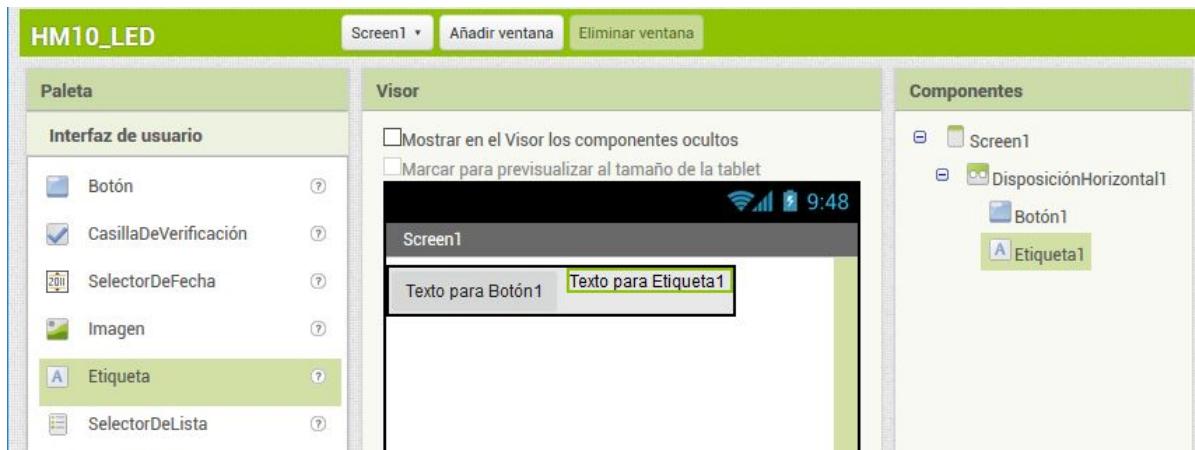


Luego se añade un componente **Boton** dentro del menú “Interfaz de usuario” y colocarlo dentro del componente “DisposiciónHorizontal1” creado previamente:

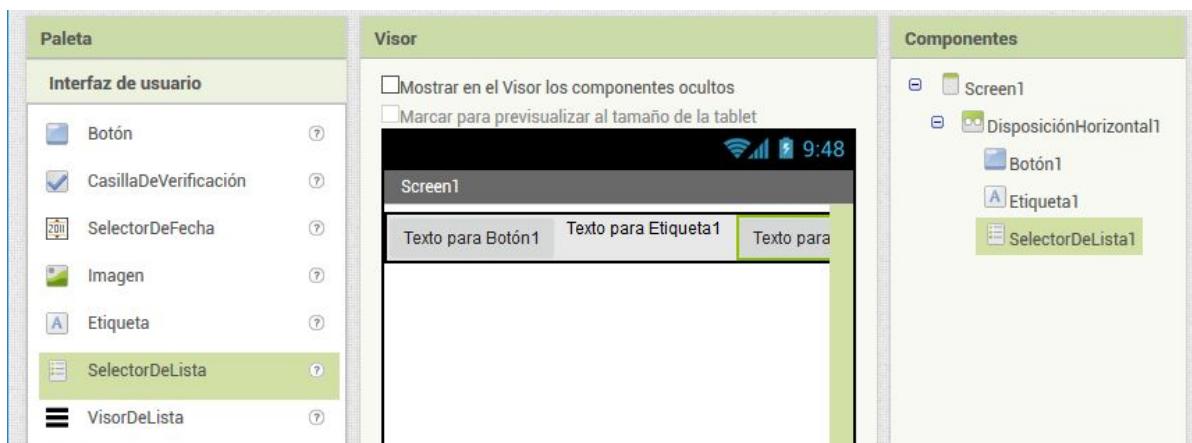


Observar el árbol de componentes que se va creando en el panel “Componentes”.

Añadir una **Etiqueta** a la derecha del Botón:

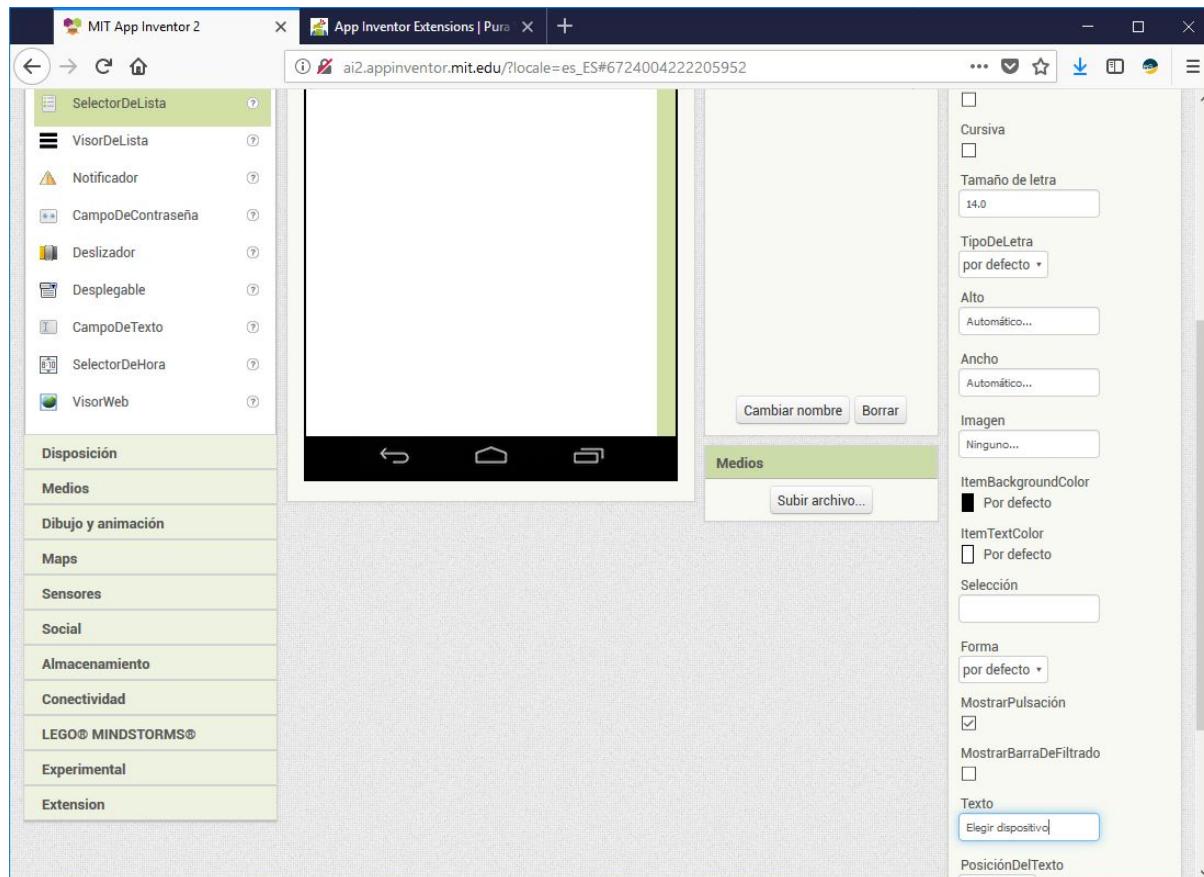


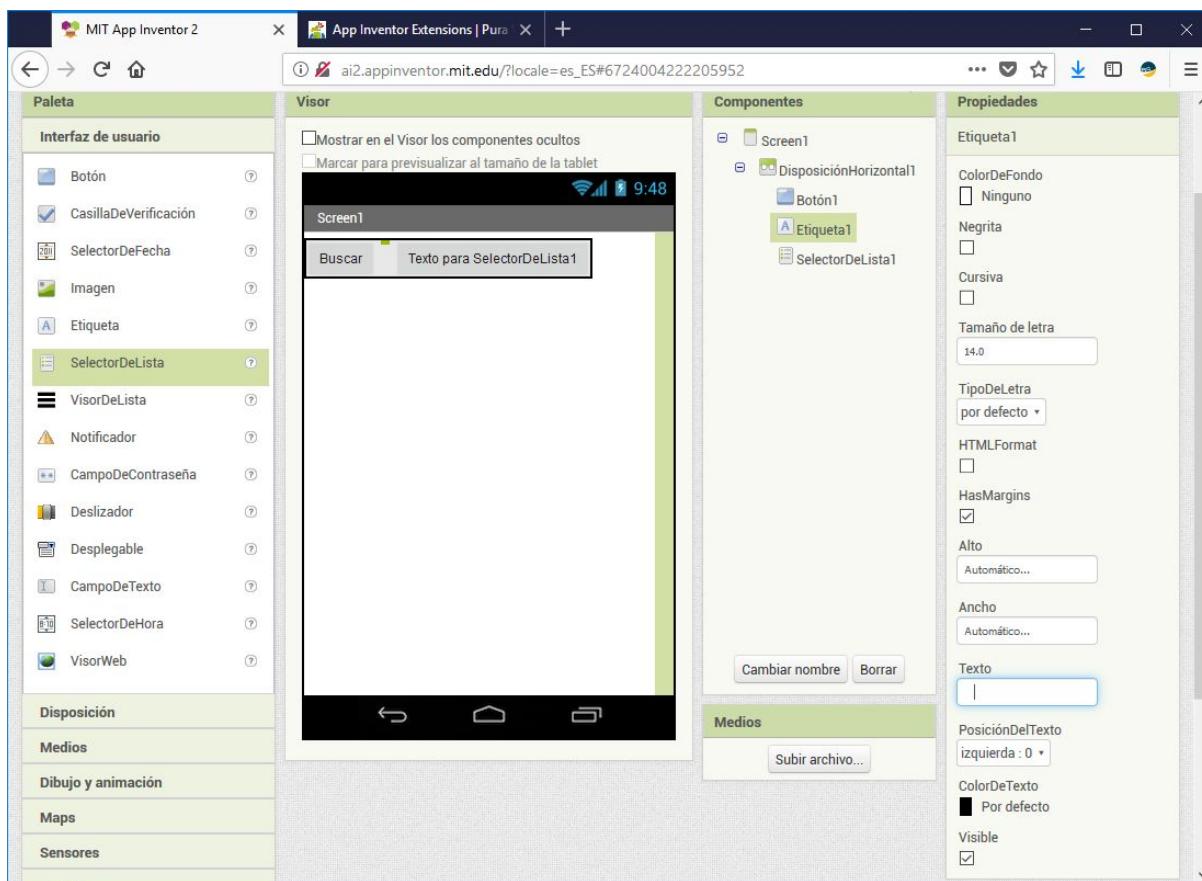
Añadir un **SelectorDeLista** a la derecha de la Etiqueta:



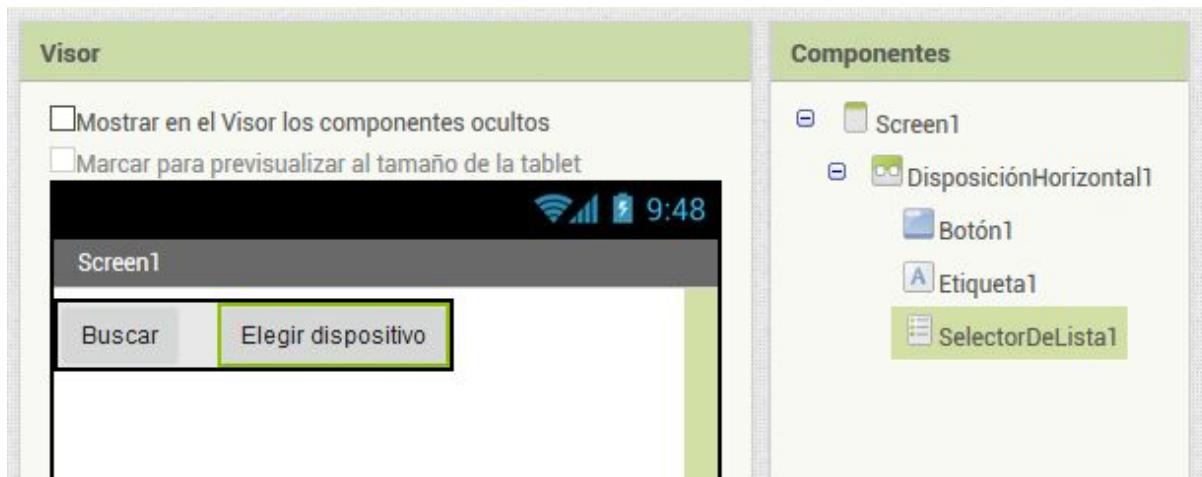
Renombrar cada uno de los componentes creados seleccionandolos de a uno por vez en el panel “Componentes” y cambiando la opción “Texto” en el panel “Propiedades” de la siguiente manera.

- **Boton1** cambiarle el texto a “Buscar”.
- **Etiqueta1** cambiar el texto a “ ” (2 espacios, lo utilizaremos como separador).
- **SelectorDeLista1** cambiar el texto a “Elegir dispositivo”.





De esta manera, la interfaz gráfica debería lucir así:



Agregar los componentes y ajustar sus nombres como se muestra en la siguiente pantalla.

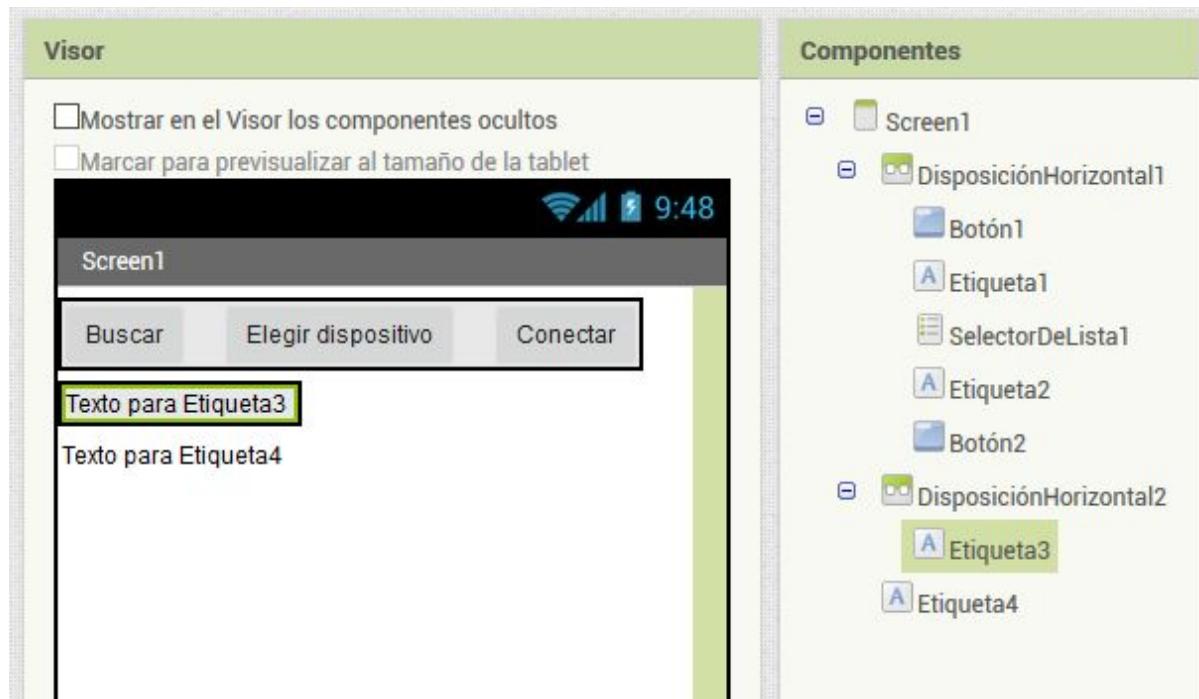
Dentro de **DispersionHorizontal1**:

- **Etiqueta2** cambiar el texto a “ ” (2 espacios, lo utilizaremos también como separador).
- **Boton2** cambiarle el texto a “Conectar”.

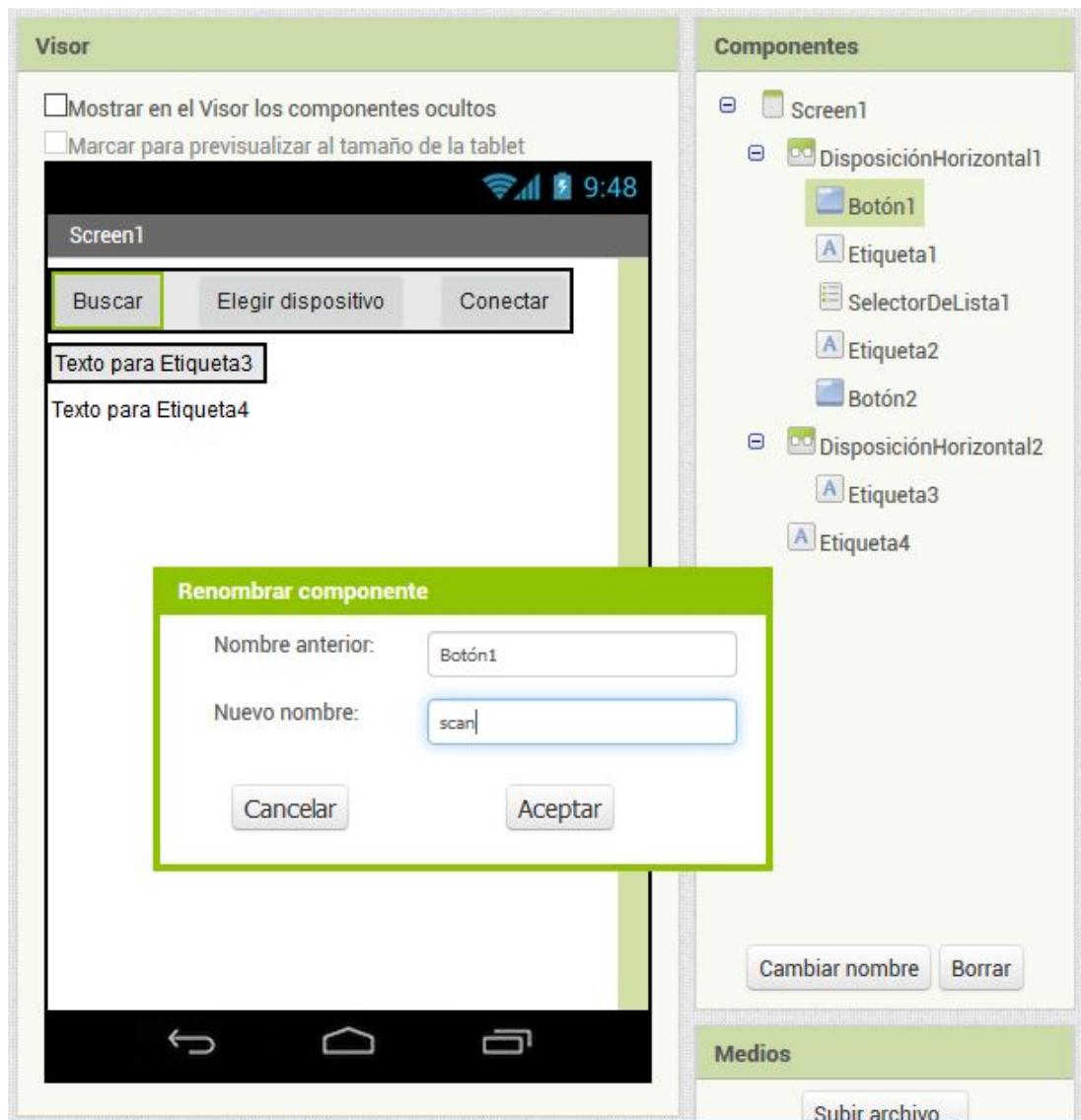
Agregar debajo de **DispencionHorizontal1** un **DispencionHorizontal2** y en su interior una **Etiqeta3**.

Agregar una **Etiqeta4** debajo de **DispencionHorizontal2**.

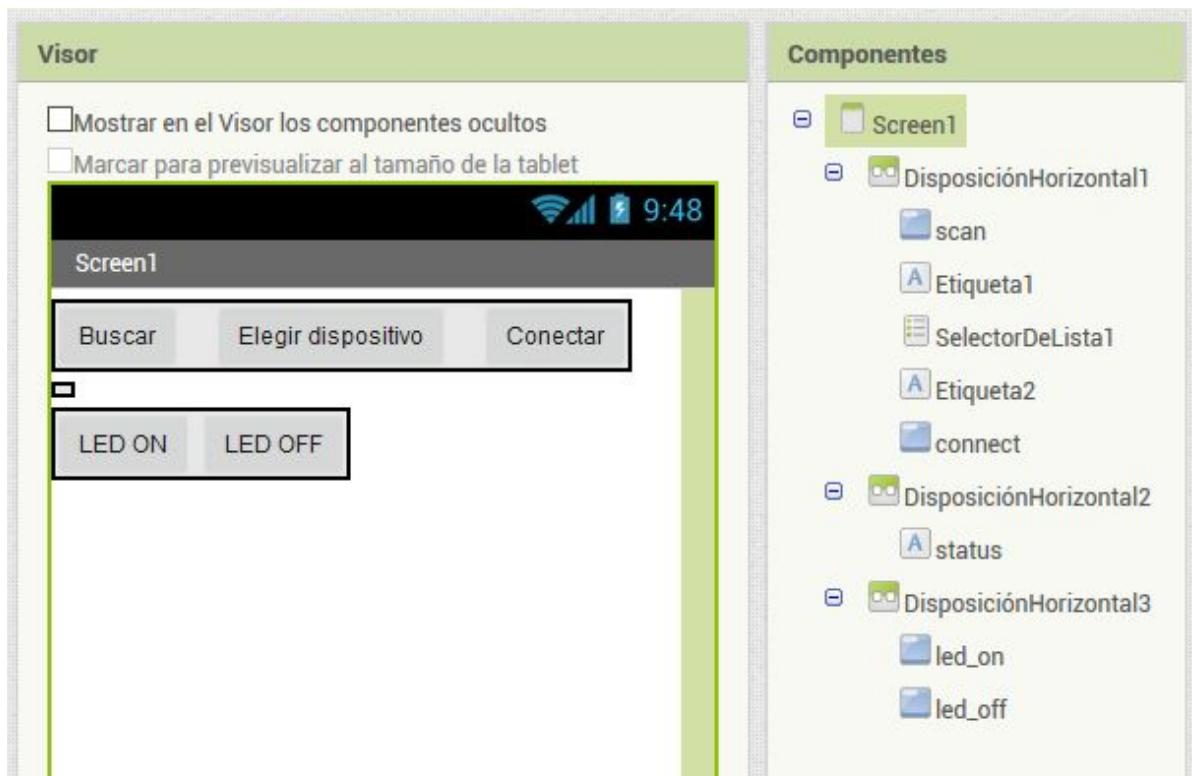
De esta manera, la interfaz gráfica debería lucir así:



Seleccionar **Boton1** en el panel “Componentes” y renombrarlo a “scan” con el botón “Cambiar nombre de este panel”:

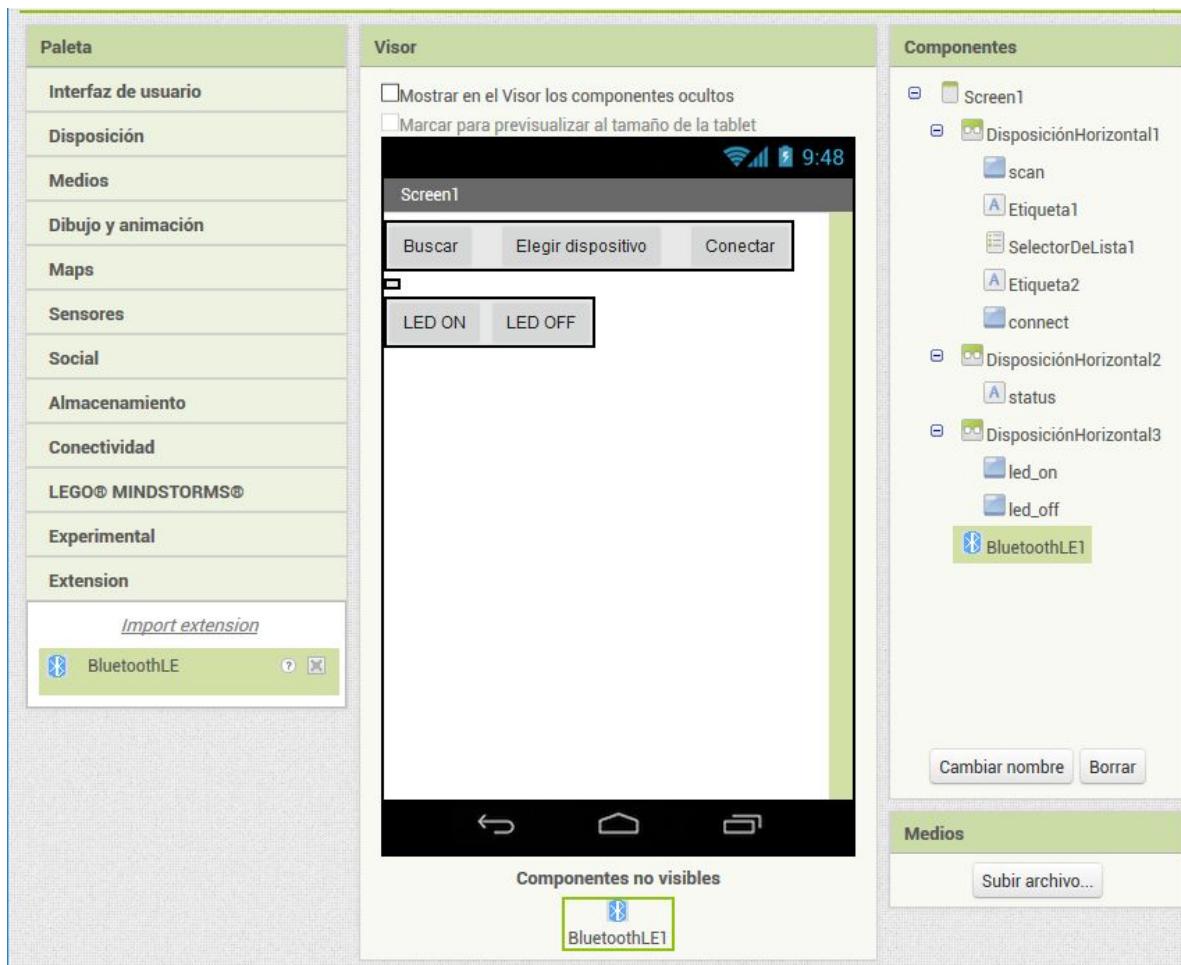


Crear el resto de los componentes y renombrarlos para lograr la siguiente interfaz gráfica:

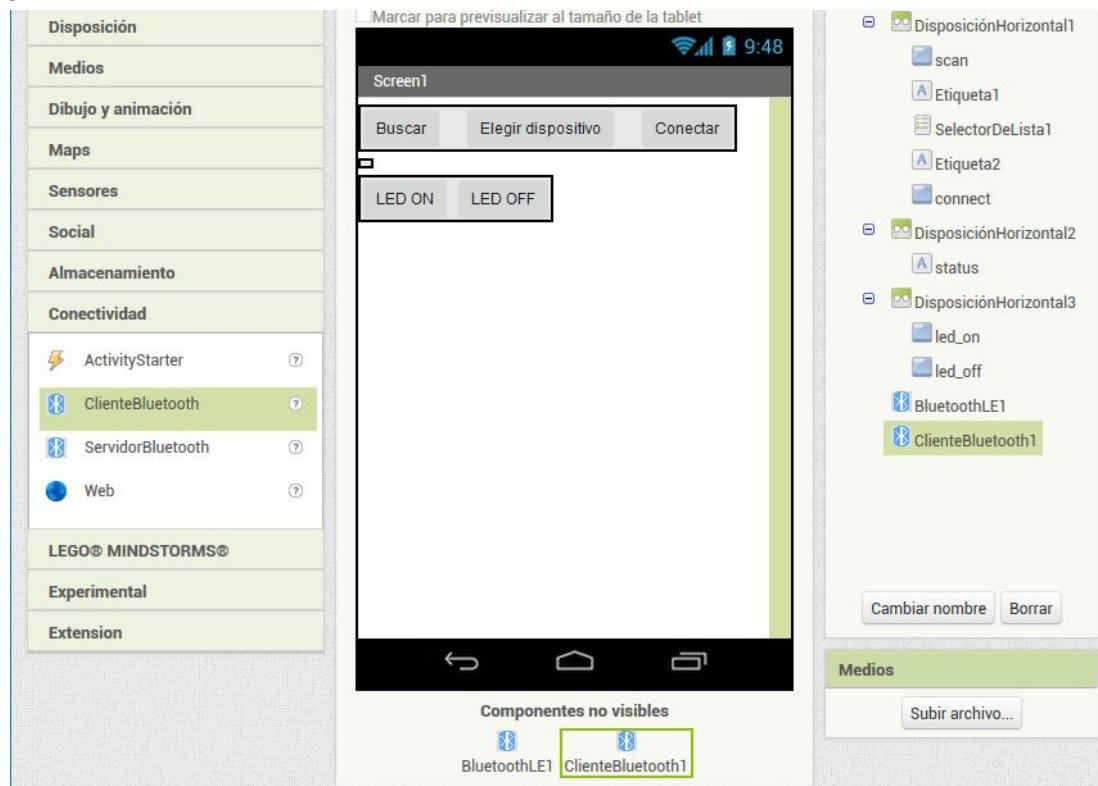


La etiqueta **status** tiene el texto vacío (se ve como un pequeño rectángulo negro). No se utiliza una etiqueta como separador entre los botones **led_on** y **led_off**.

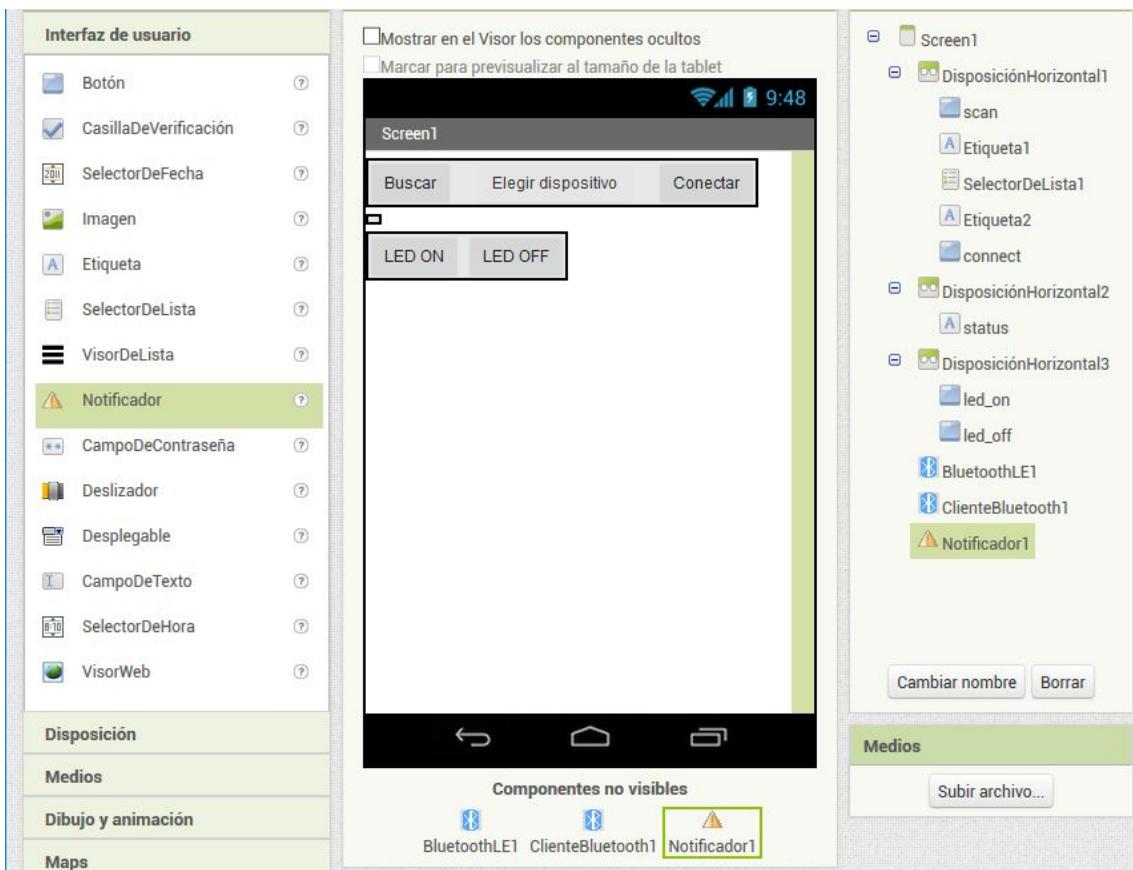
Agregar el componente **BluetoothLE** el cual habíamos importado previamente desde el menú “Extensión” (arrastrar a cualquier parte de la pantalla dibujada y se colocará debajo de esta automáticamente):



Luego el componente **ClienteBluetooth** del menú “Conectividad”:

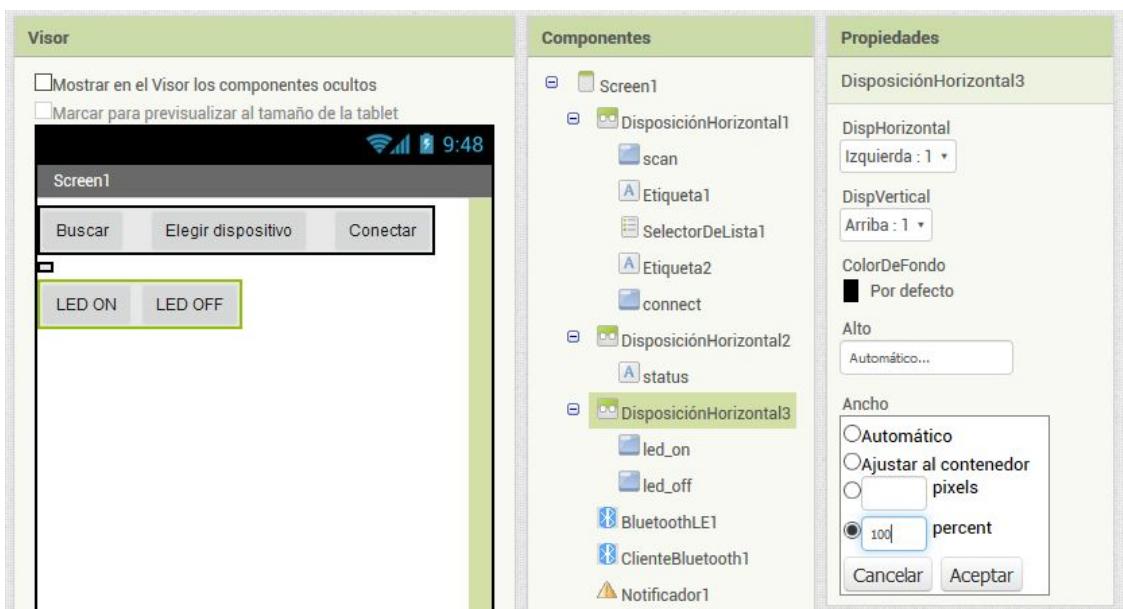


Agregar un componente Notificador desde el menú “Interfaz de usuario”:

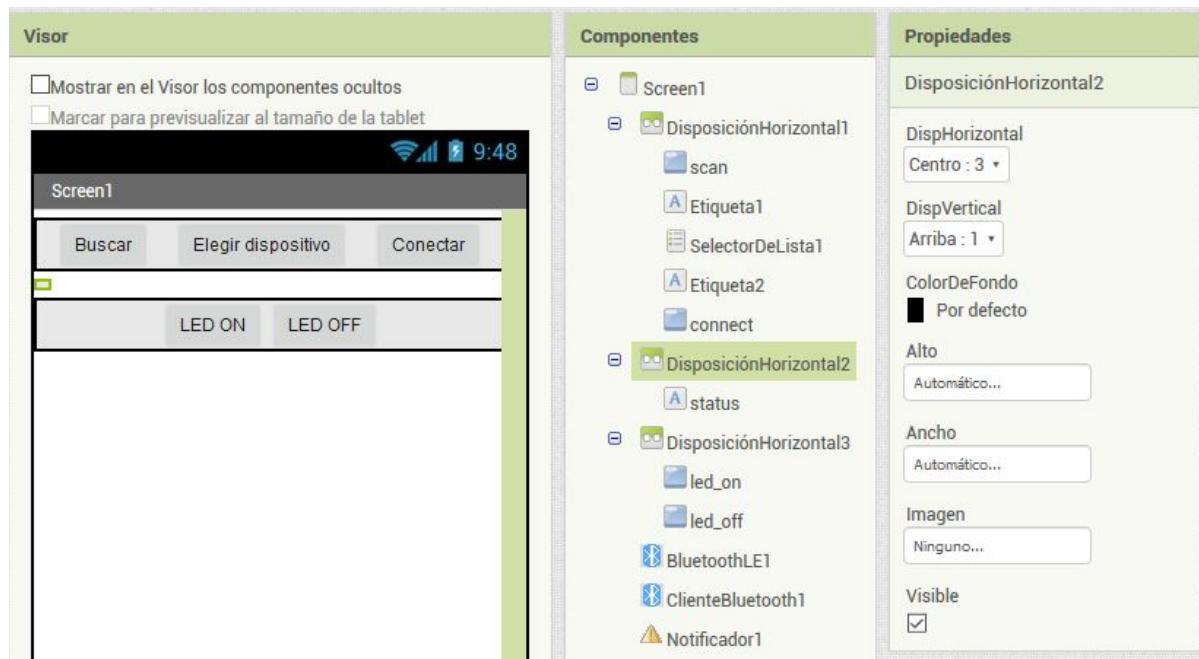


Este componente Notificador muestra “mensajes de alerta” que el usuario puede cerrar presionando un botón.

Para acomodar las cosas en pantalla vamos a ajustar las Propiedades de los componentes “DisposiciónHorizontal” para que ocupen el 100% del ancho en pantalla:

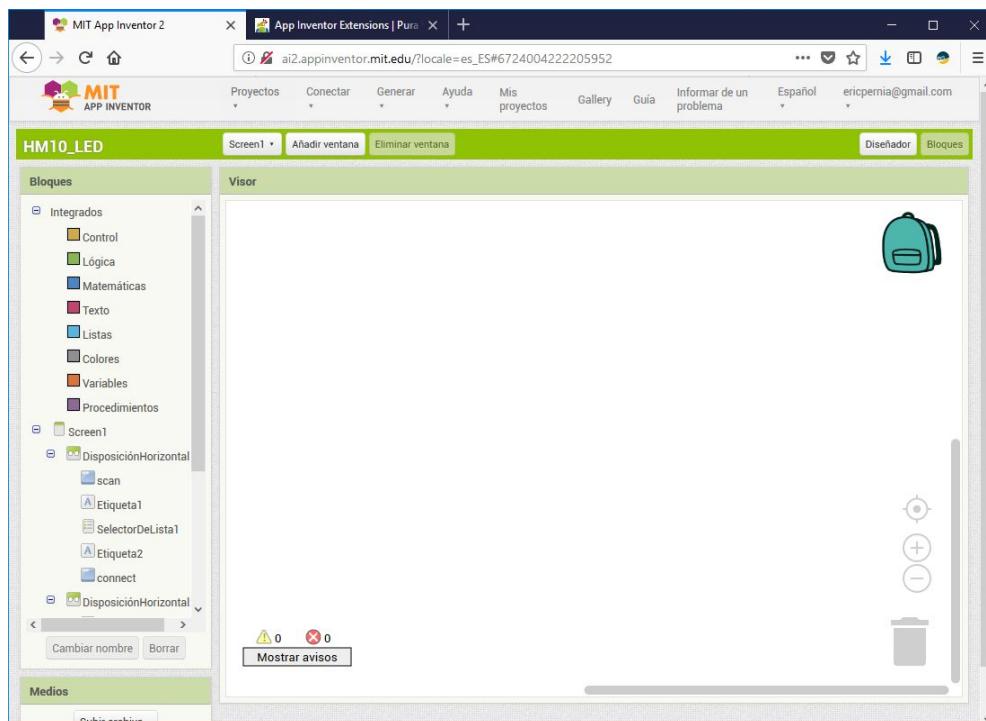


También la propiedad “DispHorizontal” en “Centro” quedando todo ordenado en pantalla:



Programar el comportamiento de los elementos de la interfaz gráfica mediante bloques encastrables

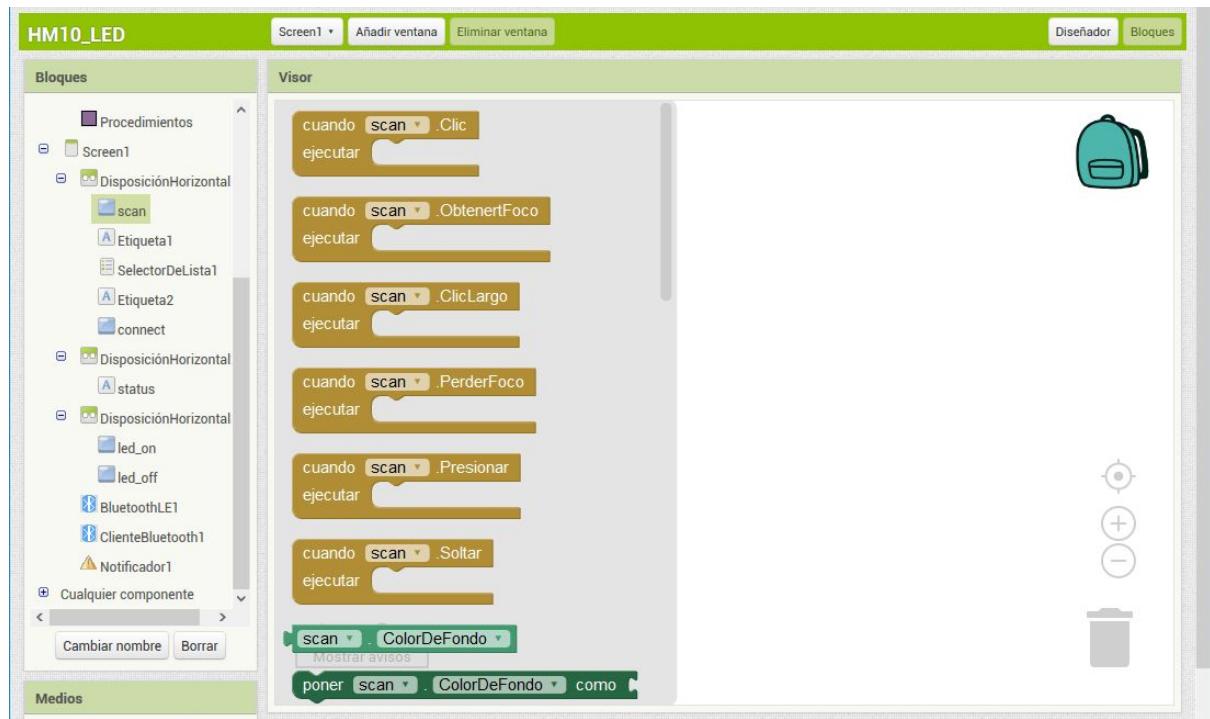
Se selecciona la vista "Bloques" donde se programa mediante bloques gráficos encastrables el funcionamiento de la aplicación.:.



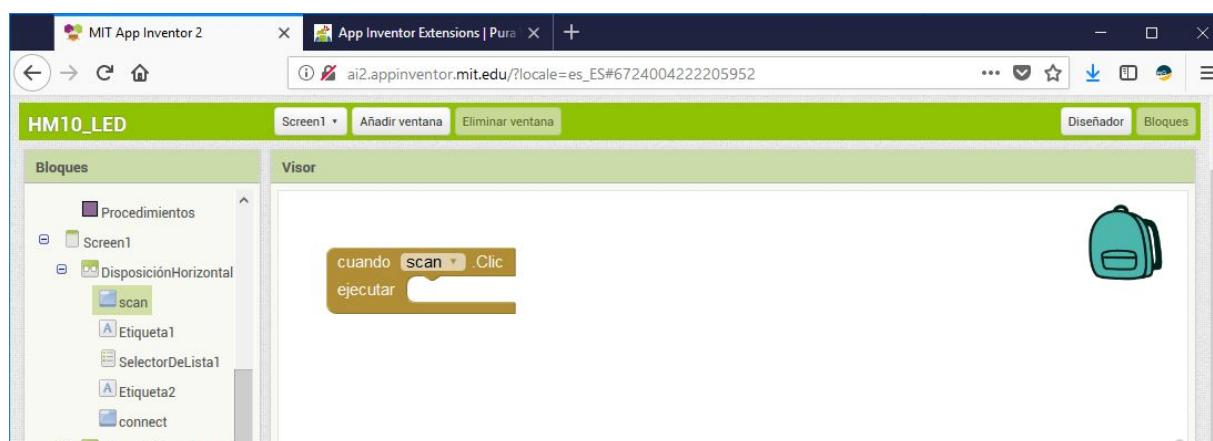
Esta vista contiene únicamente dos paneles:

- **Bloques:** que contiene los bloques con los que se construyen los programas que manejan cosas como matemática, lógica y texto con cada componente que se ha agregado.
- **Visor:** donde se crea el programa arrastrando los bloques elegidos desde el panel Bloques y se van encastrando tipo rompecabezas para definir la lógica de programa.

Se comienza el programa definiendo el comportamiento del botón scan cuando se haga click sobre él (se apriete en la pantalla del smartphone) seleccionando el botón **scan** del panel Bloques y eligiendo el bloque **cuando scan Click:**



y arrastrándolo al panel “Visor” quedando:



Note que un bloque sacado desde un cierto tipo de componente puede cambiar a cual componente afecta mediante la flecha hacia abajo al lado del nombre, en este caso “**scan**”.

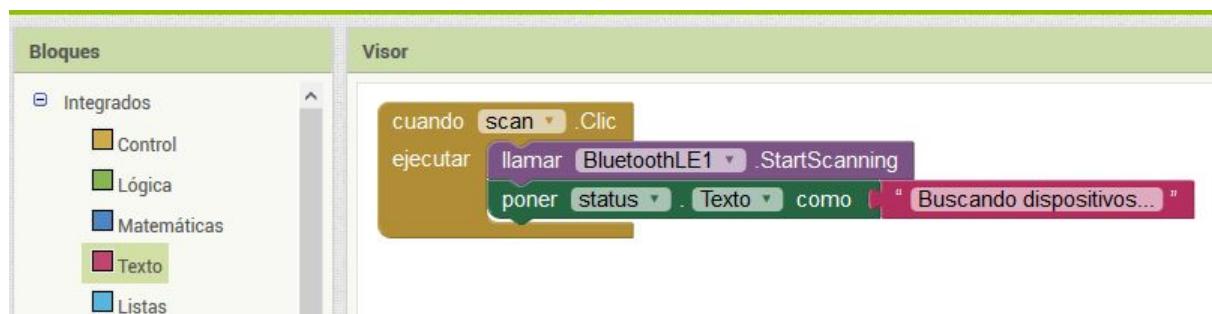
Armar el programa eligiendo los bloques correspondientes y encastrándolos entre sí:



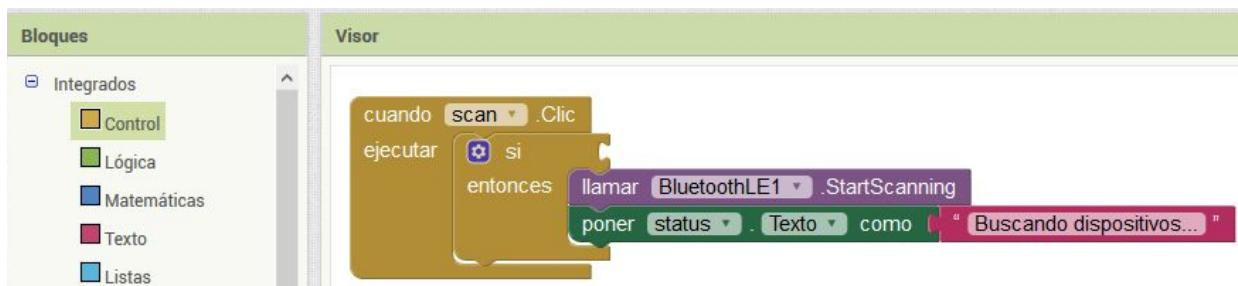
El bloque **llamar BluetoothLE1 .StartScanning** lo encontrará dentro del componente **BluetoothLE1**, mientras que el bloque **poner status . Texto como**, lo encontrará en el componente **status**.

Nótese además como se le cambia la propiedad que deseo establecer de **status** haciendo click sobre “Texto”.

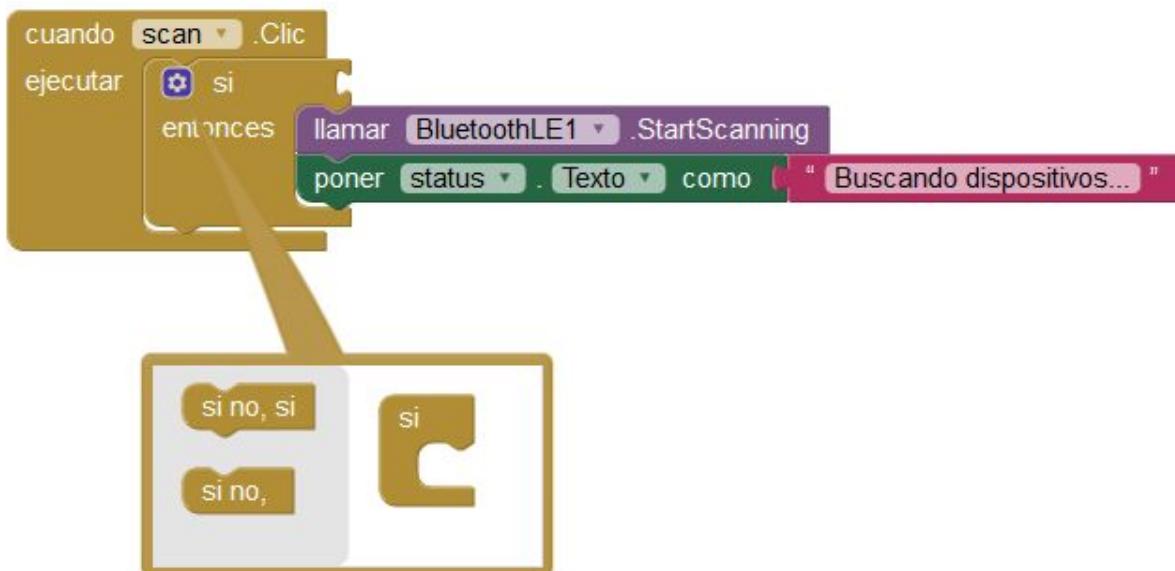
Luego se agrega el bloque **Texto** y se cambia el valor a “Buscando dispositivos...”:



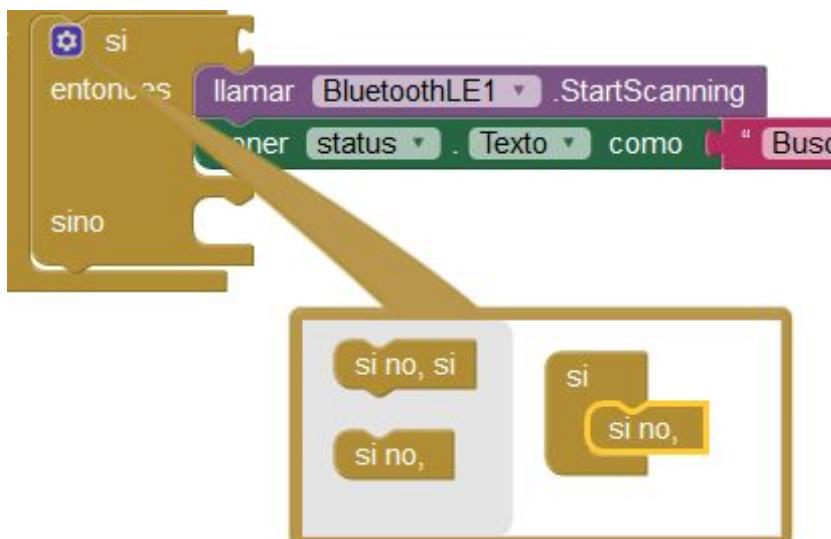
A continuación se agrega un bloque de ejecución condicional **si entonces** (equivalente a un if-then):



Para agregar la opción “else” del “if” hay que presionar sobre el engranaje de fondo azul y en la subvista que aparece



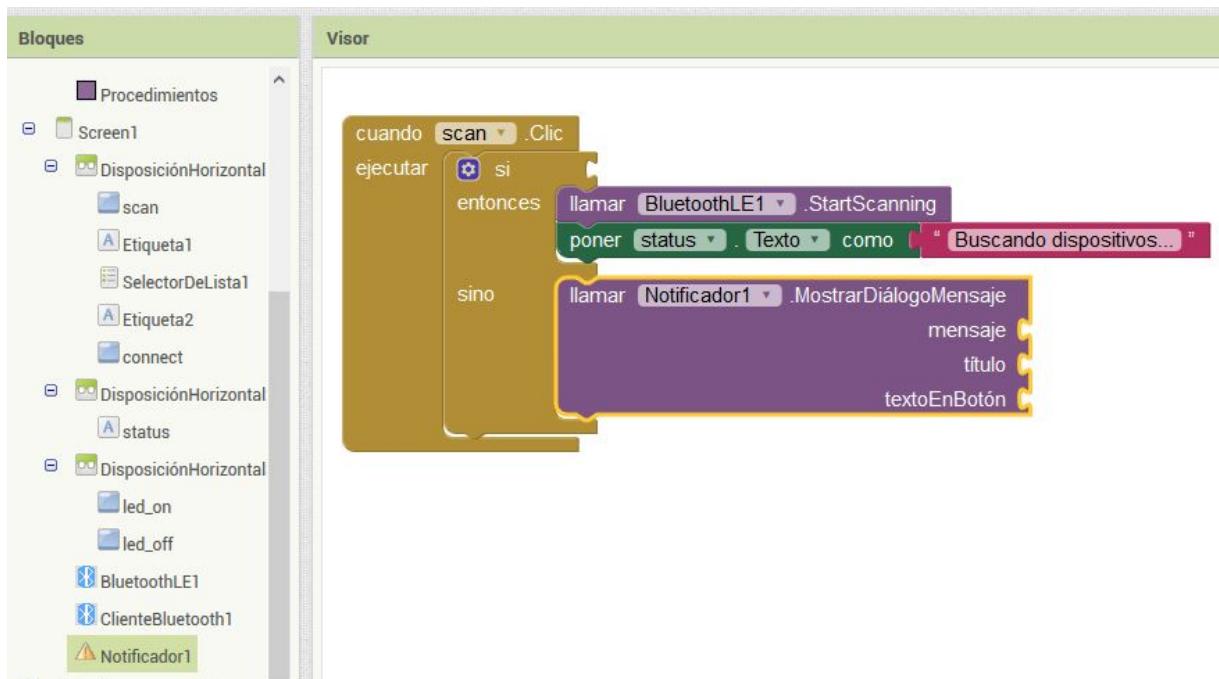
arrastrar el bloque “si no,” desde la mitad izquierda hacia la mitad derecha y encastrarlo dentro del bloque “si”:



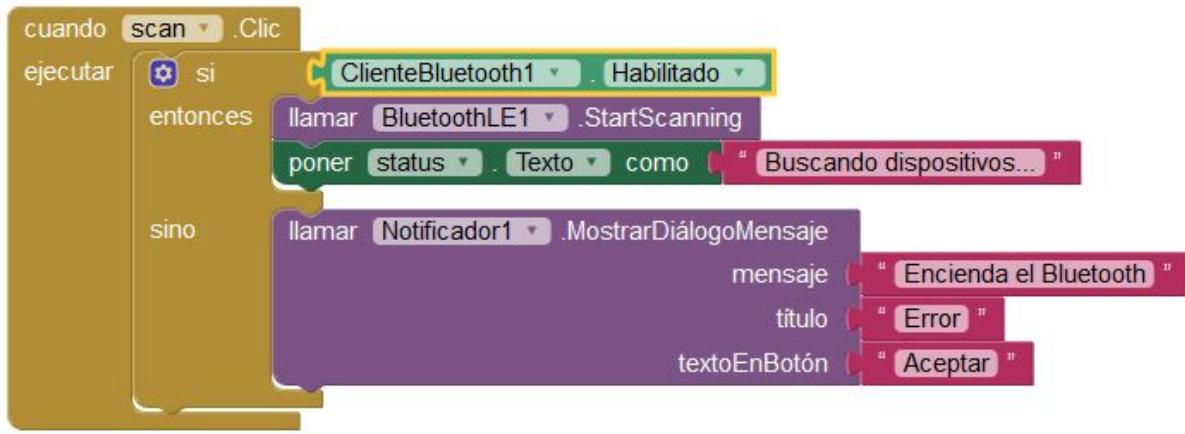
Con eso aparecerá la opción “else” en el bloque “if”.

Nota: El bloque “si no, si” sería el equivalente al “else if”.

Agregar el bloque **llamar Notificador1 .MostrarDialogoMensaje:**



Completar agregando bloques de Texto como se muestra a continuación:



De esta forma se finaliza la programación del evento click del botón **scan** (botón que dice Buscar en la interfaz gráfica) cuya tarea es verificar que esté el bluetooth habilitado antes de escanear en busca de dispositivos. Si el bluetooth está encendido, comienza la búsqueda de dispositivos y pone en status el mensaje “Buscando dispositivos...” y si en cambio el bluetooth está apagado, muestra un mensaje de notificación para que el usuario de la aplicación recuerde encender el bluetooth.

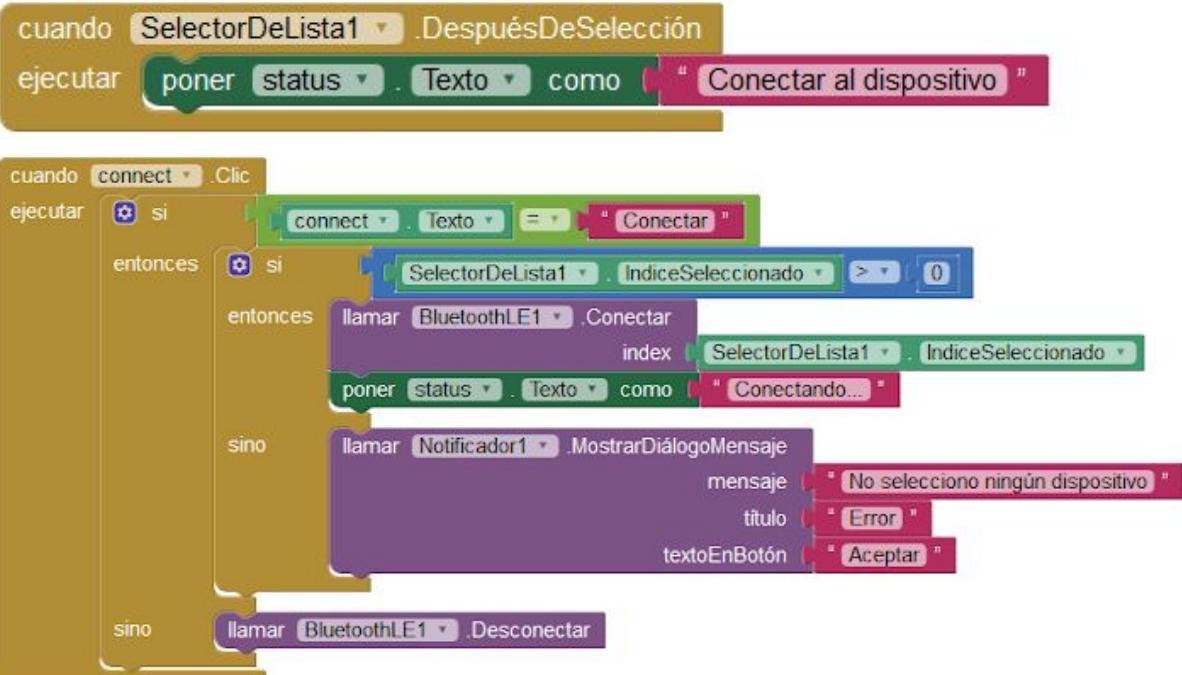
Se prosigue con el programa con del evento **DeviceFound** de **BLuetoothLE1**, es decir cuando se encuentra un dispositivo:

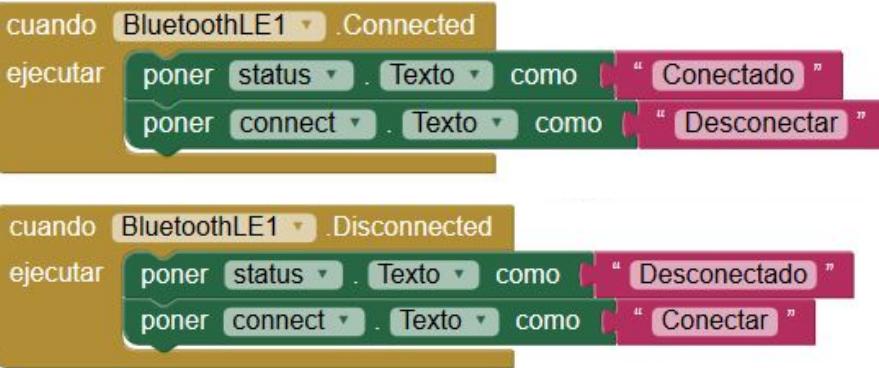


Cuando ocurre este evento llama a la función StopScanning del componente **BluettohLE1** y poner los elementos encontrados en el componente **SelectorDeLista1** y muestra el texto “Seleccione el dispositivo” en el componentes status.

A continuación se programan los siguientes eventos:

- **cuando SelectorDeLista1 .DespuesDeSelección:** que ocurre al elegir un dispositivo de la lista de dispositivos a conectar. En este caso se muestra el texto “Conectar al dispositivo” en status.
- **cuando connect .Click:** El evento de presionar el botón **connect** para conectar dispositivo. Se encarga tanto de la lógica de conexión como de desconexión.
- **cuando BluetoothLE1 .Connected:** Este evento ocurre cuando efectivamente se realiza la conexión. Simplemente pone el texto “Conectado” en **status** y cambia el texto del botón **connect** a “Desconectar” ya que va a hacer la acción que realiza una vez que la conexión fue establecida (se usa el mismo botón para conectar y desconectar).
- **cuando BluetoothLE1 .Disconnected:** Este evento ocurre cuando se realiza la desconexión. Simplemente pone el texto “Desconectado” en **status** y cambia el texto del botón **connect** a “Conectar”.





Para programar los eventos que controlan de click de los botones **led_on** y **led_off** que se encargan de enviar los mensajes para controlar el estado del LED utilizando el módulo BLE se requiere escribir los datos en la **característica personalizada** del **servicio personalizado**. Para esto se debe conocer el UUID (identificador único universal) de ambos.

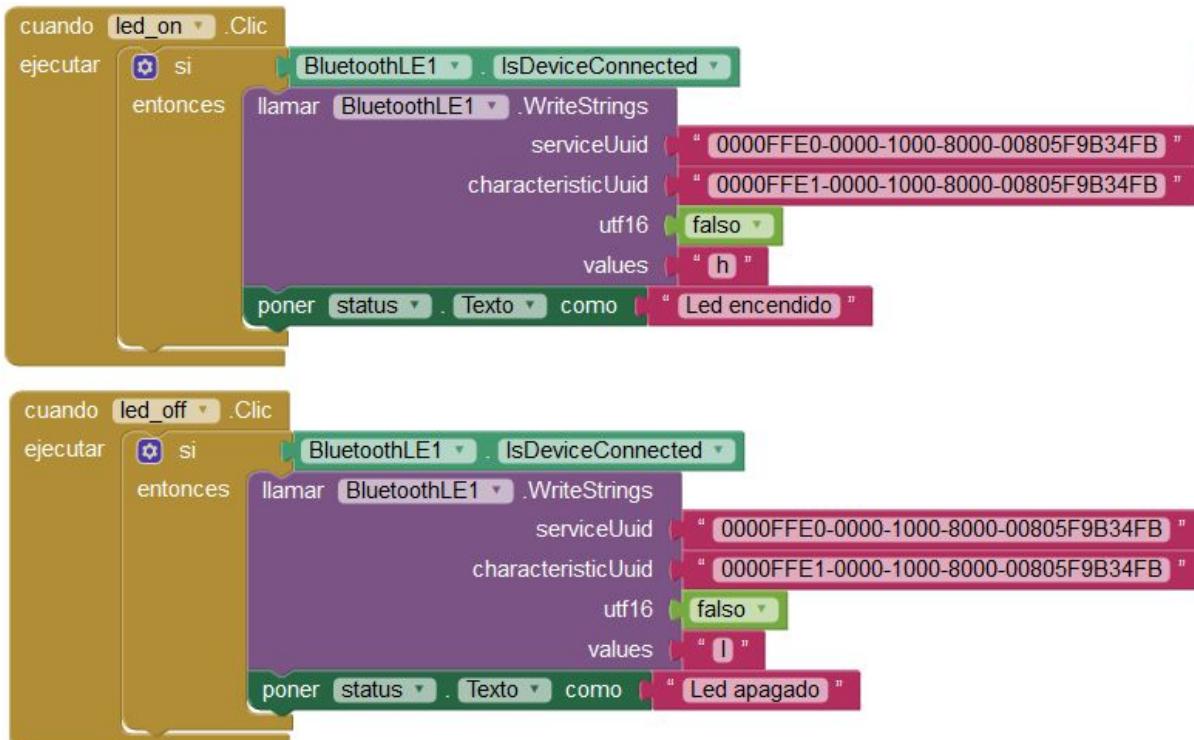
El UUID de la servicio personalizado del HM10 por defecto es:

0000FFE0-0000-1000-8000-00805F9B34FB

El UUID de la característica personalizada del HM10 por defecto es:

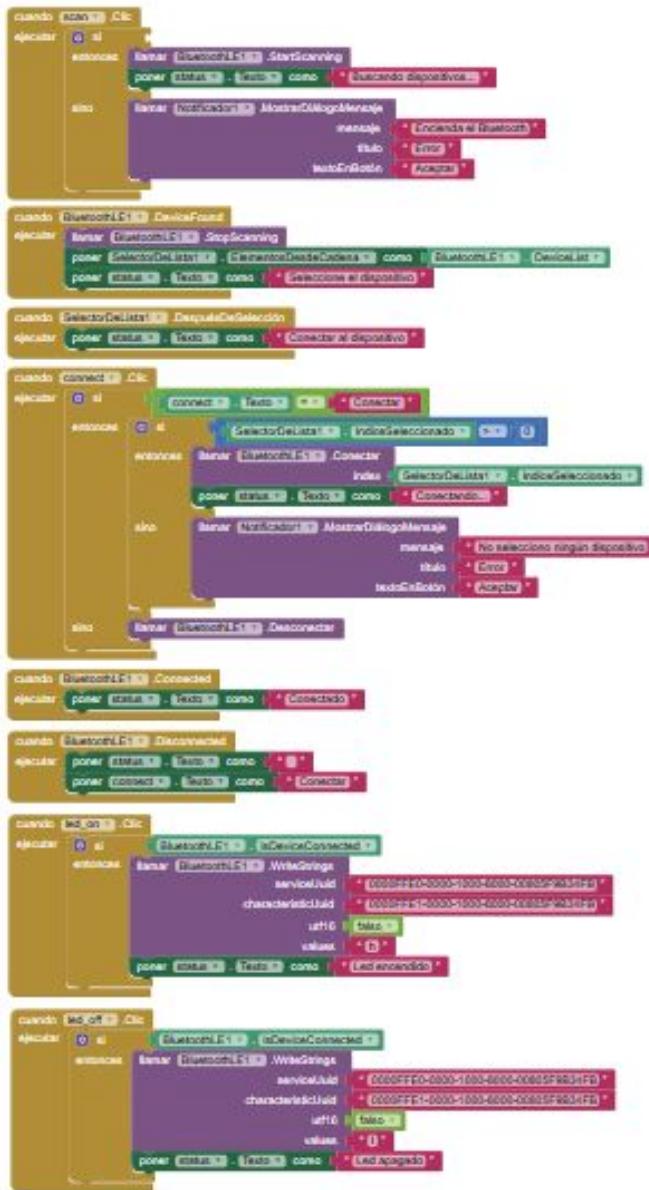
0000FFE1-0000-1000-8000-00805F9B34FB

Con estos datos se pueden programar los eventos de la siguiente forma:



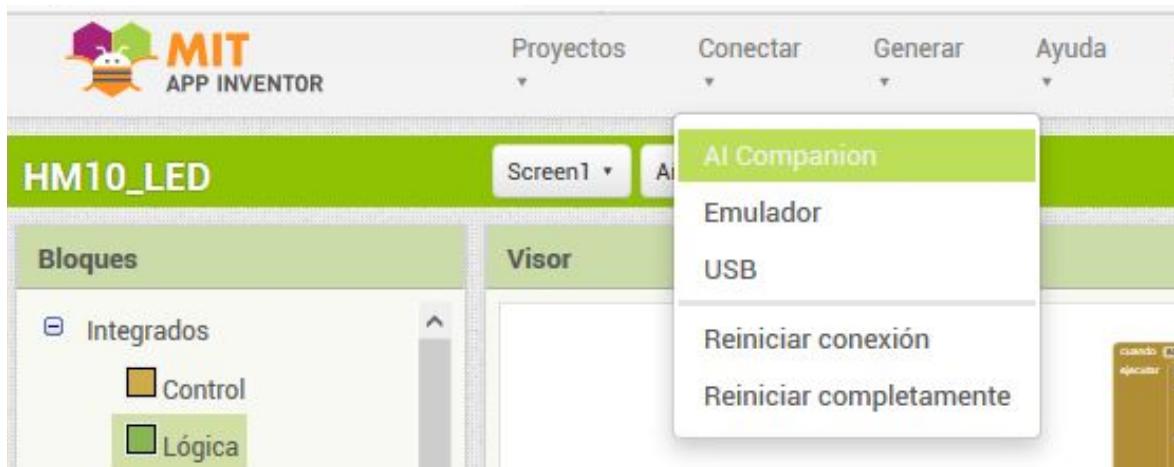
Obsérvese que cuando se presiona el botón **led_on** se envía una ‘h’ (High) la cual es interpretada por el programa en la EDU-CIAA-NXP como la orden de encendido del LEDB, mientras que el botón **led_off** envía una ‘l’ (low) que realiza la acción de apagar el LEDB.

Entonces queda el programa completo:



Comprobar su funcionamiento en el Smartphone

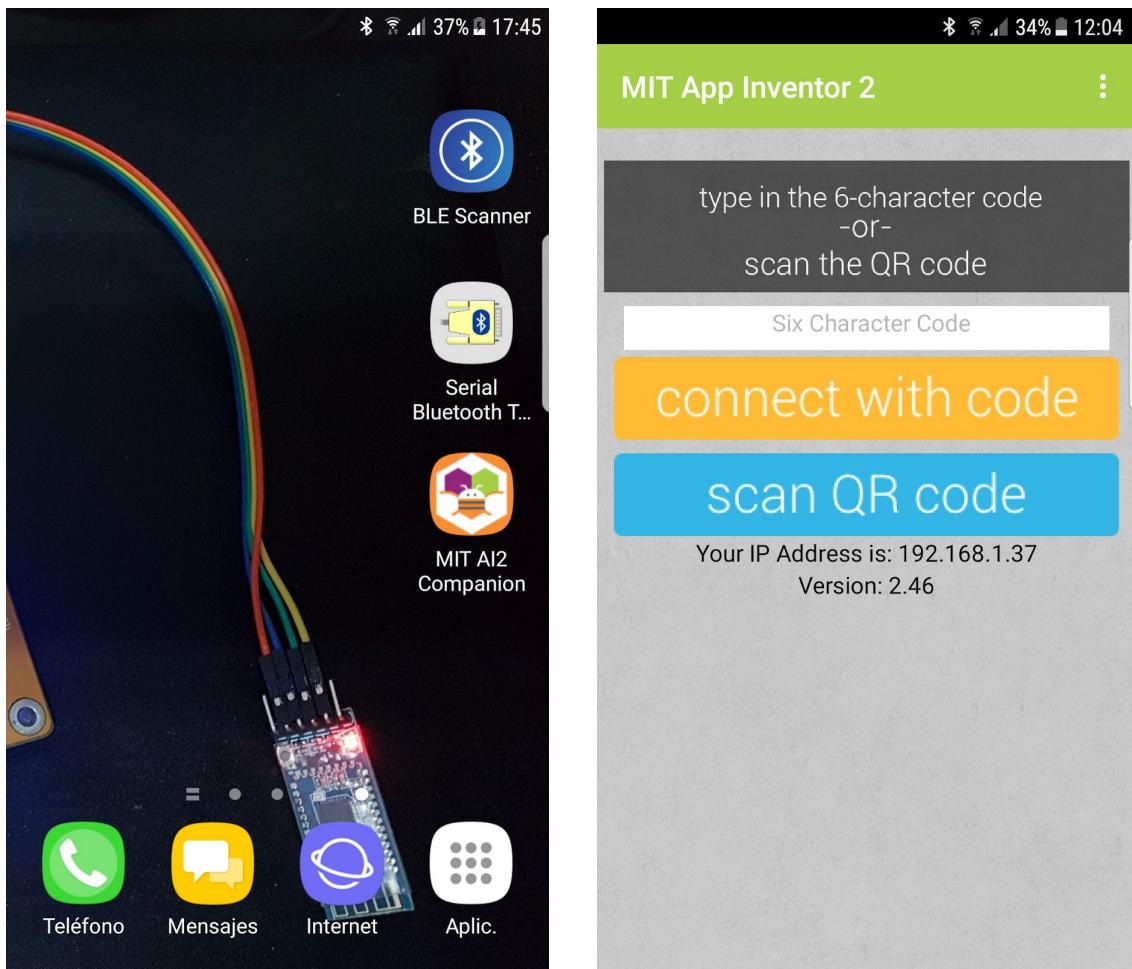
Para comprobar el funcionamiento en el Smartphone vamos a utilizar la aplicación Android “MIT AI2 Companion” descargada previamente. En el software App Inventor se debe ir al menú **Conectar** seleccionar “AI Companion”:



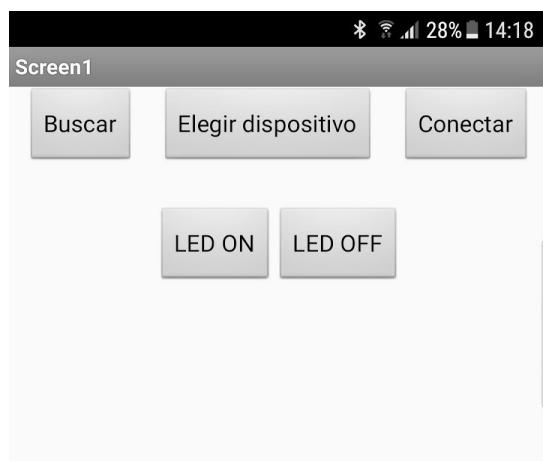
Esto mostrará un código QR el cual se deberá escanear con la aplicación “MIT AI2 Companion”:



Entonces en el Smartphone se debe **encender el bluetooth**, abrir la aplicación MIT AI2 Companion y se elige la opción **scan QR code** (es importante encender el bluetooth antes de abrir la aplicación):



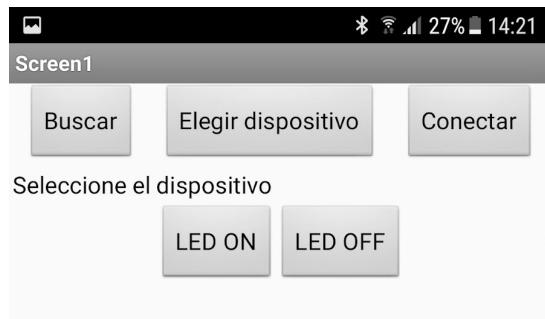
Cuando la aplicación reconoce el código QR esperar unos segundos mientras se transfiere desde App Inventor al smartphone. Luego se abrirá la aplicación desarrollada automáticamente, se debe ser paciente:



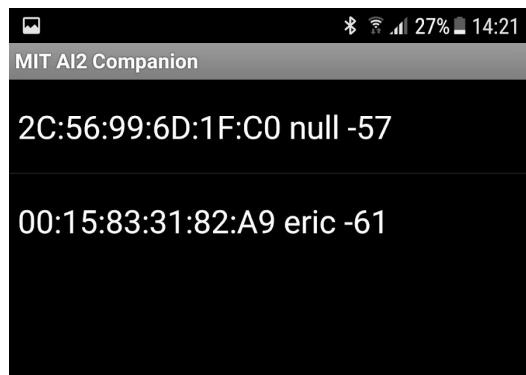
Apagar el bluetooth. Si se presiona el botón “Buscar” y el bluetooth está apagado se muestra el mensaje de error como se ha programado:



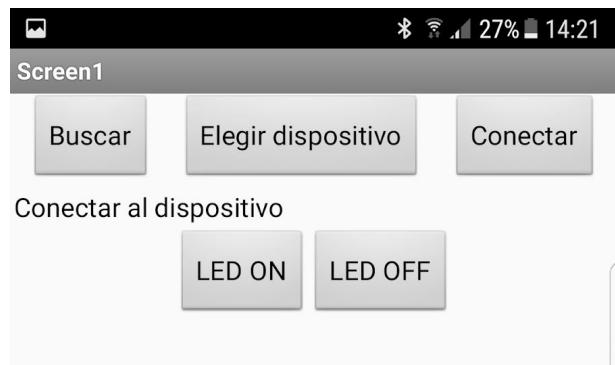
Encender el bluetooth y presionar “Buscar” nuevamente. En esta oportunidad se buscan los dispositivos y aparece en pantalla el mensaje “Seleccione el dispositivo”:



Entonces se presiona el botón “Elegir dispositivo” para elegir el módulo HM10:



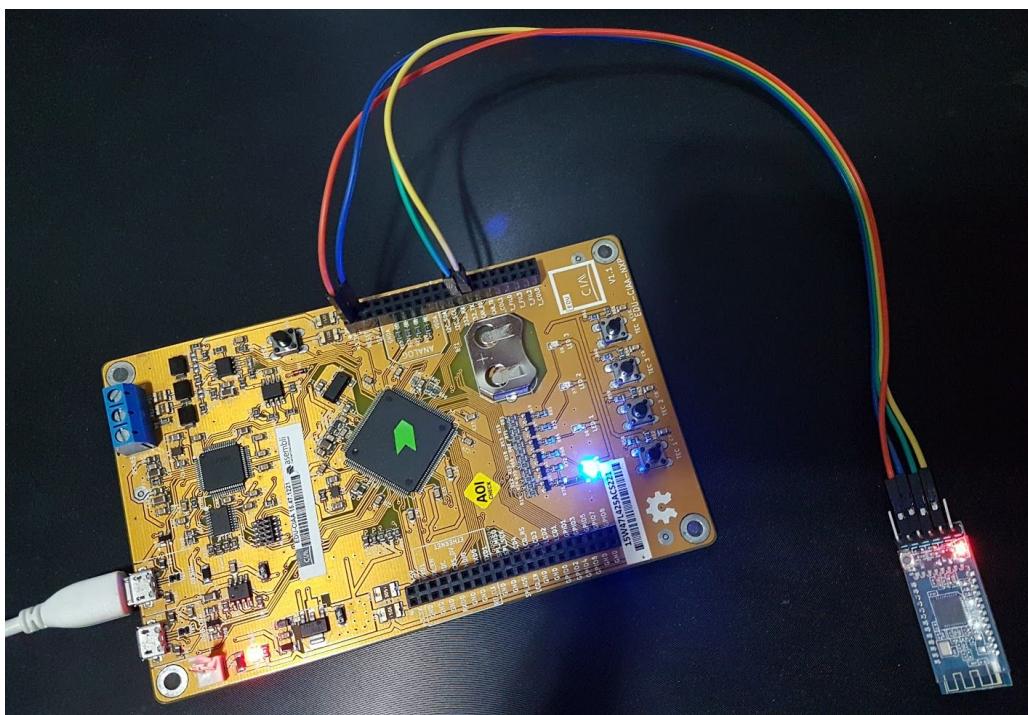
Se elige el dispositivo “00:15:83:31:82:A9 eric -61” y en la pantalla principal se muestra el mensaje “Conectar al dispositivo”:



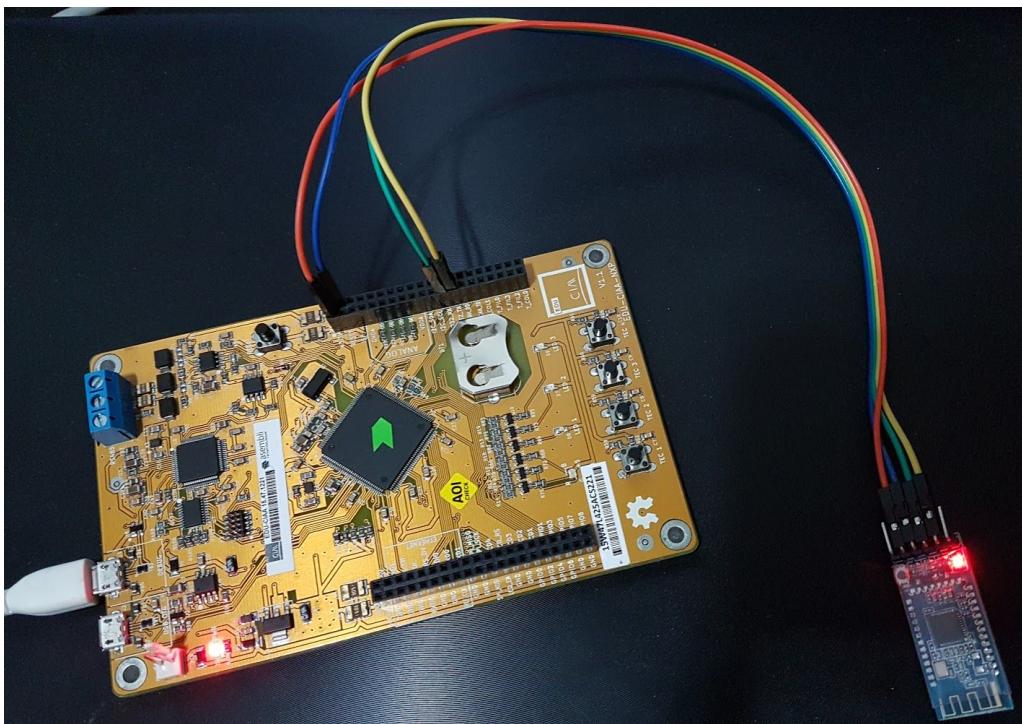
Se presiona el botón “Conectar” que realiza la conexión, cambia su texto a “Desconectar” y muestra el mensaje de estado “Conectado”:



A partir de este momento si se presionan los botones “LED ON” y “LED OFF” se puede ver como desde el Smartphone se controla mediante bluetooth el LEDB de la EDU-CIAA-NXP. Presionado “LED ON” se enciende el LED azul:



Presionado “LED OFF” se apaga el LED azul:

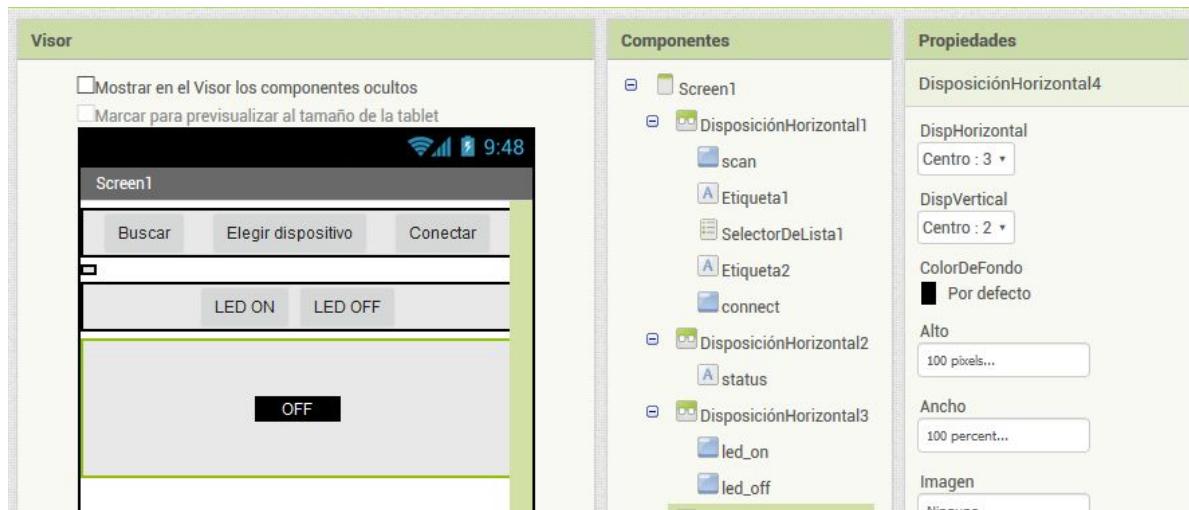


Programa HM10_LED_BOTON

Este programa se basa en el anterior, con el agregado de que la aplicación del Smartphone además, posee una etiqueta (que simula un LED) controlable desde las teclas TEC3 y TEC4 de la EDU-CIAA-NXP.

Modificar la interfaz gráfica

Para realizarla primero se modifica la interfaz gráfica para incluir esta **etiqueta**:



Como se observa, la misma se pone con color de fondo negro y el texto OFF en blanco por defecto. Para alinear en pantalla se agrega un componente **DisposiciónHorizontal** extra.

Agregar el comportamiento de recepción

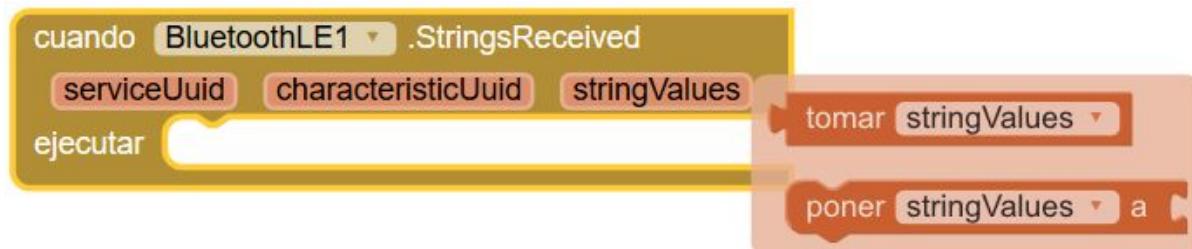
A partir de aquí es necesario programar el comportamiento de la aplicación para que permita también recibir datos desde la EDU-CIAA-NXP y controle el color de esta etiqueta.

Para recibir datos desde un dispositivo BLE se necesita saber:

- El UUID del servicio y de la característica de interés.
- El tipo de datos (byte, string, integer, float, short, long).

Como se utiliza el servicio y la característica personalizados predeterminados en el HM-10 ya se cuenta con la información de sus UUID.

La extensión BLE tiene varios bloques para recibir diferentes tipos de datos, en este ejemplo se utilizan Strings, con lo cual se necesita el evento **cuando BluetoothLE1.StringsReceived**:



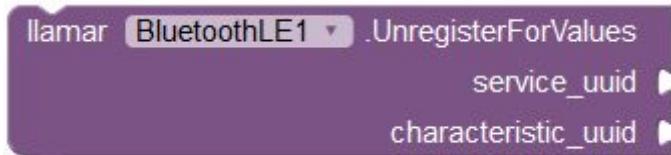
Si se presiona sobre "serviceUuid", "characteristicUuid" o "stringValues" permite elegir más bloques. Como se muestra en la imagen anterior.

Para que este bloque funcione se requiere previamente usar el bloque **llamar BluetoothLE1 .RegisterForStrings** que se colocará luego de la conexión del módulo bluetooth:

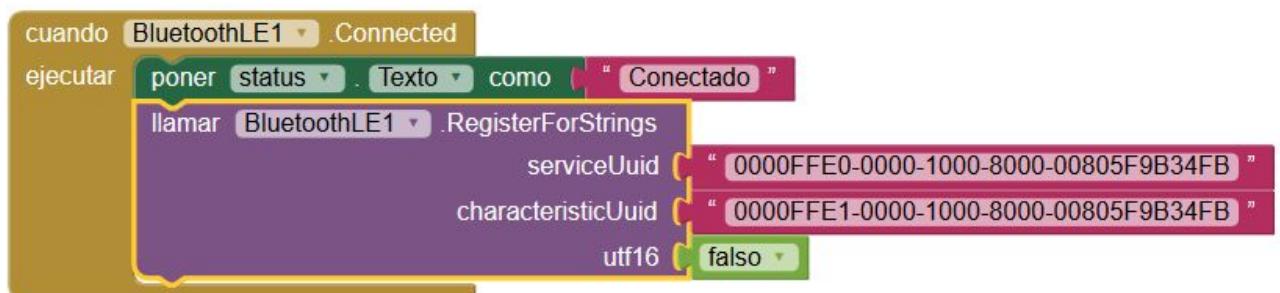
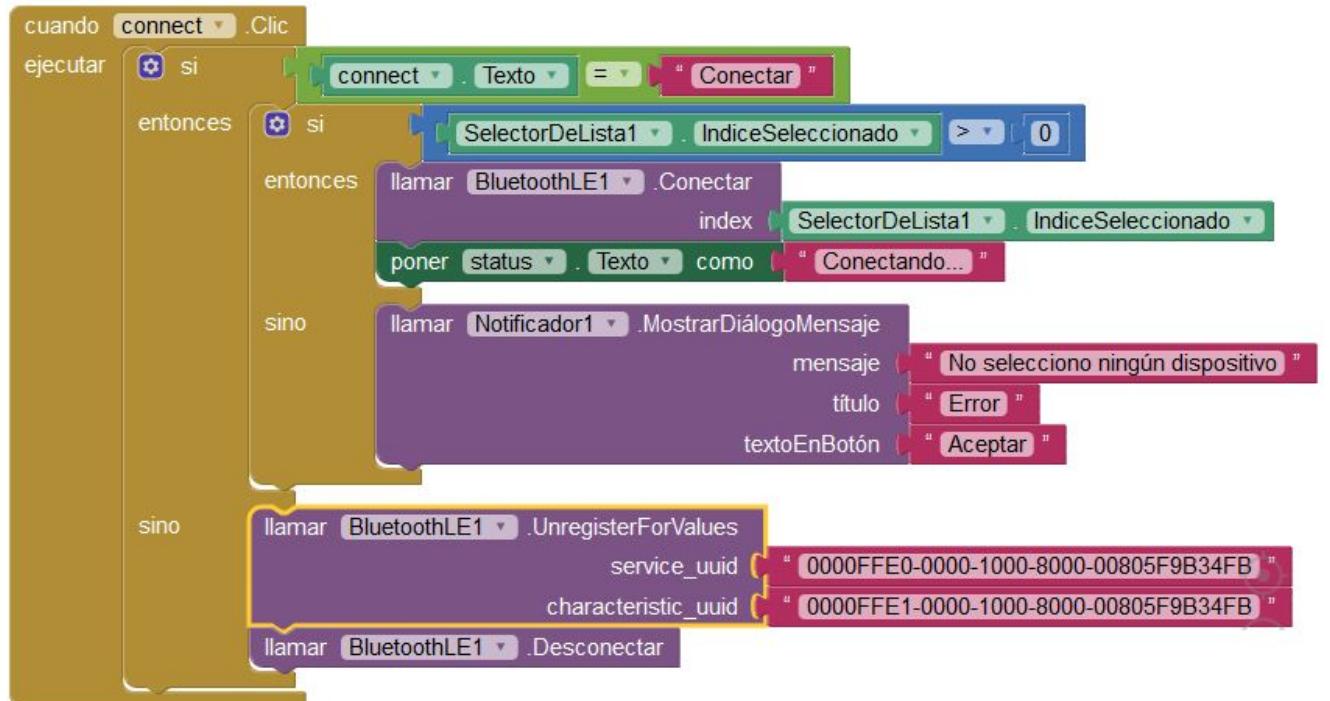


Esto depende de que la **característica** tiene una **propiedad de notificación**. Internamente AI2 le dice al módulo BLE que active las notificaciones y luego usa las notificaciones para generar el evento de datos recibidos.

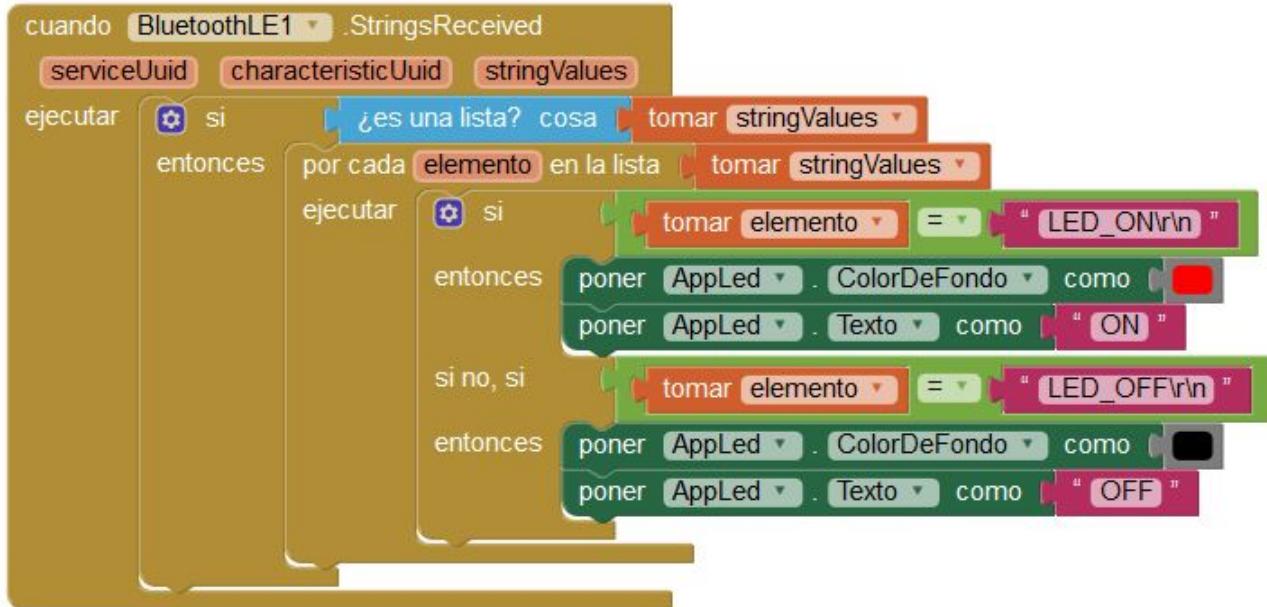
Cuando se desconecta el dispositivo, es una buena práctica anular antes de desconectar el registro o cancelar el evento de datos recibidos. Esto se realiza con el bloque **llamar BluetoothLE1 .UnregisterForValues**



De esta forma se modifican los eventos **cuando connect .Click** y **cuando BluetoothLE1 .Connected** como se muestra a continuación:



Finalmente se agrega el evento **cuando BluetoothLE1 .StringReceived** que se ejecuta cuando existe un nuevo dato recibido.



Como particularidad los valores se reciben en forma de una lista por lo cual hay que recorrerla con un bloque **por cada elemento** (for each). Se compara cada elemento recibido en la lista y en caso de coincidir con "LED_ON\r\n" se cambia el color de fondo de la etiqueta **AppLed** a rojo y se cambia su texto a "ON" para simular el encendido. En caso de coincidir con "LED_OFF\r\n" se cambia el color de fondo de la etiqueta **AppLed** a negro y se cambia su texto a "OFF" para simular el apagado.

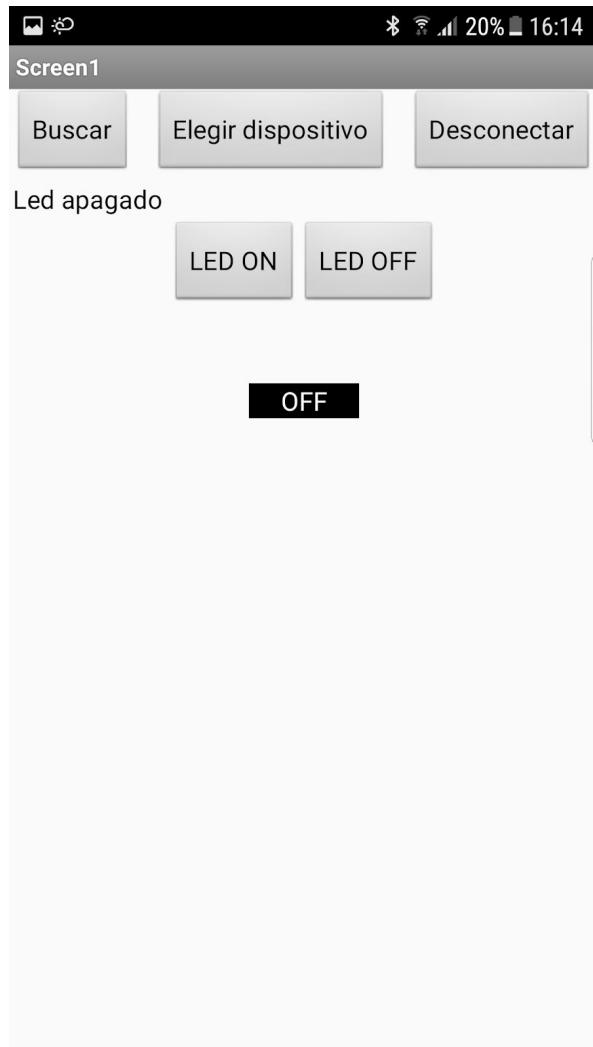
Comprobar su funcionamiento en el Smartphone

Para comprobar el funcionamiento en el Smartphone se procede a realizar los mismos pasos que en la aplicación anterior.

Una vez conectado al dispositivo se puede observar que al presionar el botón TEC3 de la EDU-CIAA-NXP "se enciende" el "LED" simulado en nuestra aplicación:



Finalmente con TEC 4 “se apaga” el “LED” simulado:



Bibliografía

- <https://roboindia.com/tutorials/ble-4.0-arduino-led-control-mit-app-inventor>
- <http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/>
- http://wiki.makespacemadrid.org/index.php?title=M%C3%B3dulo_HM-10