

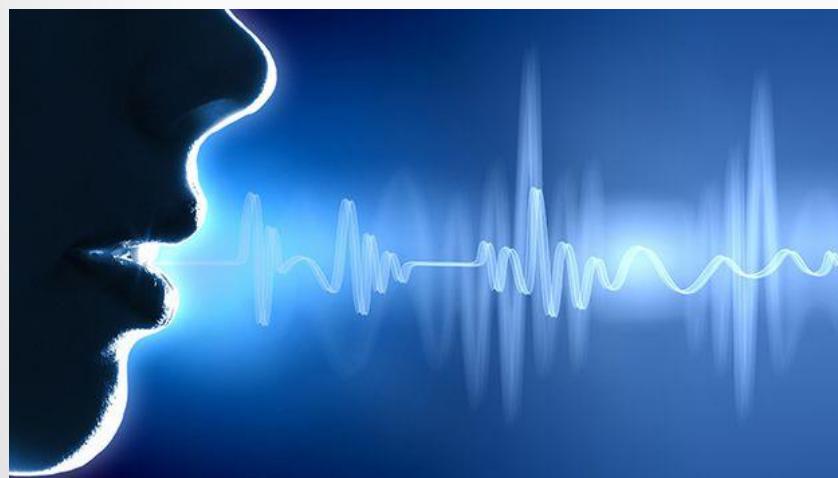
RECONOCIMIENTO DE VOZ EN ARDUINO

TUTORA: ANGELA JAZMIN MIRANDA FLORES



¿QUÉ ES?

El control por voz es una disciplina de la inteligencia artificial que tiene como objetivo permitir la comunicación entre seres humanos y computadoras.



TUTORA: ANGELA JAZMIN MIRANDA FLORES

¿QUÉ ES UN SISTEMA DE RECONOCIMIENTO DE VOZ?

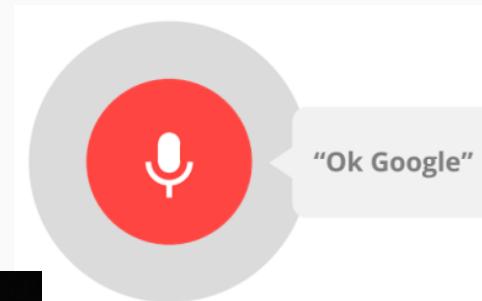
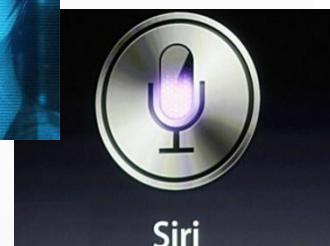
Es una herramienta computacional capaz de procesar la señal de la voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o ejecutando una acción.



TUTORA: ANGELA JAZMIN MIRANDA FLORES

¿EN QUÉ SE UTILIZA?

- Control de casas domóticas
- Automóviles de última generación
- Smartphone
- Asistentes personales, etc.



TUTORA: ANGELA JAZMIN MIRANDA FLORES



¿Como lo realizamos?

TUTORA: ANGELA JAZMIN MIRANDA FLORES

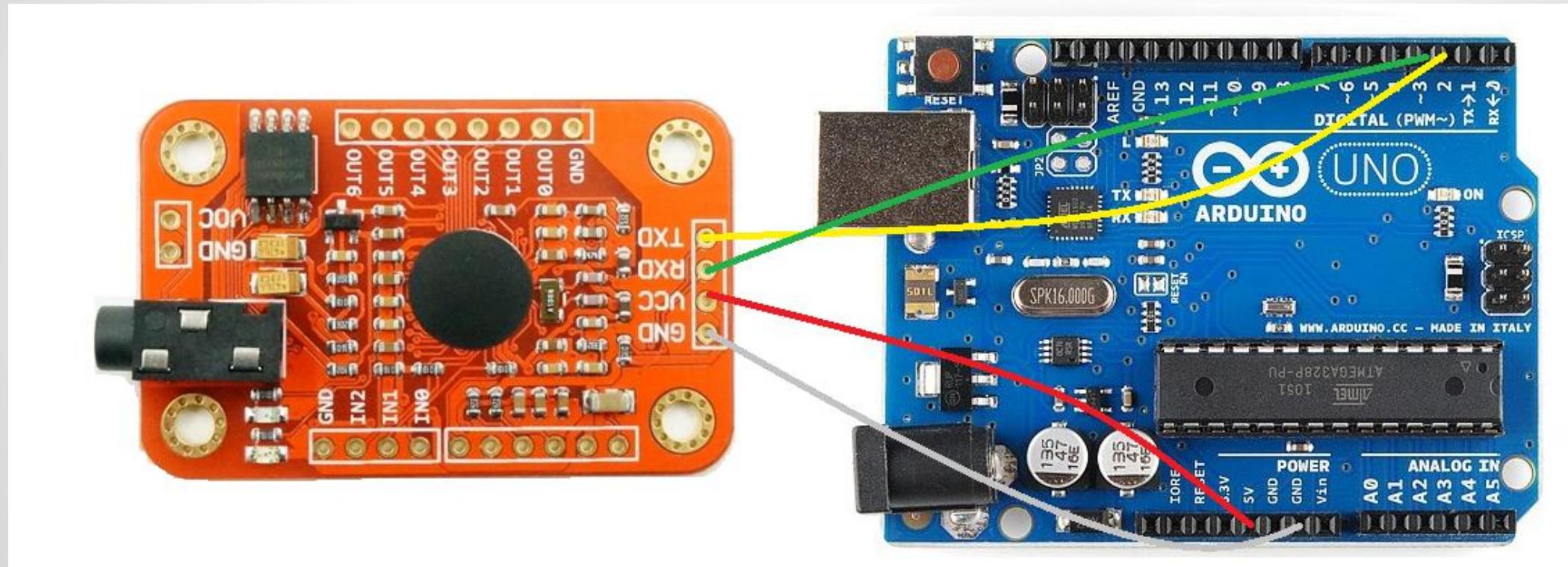
Módulo de reconocimiento de voz

Módulo Elechouse Voice Recognition V3

- Puede almacenar 80 comandos de voz
- Solo se pueden utilizar 7 a la vez
- Los comandos solo pueden durar 2 o 3 seg.
- Utiliza su librería ***voicerecognitionv3***
- Voltaje de 4.5 a 5 V.
- Se programa con un adaptador USB a serial
- Velocidad de conexión 9600 - 115200 baudios



Conexión del módulo de reconocimiento de voz



TUTORA: ANGELA JAZMIN MIRANDA FLORES

Configuración por puerto serial

```
Elechouse Voice Recognition V3 Module "train" sample.

Usage:
-----

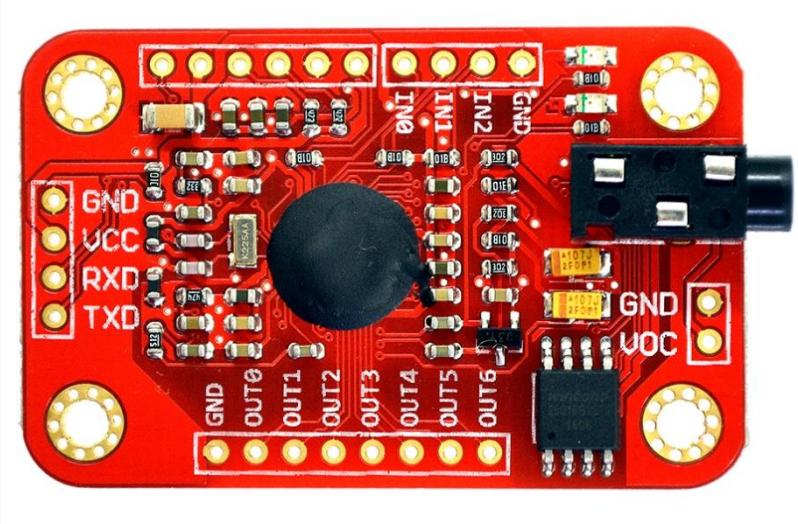
COMMAND      FORMAT          EXAMPLE           Comment
-----
train        train (r0) (r1)...   train 0 2 45    Train records
load         load (r0) (r1) ...  load 0 51 2 3   Load records
clear        clear             clear            remove all records in Recognizer
record       record / record (r0) (r1)... record / record 0 79 Check record train status
vr           vr                vr               Check recognizer status
getsig       getsig (r)        getsig 0        Get signature of record (r)
sigtrain     sigtrain (r) (sig) sigtrain 0 ZERO Train one record(r) with signature(sig)
settings     settings          settings         Check current system settings
help         help              help            print this message
-----
```

Autoscroll Ambos NL & CR 115200 baudio

TUTORA: ANGELA JAZMIN MIRANDA FLORES

Configuración

- LED naranja parpadea rápido: Prepárate.
- LED rojo fijo: Grabar comando primera vez.
- LED naranja parpadea lento: Prepárate para la segunda grabación.
- LED rojo fijo: Grabar comando segunda vez.
- LED rojo y naranja parpadeando a la vez: Los comandos coinciden y se guardan.



TUTORA: ANGELA JAZMIN MIRANDA FLORES

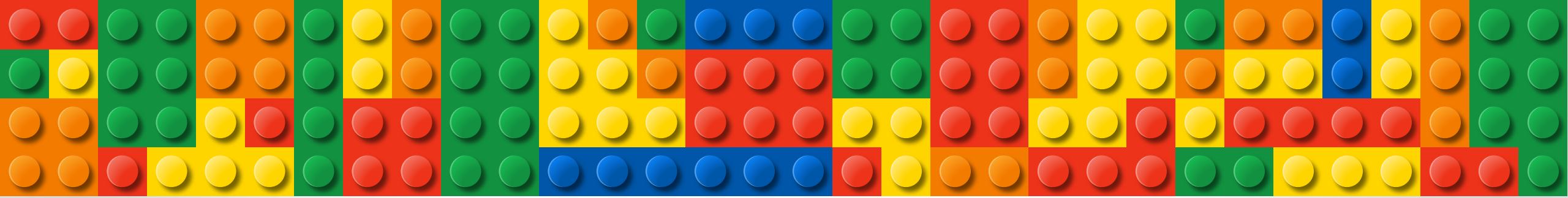
Desventajas

- Solo se pueden utilizar 7 comandos como máximo por programa.
- Para almacenar los comandos deben coincidir con el comando anteriormente grabado.
- Solo reconoce la palabra en el mismo tono en el que fue grabado.
- La programación es un tanto compleja pese a utilizar una librería.

```
COM2 (Arduino/Genuino Uno)

-----
sigtrain 0 Encender
-----
Record: 0      Speak now
Record: 0      Speak again
Record: 0      Cann't matched
Record: 0      Speak now
Record: 0      Speak again
Record: 0      Success
Success: 1
Record 0      Trained
SIG: Encender
-----
sigtrain 1 Apagar
-----
Record: 1      Speak now
Record: 1      Speak again
Record: 1      Success
Success: 1
Record 1      Trained
SIG: Apagar
```

TUTORA: ANGELA JAZMIN MIRANDA FLORES



¿CÓMO LO REALIZAMOS UTILIZANDO EL MÓDULO BLUETOOTH?

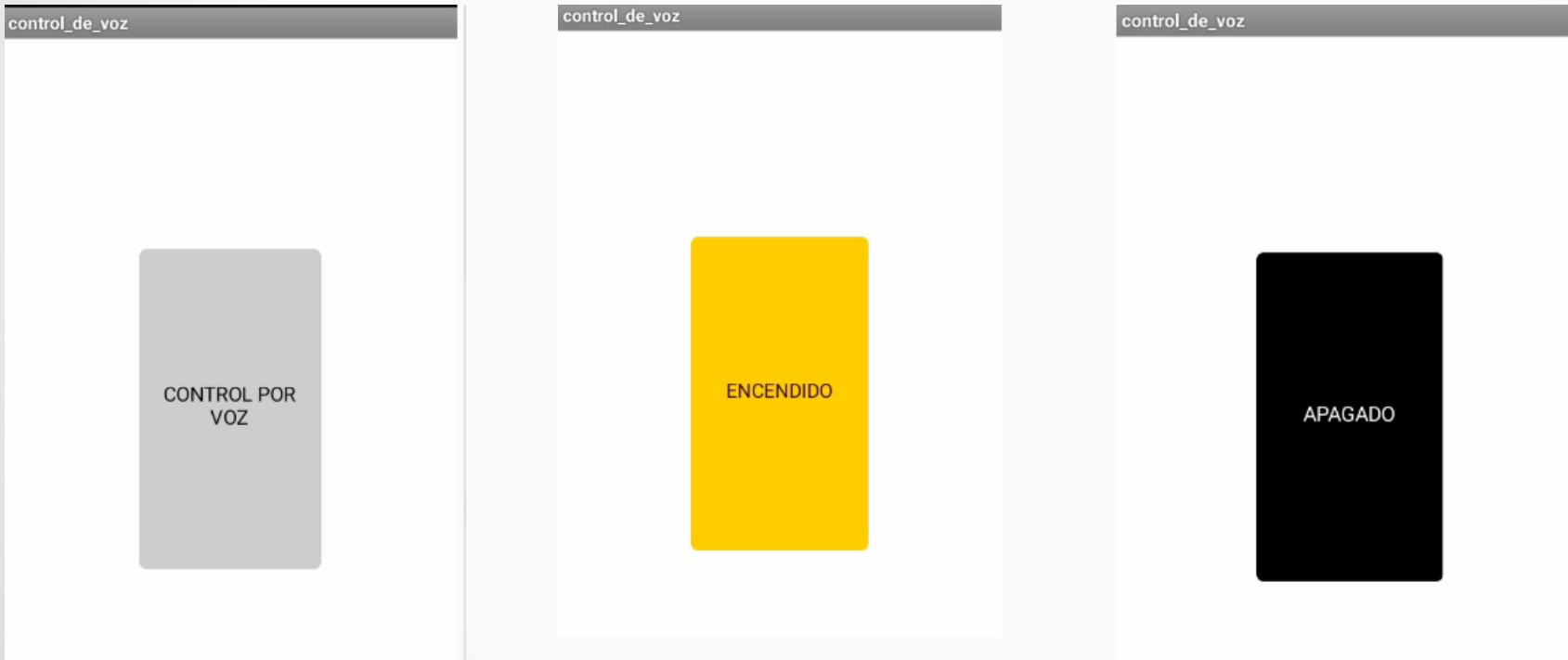


TUTORA: ANGELA JAZMIN MIRANDA FLORES



EJERCICIO DE APLICACIÓN

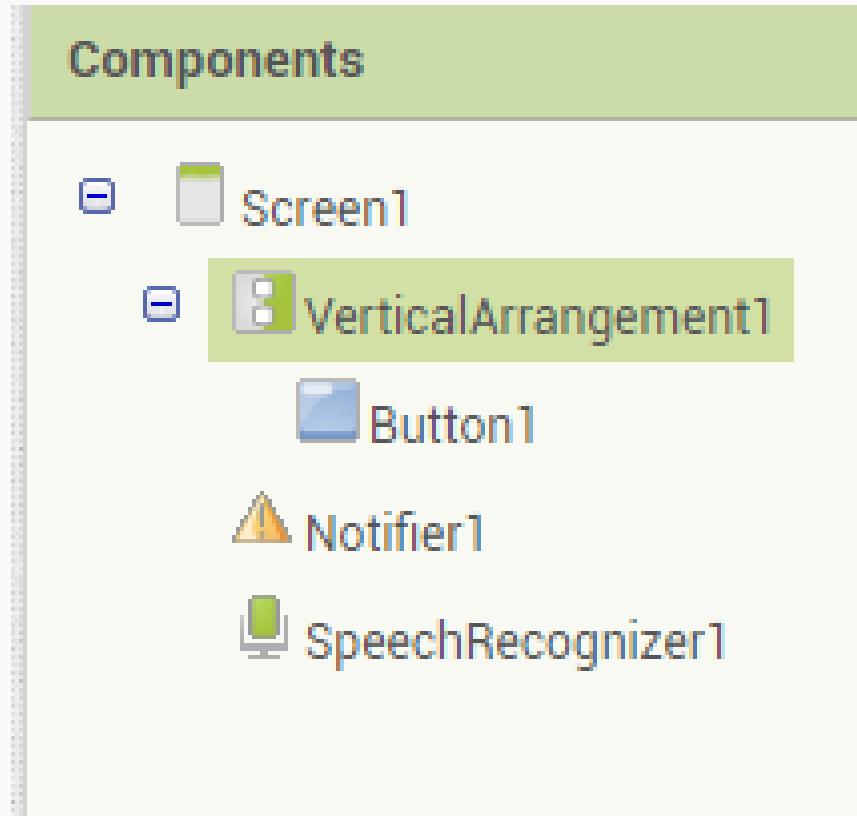
Realizar una aplicación que cambie el color y el texto de un botón a través de app inventor utilizando el reconocimiento de voz; el texto debe ser “encendido” y “apagado”.



TUTORA: ANGELA JAZMIN MIRANDA FLORES



COMPONENTES UTILIZADOS

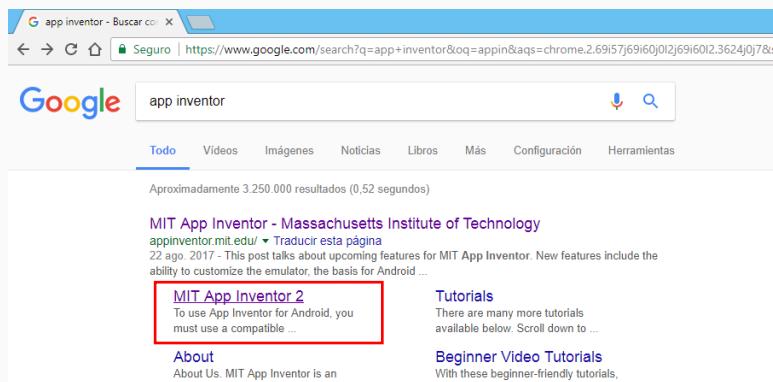


TUTORA: ANGELA JAZMIN MIRANDA FLORES

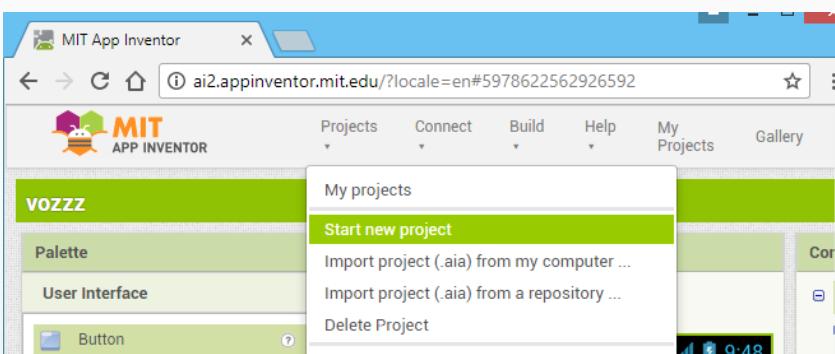


¿Cómo lo realizamos?

1. Ingresamos al sitio de app inventor



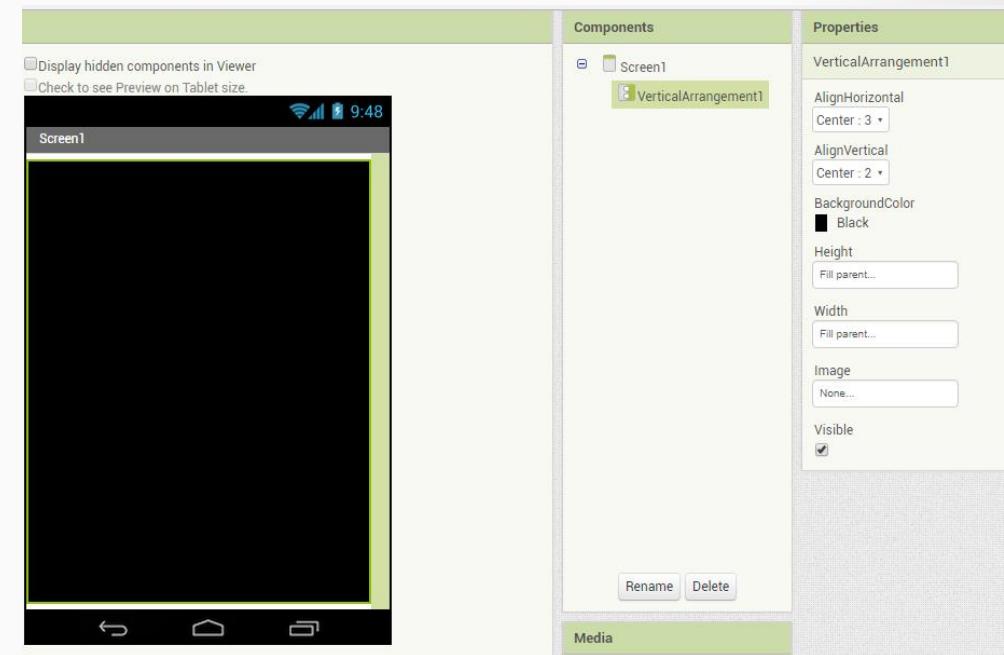
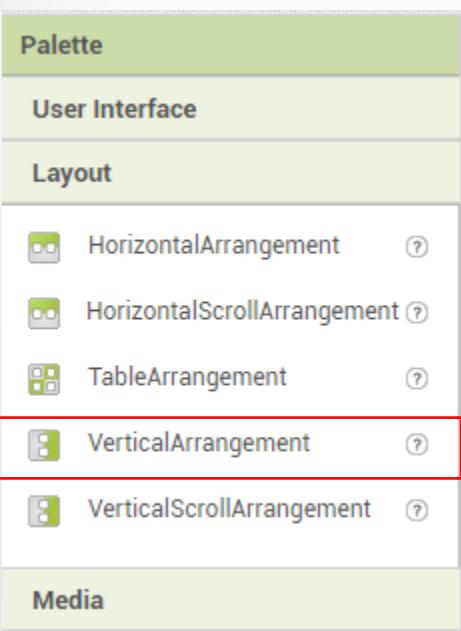
2. Creamos un nuevo proyecto dentro nuestra cuenta en app inventor.





¿Cómo lo realizamos?

3. En la paleta de app inventor buscamos la sección de Layout, de la cual utilizaremos el alineamiento vertical, lo arrastramos a nuestra app y le cambiamos las propiedades (ver gráficos).





¿Cómo lo realizamos?

4. Ahora en la sección interfaz de usuario arrastramos un botón a nuestra app y configuraremos sus propiedades (ver gráficos).

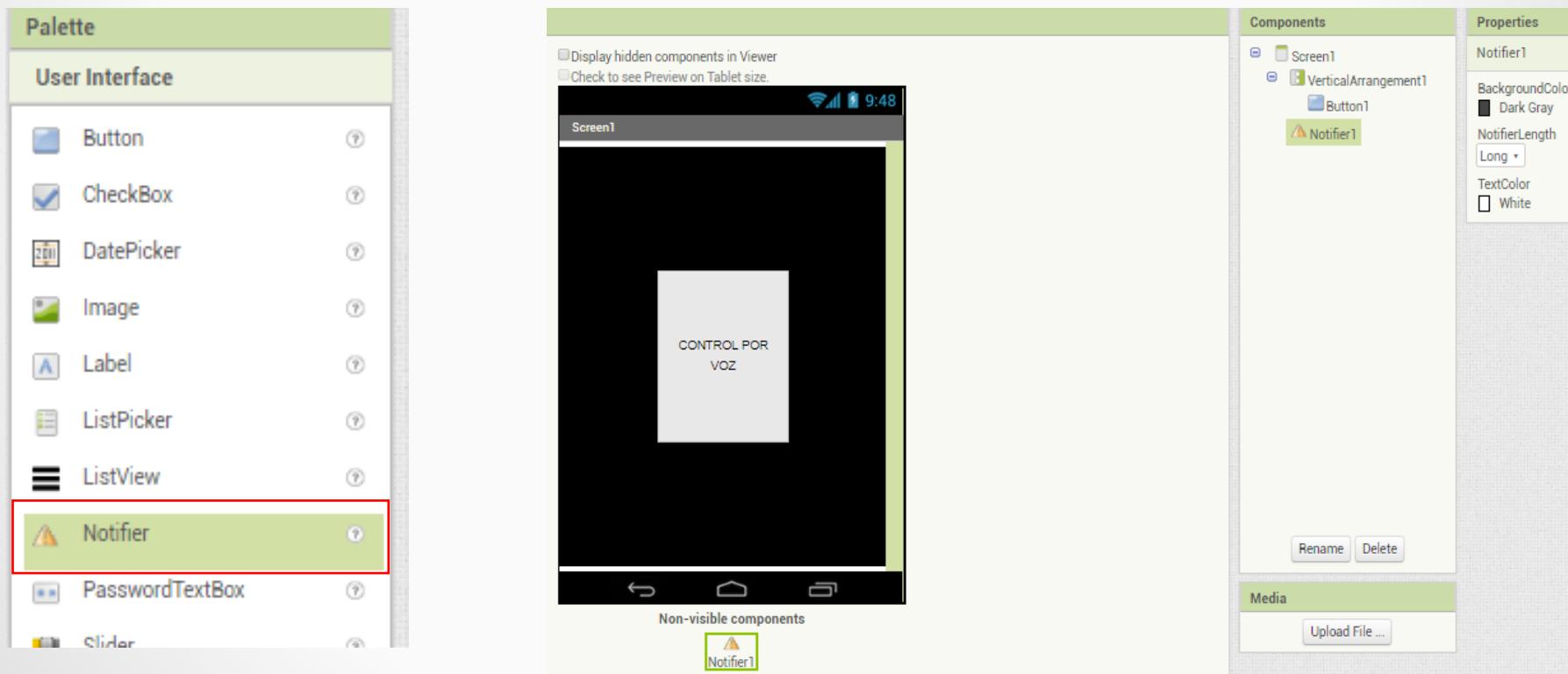
The screenshot shows a software interface for mobile application development. On the left, a **Palette** window titled **User Interface** contains icons for various UI components: **Button**, **CheckBox**, **DatePicker**, **Image**, and **Label**. The **Button** icon is highlighted with a red border. In the center, a preview window shows a smartphone screen with the text "CONTROL POR VOZ". To the right, the **Components** pane lists "Screen1" containing a "VerticalArrangement1" which holds a "Button1". The **Properties** pane on the far right provides detailed settings for "Button1", including:

- BackgroundColor**: Default
- Enabled**: checked
- FontBold**: unchecked
- FontItalic**: unchecked
- FontSize**: 14.0
- FontTypeface**: default
- Height**: 40 percent
- Width**: 40 percent
- Image**: None
- Shape**: default
- ShowFeedback**: checked
- Text**: CONTROL POR VOZ
- TextAlignment**: center
- TextColor**: Default
- Visible**: checked



¿Cómo lo realizamos?

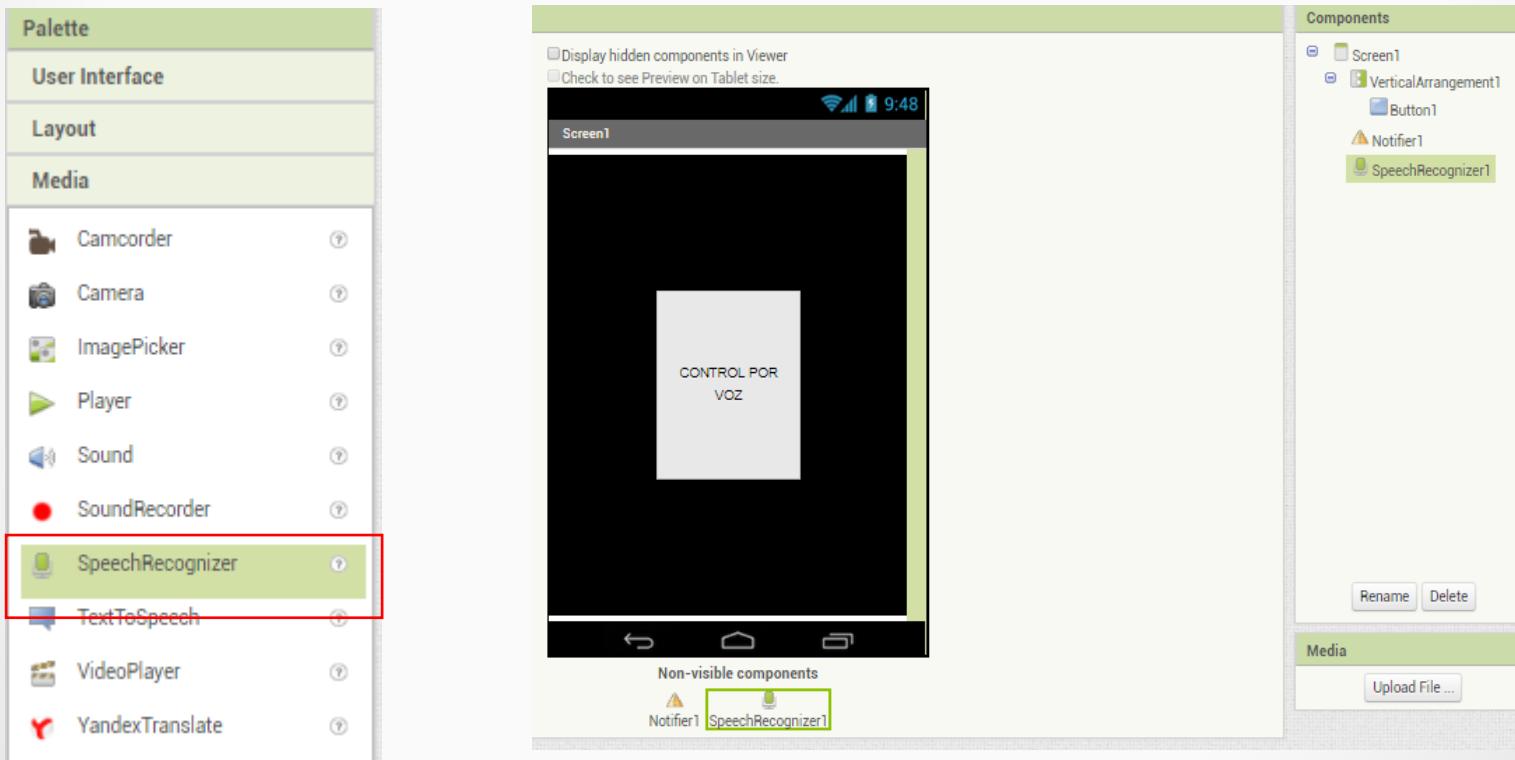
5. En la sección interfaz de usuario arrastramos una notificación a nuestra app (ver gráficos).





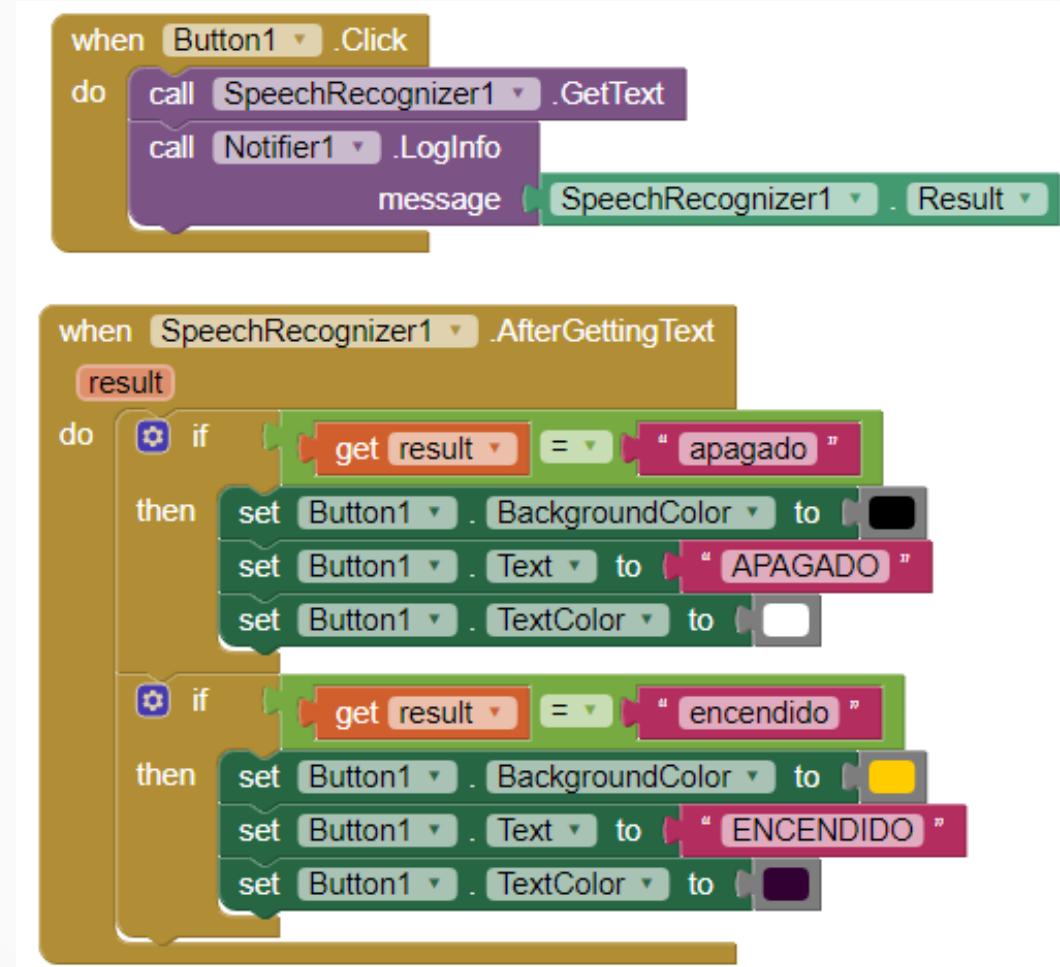
¿Cómo lo realizamos?

6. En la paleta buscamos la sección de medios y arrastramos el reconocimiento de voz a nuestra app (ver gráficos).





CÓDIGO POR BLOQUES EN APP INVENTOR



TUTORA: ANGELA JAZMIN MIRANDA FLORES



¿Cómo lo realizamos?

7. Vamos a bloques luego de terminar el diseño

The image shows the App Inventor development environment. On the left, there is a preview window of a mobile screen titled "Screen1" with the text "CONTROL POR VOZ". Below the preview are sections for "Non-visible components" and "Notifier1 SpeechRecognizer1". To the right of the preview are two tabs: "Designer" (highlighted with a red box) and "Blocks". Underneath these tabs are panels for "Components" (listing "Screen1", "VerticalArrangement1", "Button1", "Notifier1", and "SpeechRecognizer1") and "Properties" (showing settings for "Screen1"). In the center, there is a "prueba" workspace with a "Blocks" tab selected. This workspace contains a tree view of components and a large empty area for writing blocks. A blue arrow points from the text "Editor de código por bloques de app inventor" towards the workspace.

Editor de código por bloques de app inventor



¿Cómo lo realizamos?

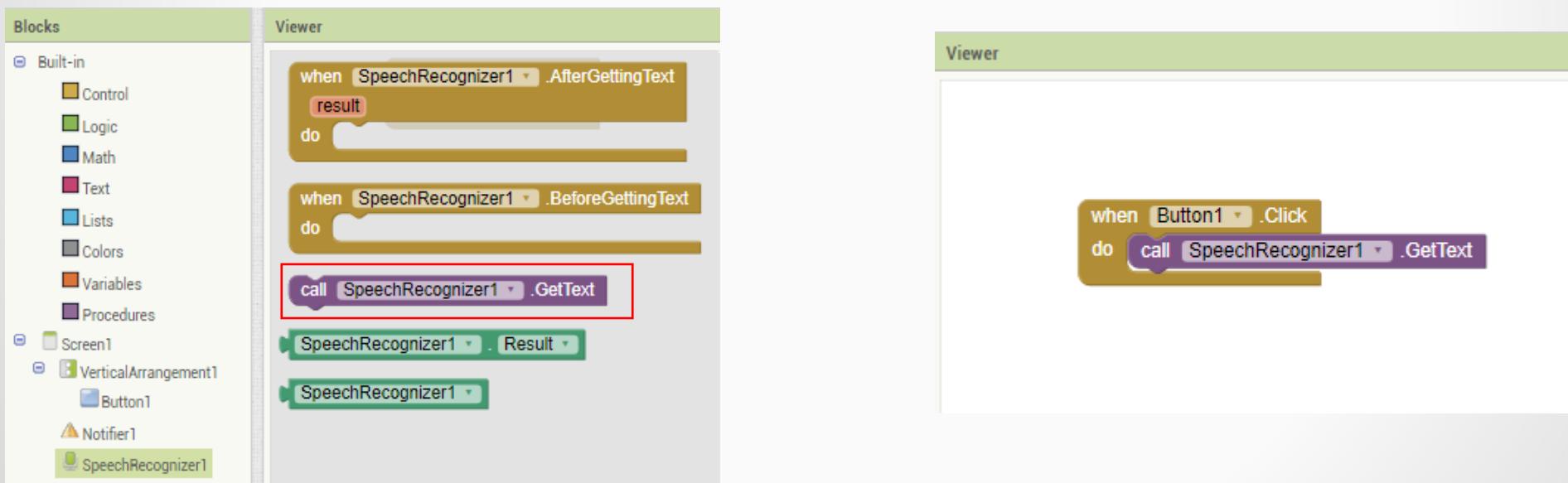
8. Hacemos click en el botón de nuestra app y nos aparece una ventana con las acciones que se puede realizar, seleccionamos la que nos permite realizar una acción cuando presionamos nuestro botón y lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

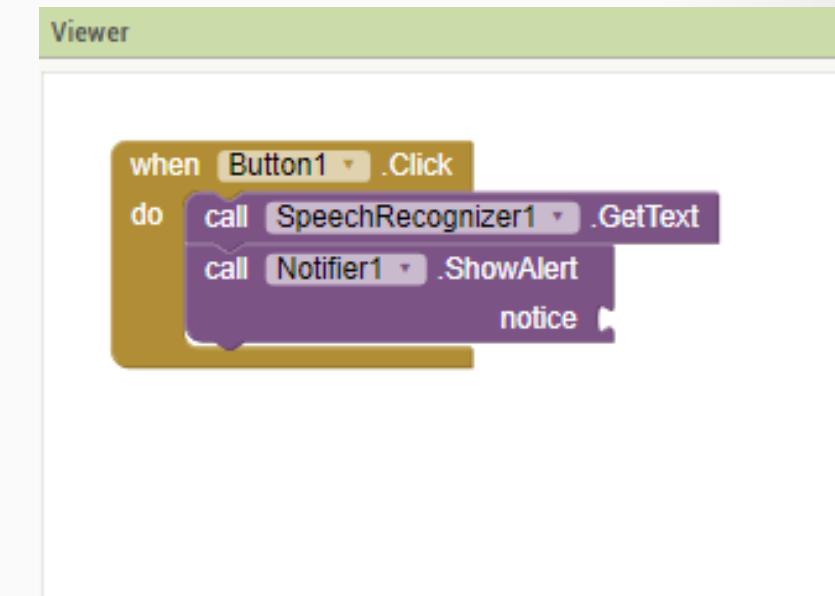
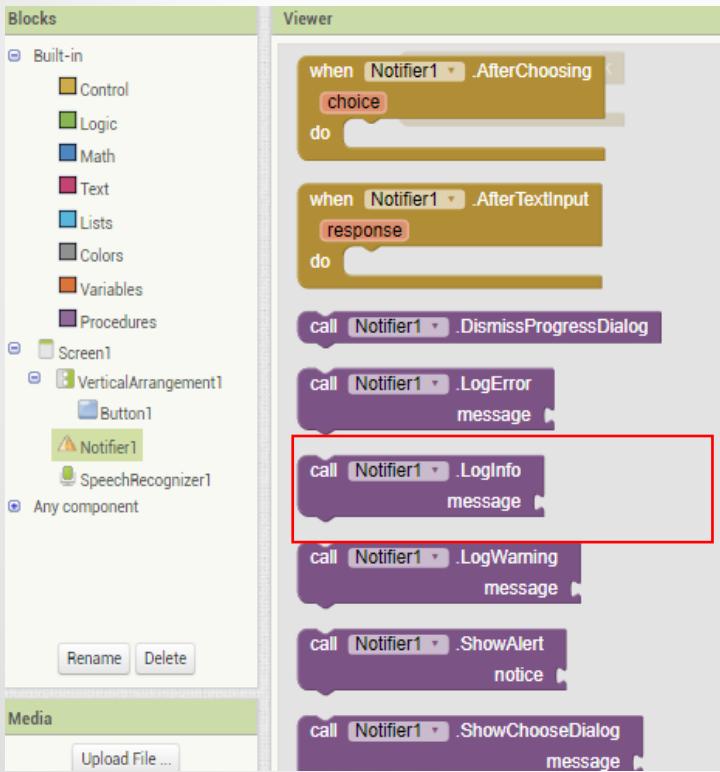
9. Hacemos click sobre el reconocimiento de voz de nuestra app y elegimos la opción de llamar al reconocimiento de voz de nuestro Smartphone; lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

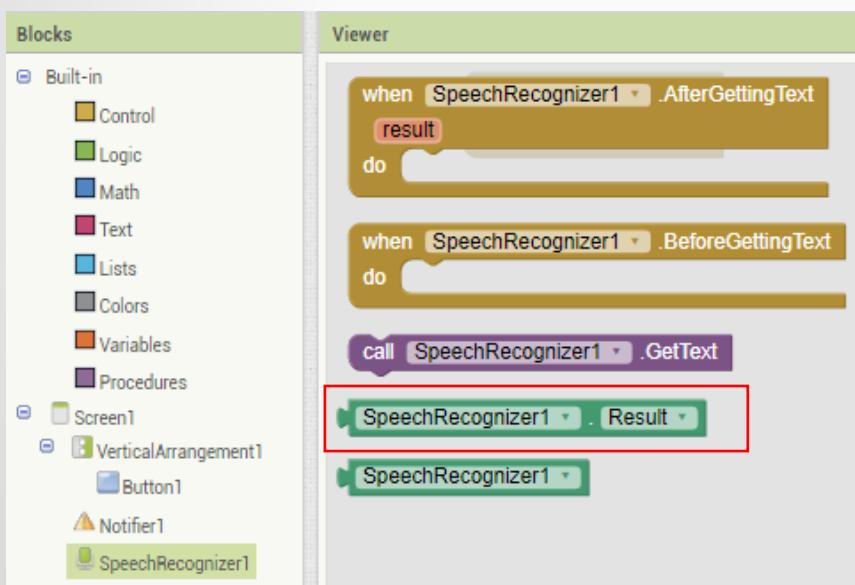
10. Hacemos click sobre la notificación nuestra app, elegimos la opción mostrar alerta y lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

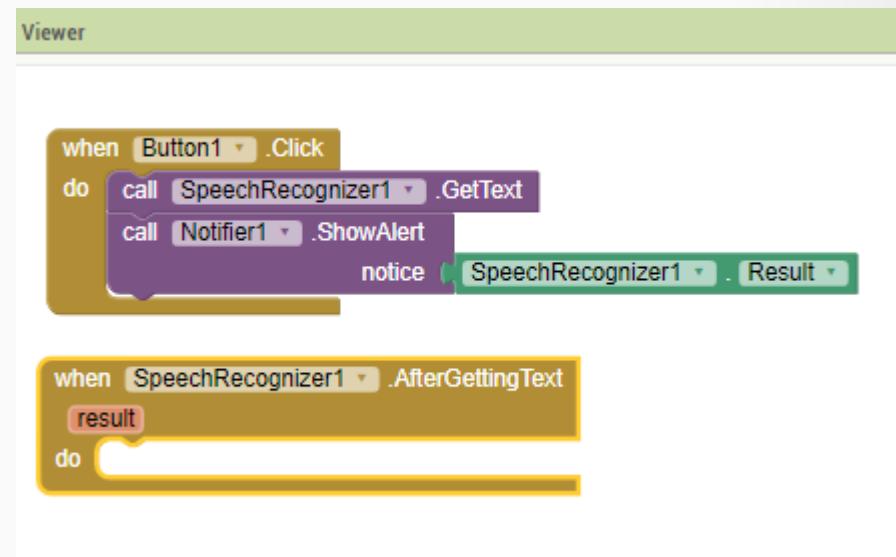
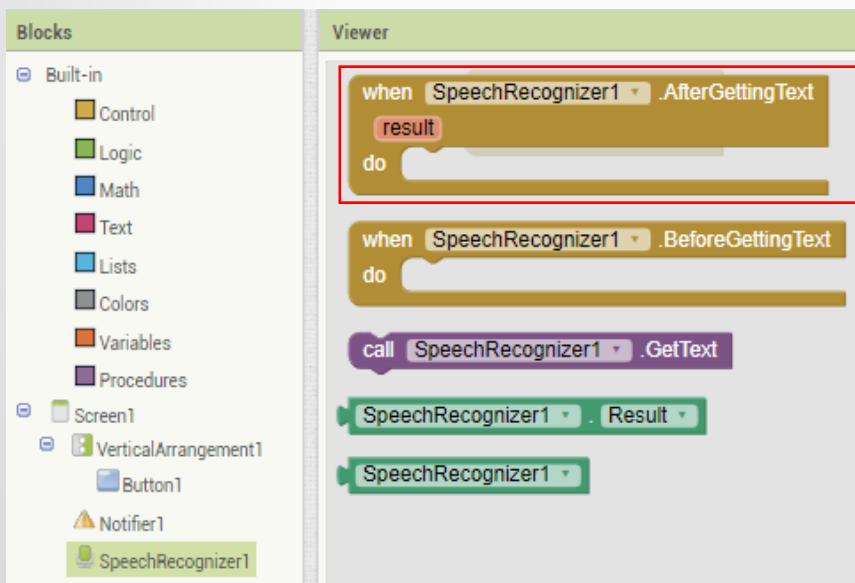
11. Hacemos click sobre el reconocimiento de voz de nuestra app y elegimos la opción del resultado del reconocimiento de voz; lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

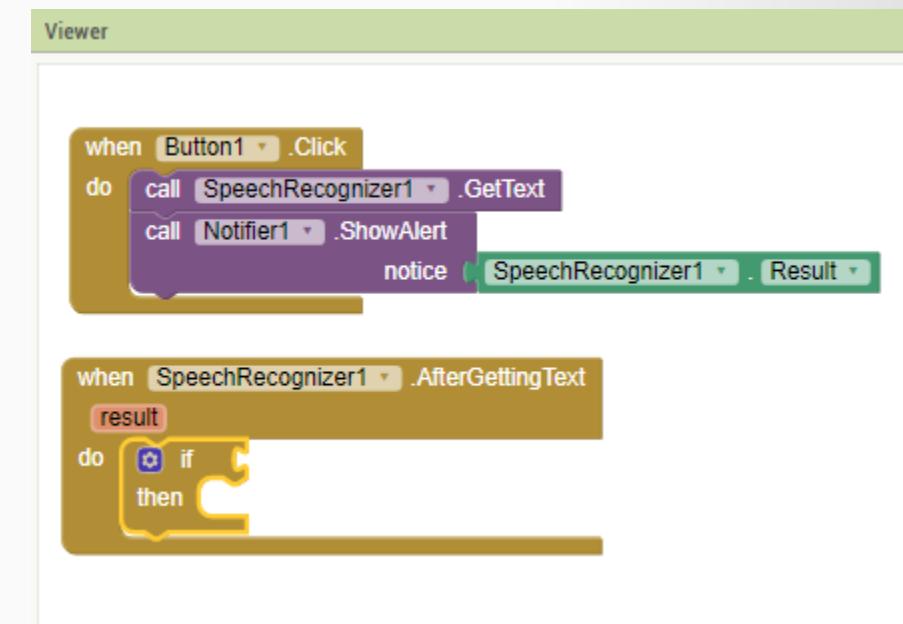
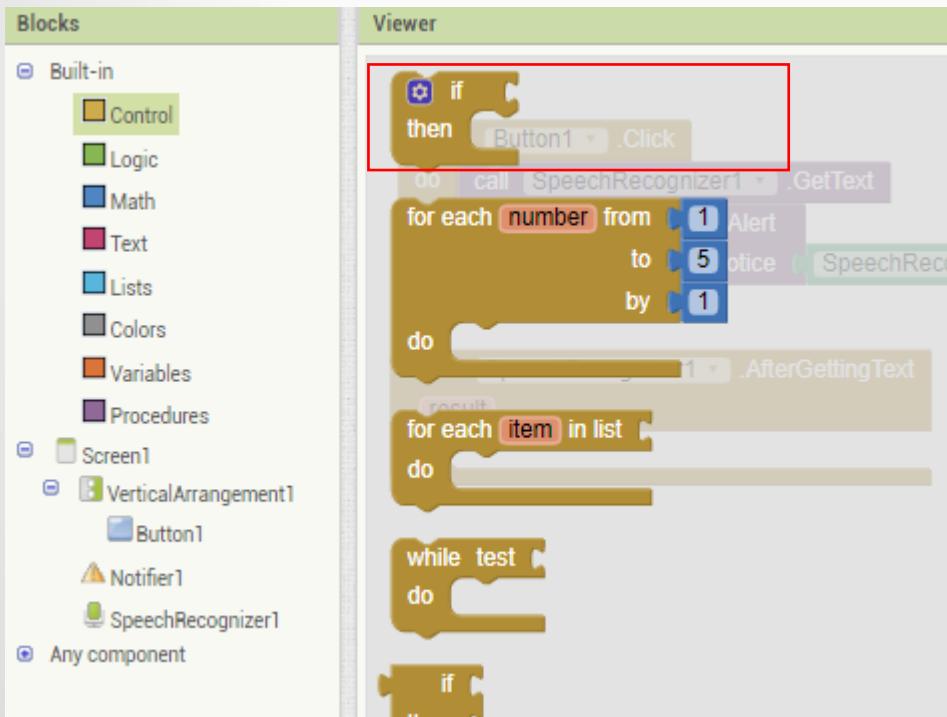
12. Hacemos click sobre el reconocimiento de voz de nuestra app y elegimos la opción antes de que el reconocimiento de voz se vuelva texto; lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

13. En control, elegimos la opción if, lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

14. En logic, elegimos la opción para comparar y lo arrastramos a nuestro editor (ver gráficos).

The image shows the Scratch interface. On the left, the 'Blocks' palette is open, showing the 'Logic' category under 'Built-in'. A red box highlights the 'equal to' block (a green flag with two puzzle pieces and an equals sign). In the center, the 'Viewer' window displays a script:

```
when Button1 .Click
do
  call SpeechRecognizer1 .GetText
  call Notifier1 .ShowAlert
  notice SpeechRecognizer1 .Result
```

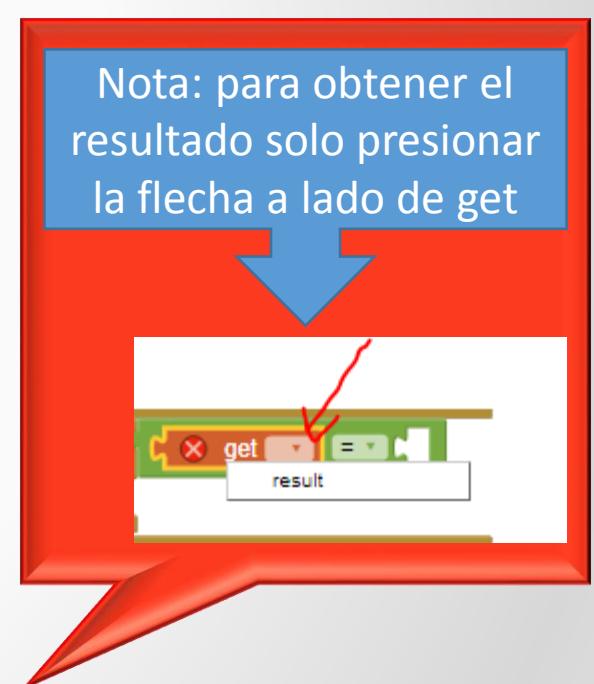
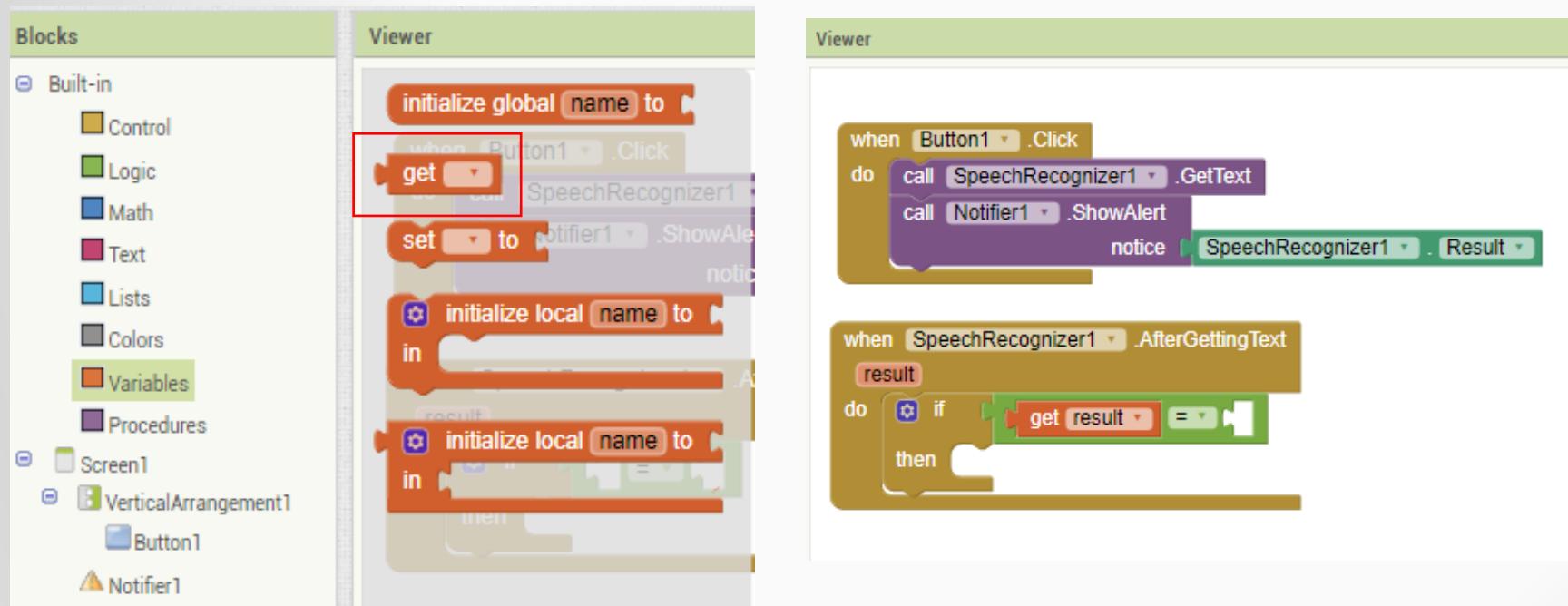
Below this, another script is shown:

```
when SpeechRecognizer1 .AfterGettingText
result
do
  if = then
```



¿Cómo lo realizamos?

15. En variables, elegimos la opción get y lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

16. En texto elegimos el bloque que nos permite escribir cadenas y lo arrastramos a nuestro editor; en ella escribimos el texto a comparar (ver gráficos).

The image shows the Scratch script editor with two scripts loaded:

- Script 1 (Top):** Triggered by "when Button1 Clicked". It calls "SpeechRecognizer1 .GetText" and "Notifier1 .ShowAlert". It also has a "notice" message to "SpeechRecognizer1 .Result".

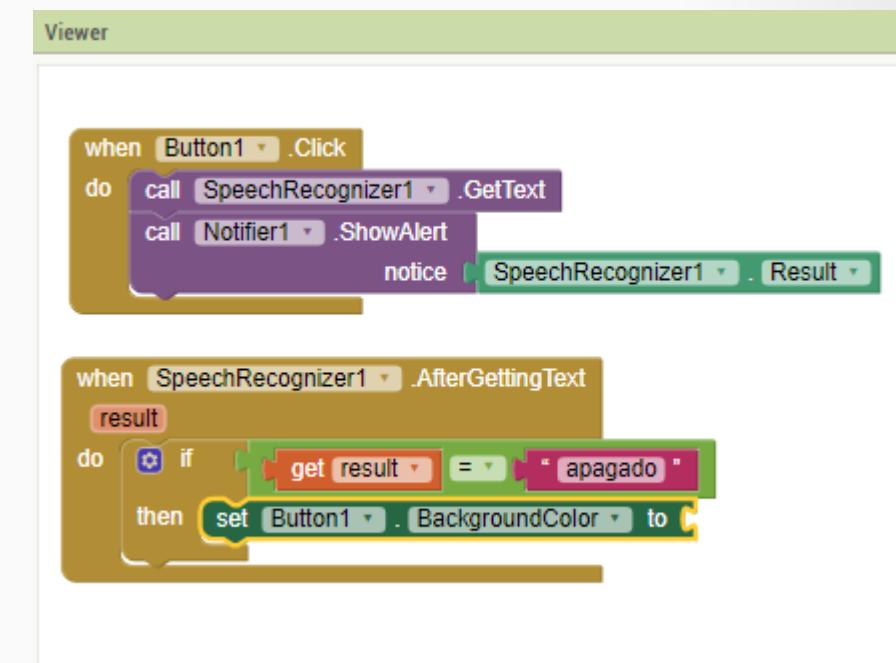
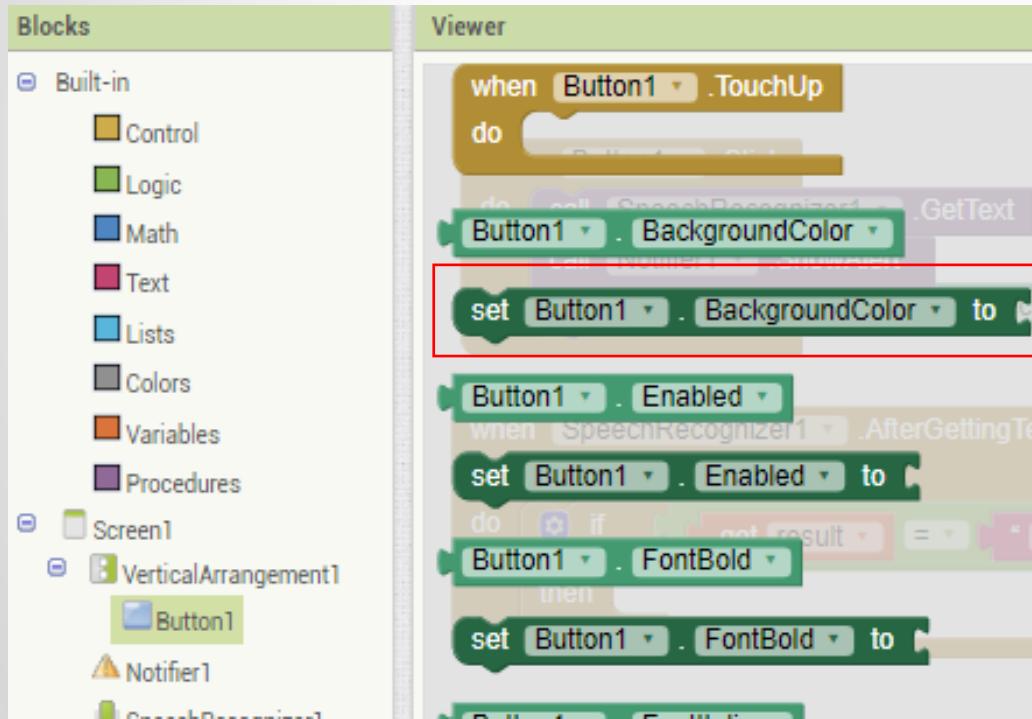
```
when Button1 Clicked
do
  call SpeechRecognizer1 .GetText
  call Notifier1 .ShowAlert
  notice SpeechRecognizer1 .Result
```
- Script 2 (Bottom):** Triggered by "when SpeechRecognizer1 AfterGettingText". It checks if the result is equal to "apagado". If true, it performs an action.

```
when SpeechRecognizer1 AfterGettingText
result
do
  if get result = "apagado" then
    [action]
```



¿Cómo lo realizamos?

17. Hacemos click sobre el botón de nuestra app, elegimos la opción cambiar color de fondo y lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

18. En colores, elegimos un color en nuestro ejemplo es negro y lo arrastramos a nuestro editor (ver gráficos).

The image shows the Scratch script editor with two scripts:

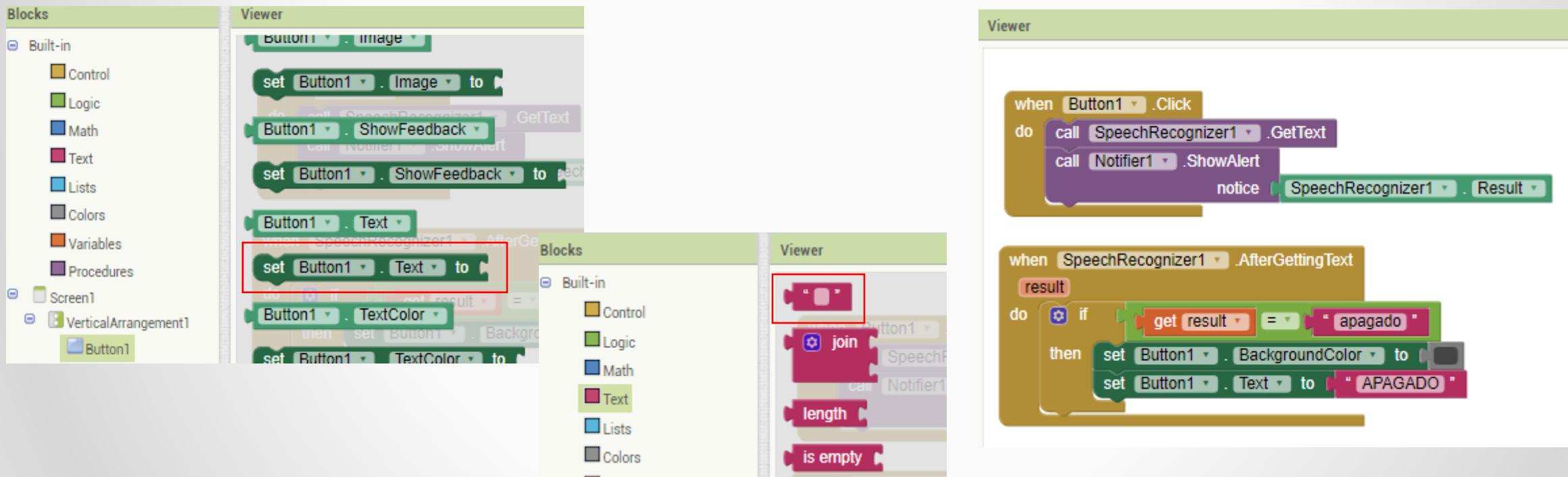
- Script 1 (when Button1.Click):** A yellow control script that calls the `SpeechRecognizer1.GetText` and `Notifier1.ShowAlert` blocks. It also receives a `notice` message from `SpeechRecognizer1.Result`.
- Script 2 (when SpeechRecognizer1.AfterGettingText):** A green control script that checks if the result is "apagado" (off). If true, it sets the background color of Button1 to black.

The **Blocks** palette on the left lists categories like Built-in, Colors, and Screen1. The **Viewer** palette shows the stage with a red button labeled "Button1". The bottom **Media** palette has options for "Upload File..." and "split color".



¿Cómo lo realizamos?

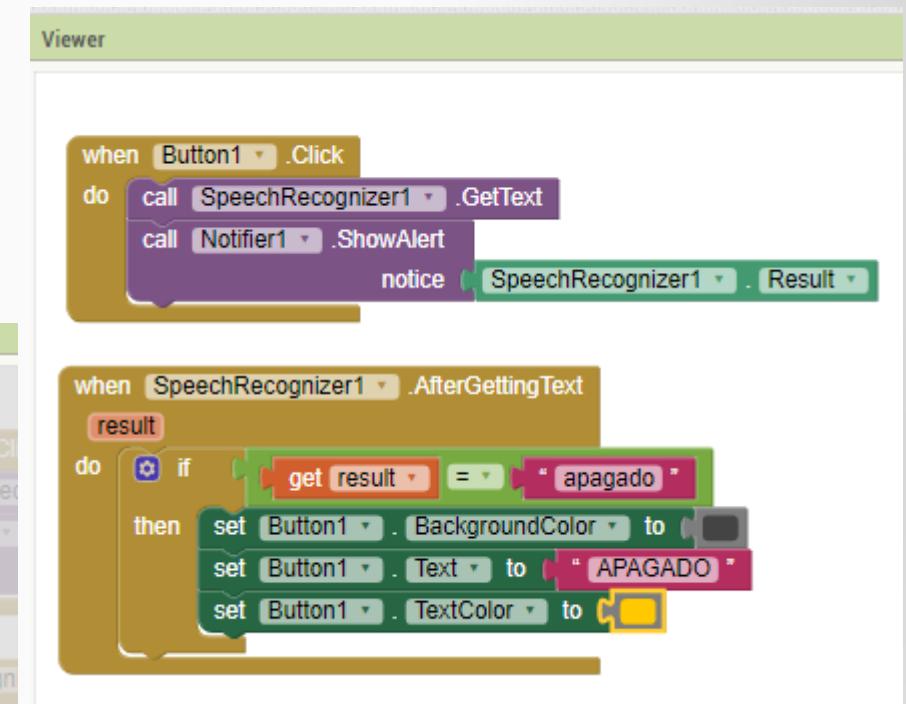
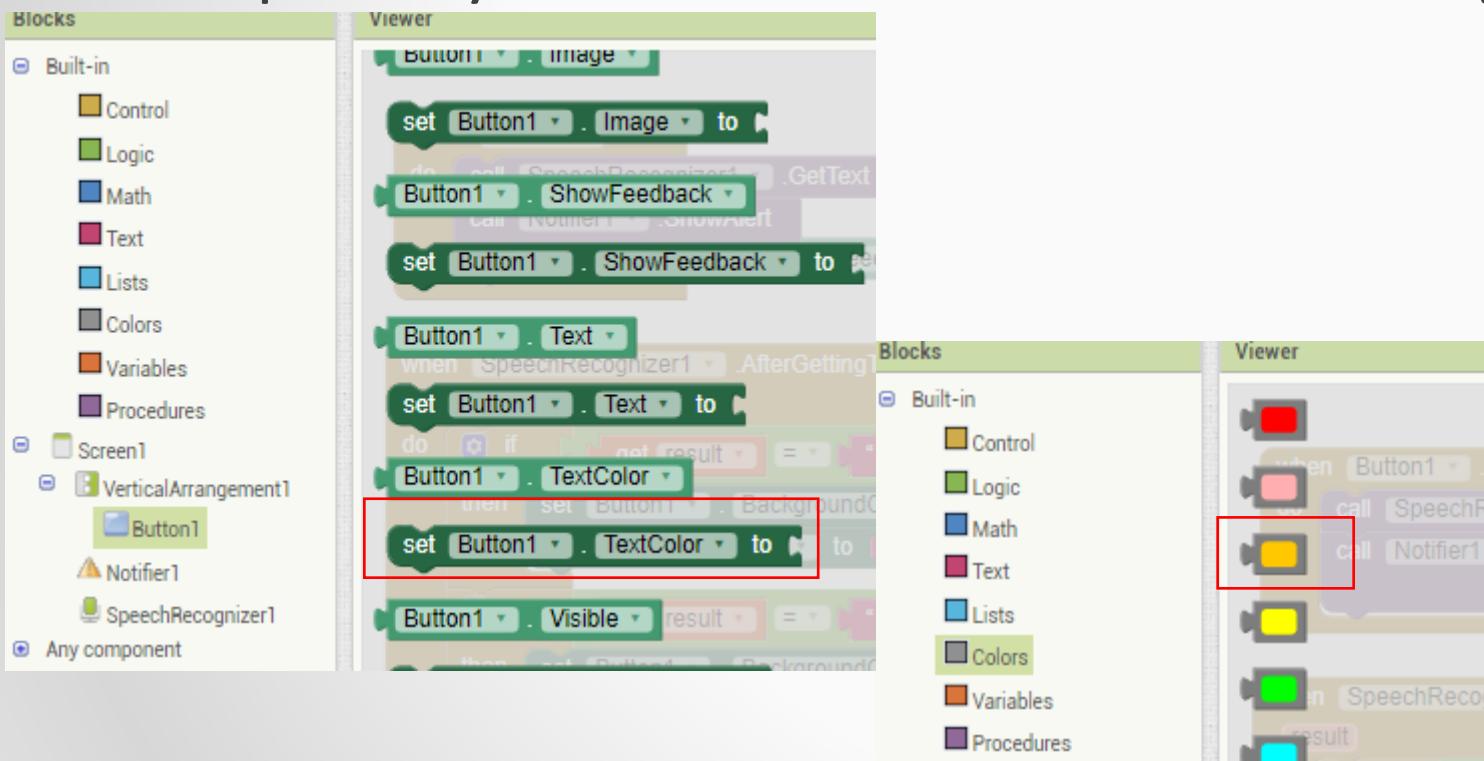
19. Hacemos click sobre el botón de nuestra app, elegimos la opción cambiar texto, luego vamos a la opción texto de nuestra paleta y lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

20. Hacemos click sobre el botón de nuestra app, elegimos la opción cambiar color del texto, luego vamos a la opción color de nuestra paleta y lo arrastramos a nuestro editor (ver gráficos).

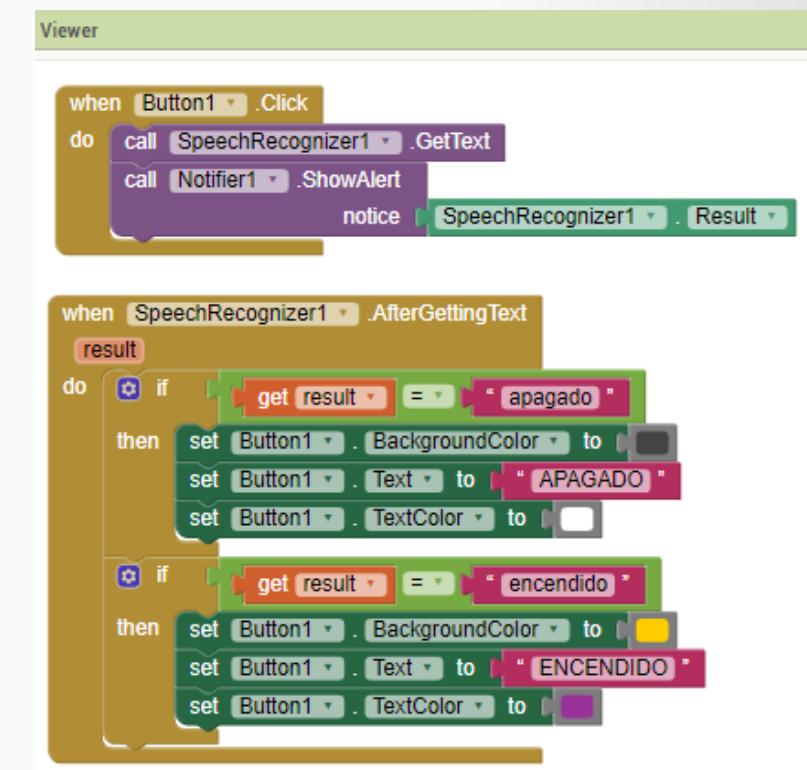
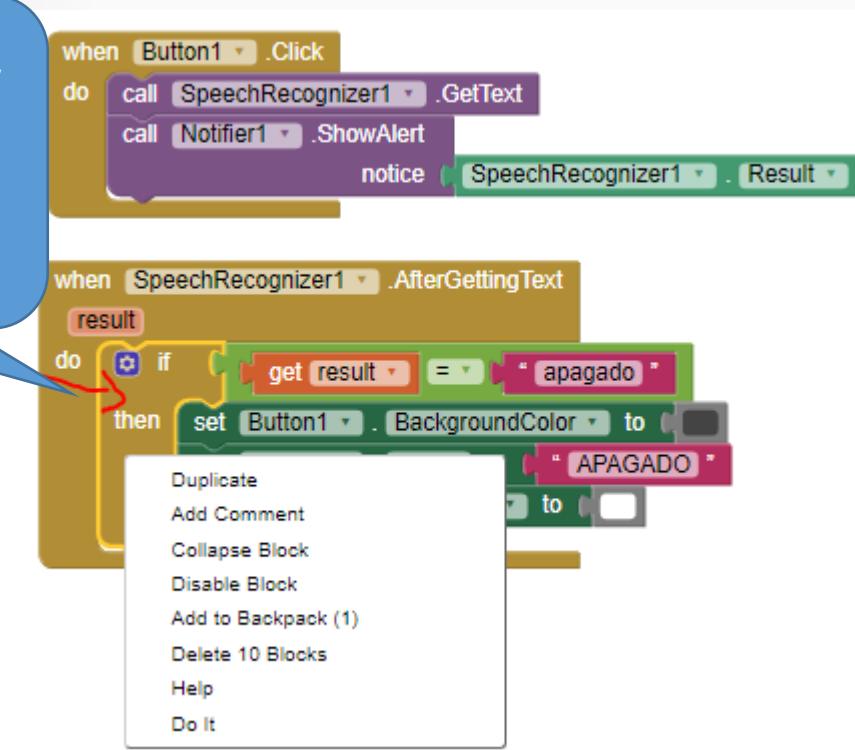




¿Cómo lo realizamos?

21. Repetimos los pasos 13 al 20 para hacer la secuencia de “encendido” o duplicamos el bloque if y cambiamos los valores(ver gráficos).

Hacer click derecho y aparecerá un menú con opciones, elegir la opción duplicar .



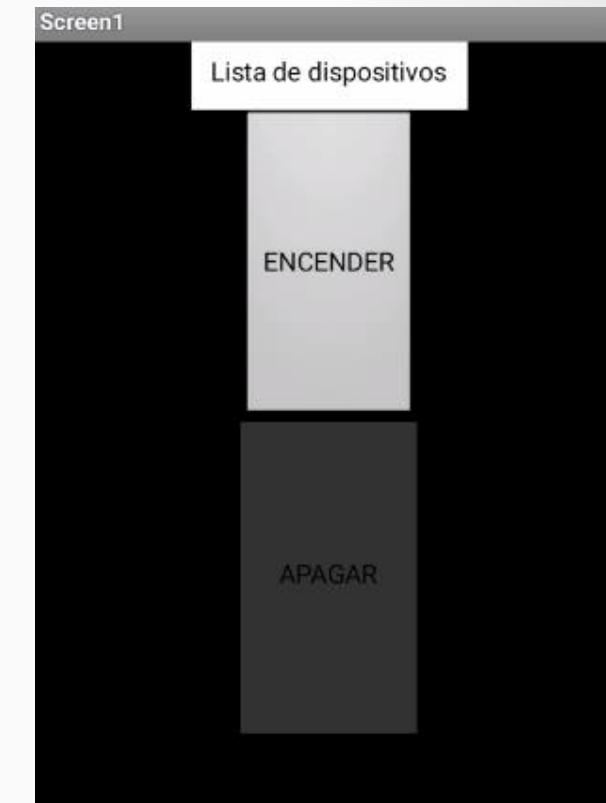
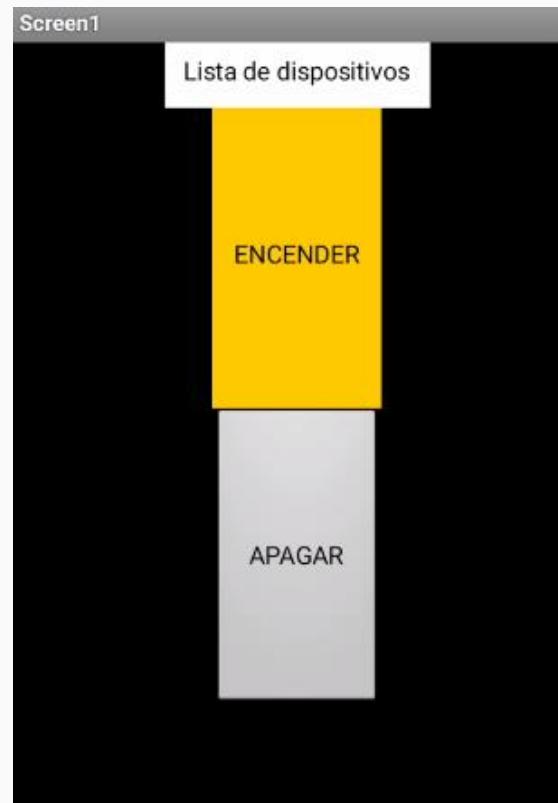
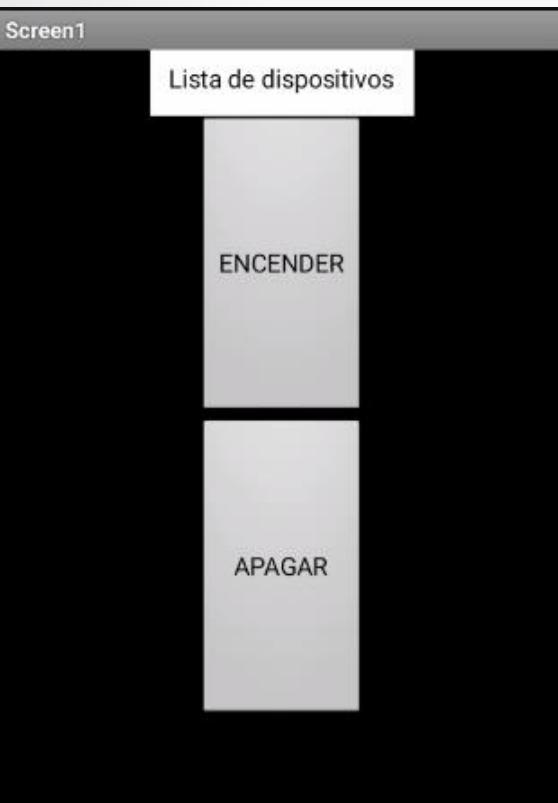
CONEXIÓN DE ARDUINO POR BLUETOOTH





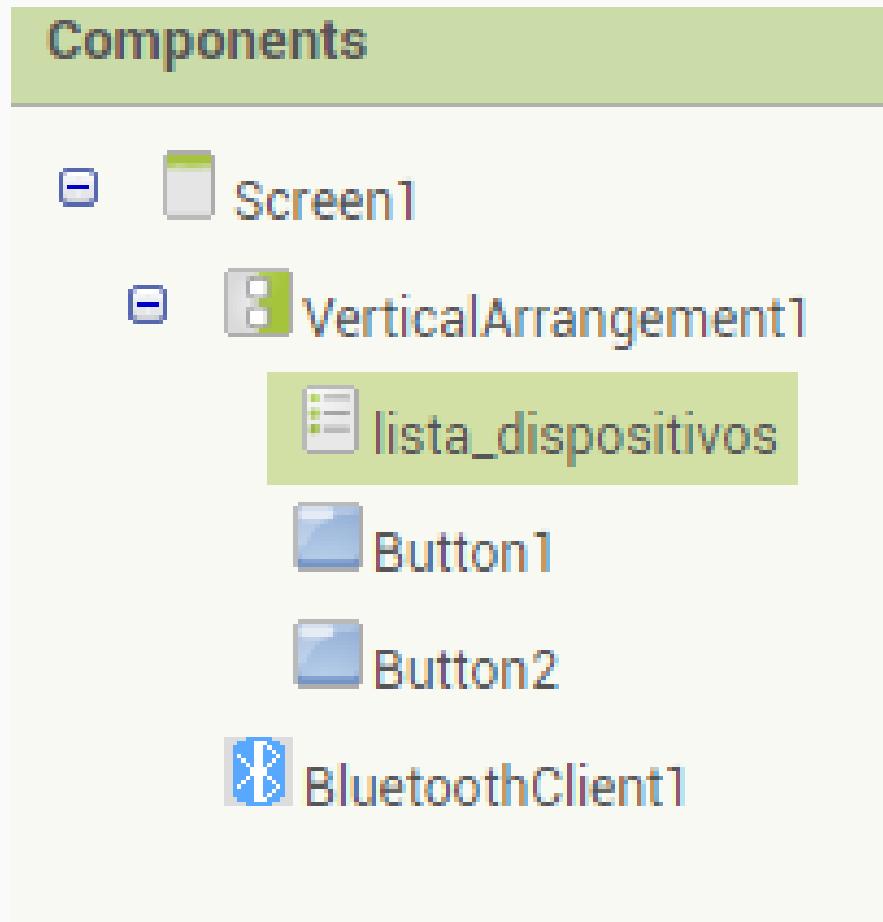
EJERCICIO DE APLICACIÓN

Realizar una aplicación que encienda y apague un led mediante bluetooth.





COMPONENTES UTILIZADOS

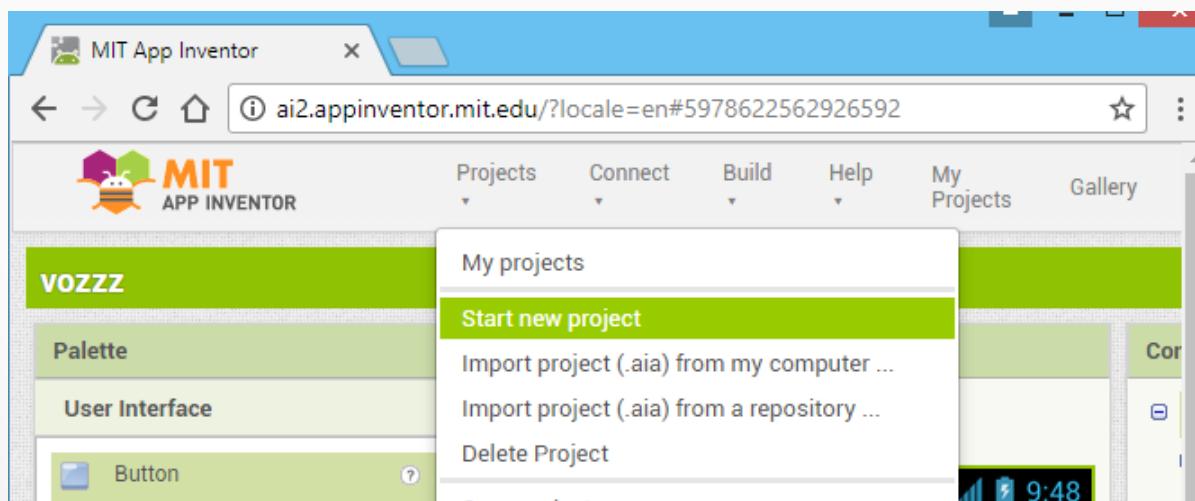


TUTORA: ANGELA JAZMIN MIRANDA FLORES



¿Cómo lo realizamos?

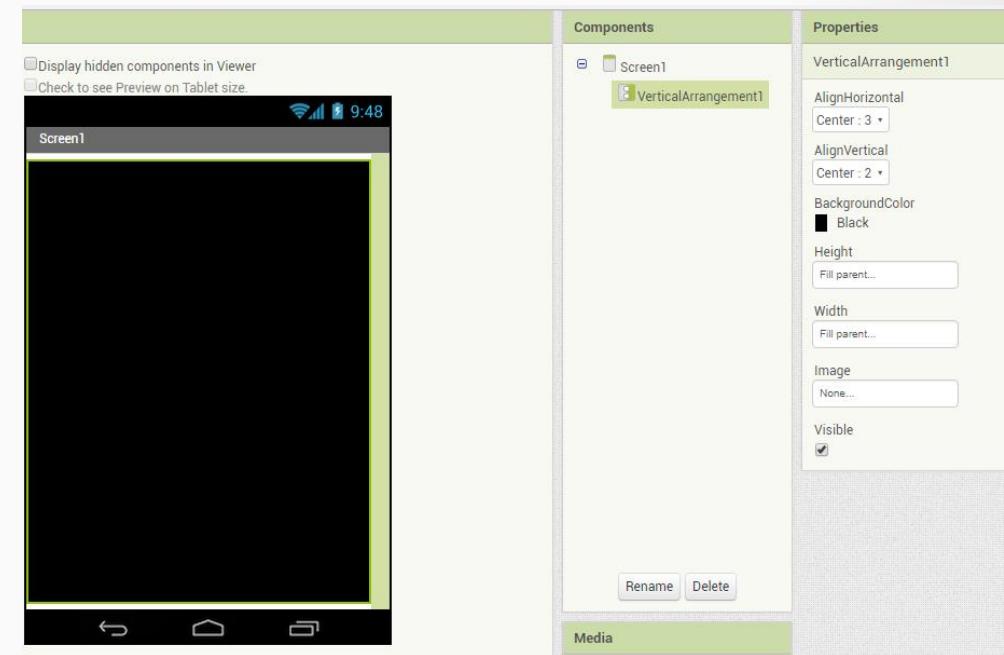
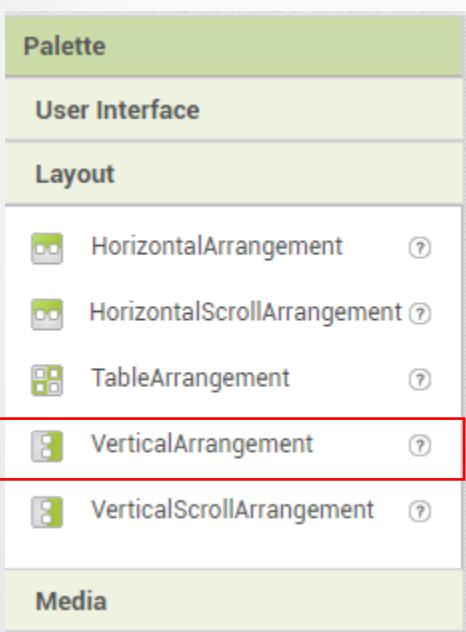
1. Creamos un nuevo proyecto dentro nuestra cuenta en app inventor.





¿Cómo lo realizamos?

2. En la paleta de app inventor buscamos la sección de Layout, de la cual utilizaremos el alineamiento vertical, lo arrastramos a nuestra app y le cambiamos las propiedades (ver gráficos).





¿Cómo lo realizamos?

3. En la sección interfaz de usuario arrastramos una lista de selección a nuestra app (ver gráficos).

The screenshot shows the MIT App Inventor interface with the following components:

- Palette (User Interface):** Shows various UI components: Button, CheckBox, DatePicker, Image, Label, ListPicker (highlighted with a red border), and ListView.
- Components View:** Displays the project structure:
 - Screen1
 - VerticalArrangement1
 - Lista_dispositivos
- Properties View:** Shows the properties for the "Lista_dispositivos" component:
 - ItemBackgroundColor: Black
 - ItemTextColor: White
 - Selection: (empty)
 - Shape: default
 - ShowFeedback: checked
 - ShowFilterBar: unchecked
 - Text: Lista de dispositivos
 - TextAlignment: center : 1
 - TextColor: Default
 - Title: (empty)
 - Visible: checked



¿Cómo lo realizamos?

3. Ahora en la sección interfaz de usuario arrastramos dos botones a nuestra app y configuraremos sus propiedades (ver gráficos).

The screenshot displays a user interface builder for a mobile application. On the left, a palette lists components: Button (selected and highlighted with a red border), CheckBox, DatePicker, Image, and Label. The main area shows a smartphone screen with a black background. At the top, there is a white bar labeled "Lista de dispositivos". In the center, there is a vertical arrangement of two buttons. The top button is labeled "ENCENDER" and the bottom button is labeled "APAGAR". To the right, there are two panels: "Components" and "Properties".

Components Panel:

- Screen1
- VerticalArrangement1
 - Lista_dispositivos
 - Button1
 - Button2

Properties Panel - Button1:

- BackgroundColor: Default
- Enabled: checked
- FontBold: checked
- FontSize: 14.0
- FontTypeface: default
- Height: 40 percent
- Width: 30 percent
- Image: None
- Shape: default
- ShowFeedback: checked
- Text: ENCENDER
- TextAlignment: center
- TextColor: Default
- Visible: checked

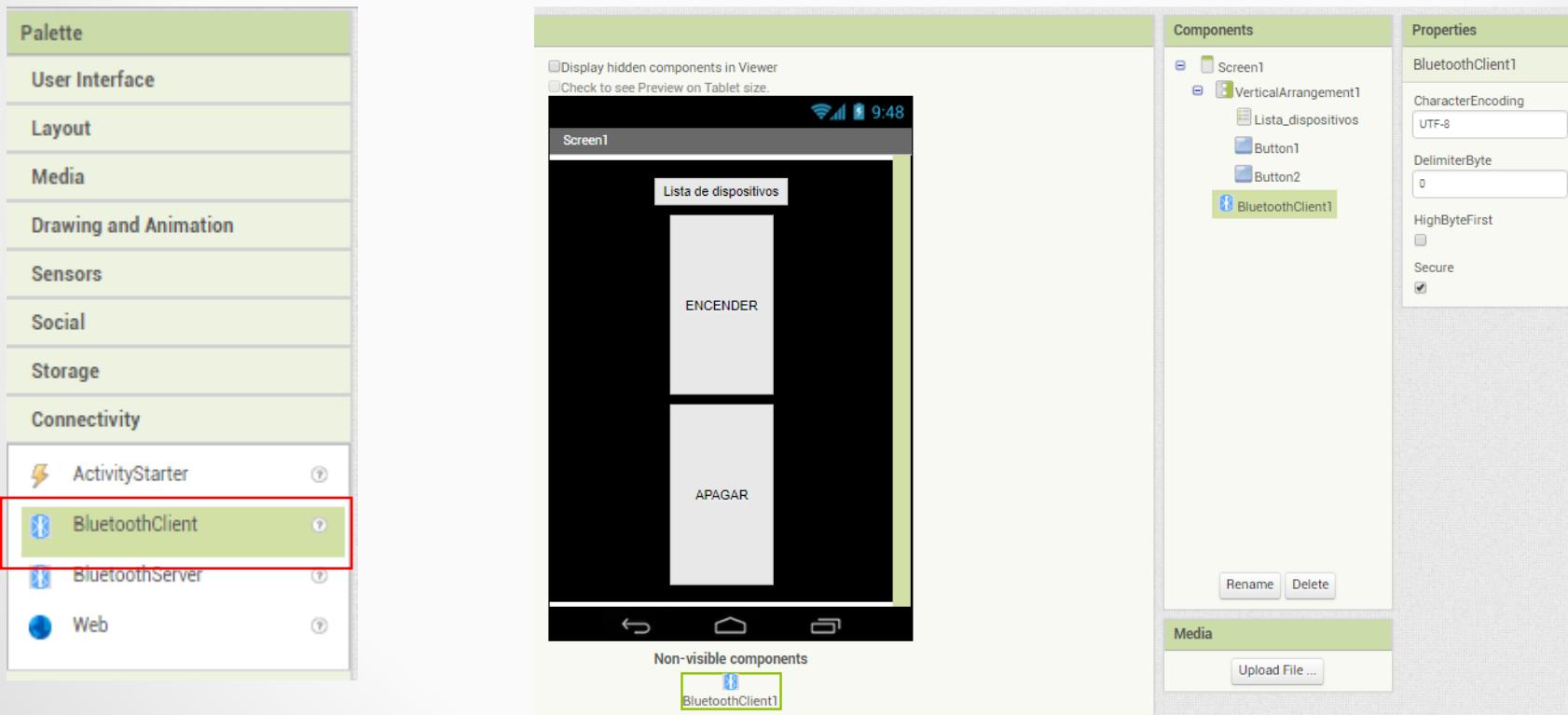
Properties Panel - Button2:

- BackgroundColor: Default
- Enabled: checked
- FontBold: checked
- FontSize: 14.0
- FontTypeface: default
- Height: 40 percent
- Width: 30 percent
- Image: None
- Shape: default
- ShowFeedback: checked
- Text: APAGAR
- TextAlignment: center
- TextColor: Default
- Visible: checked



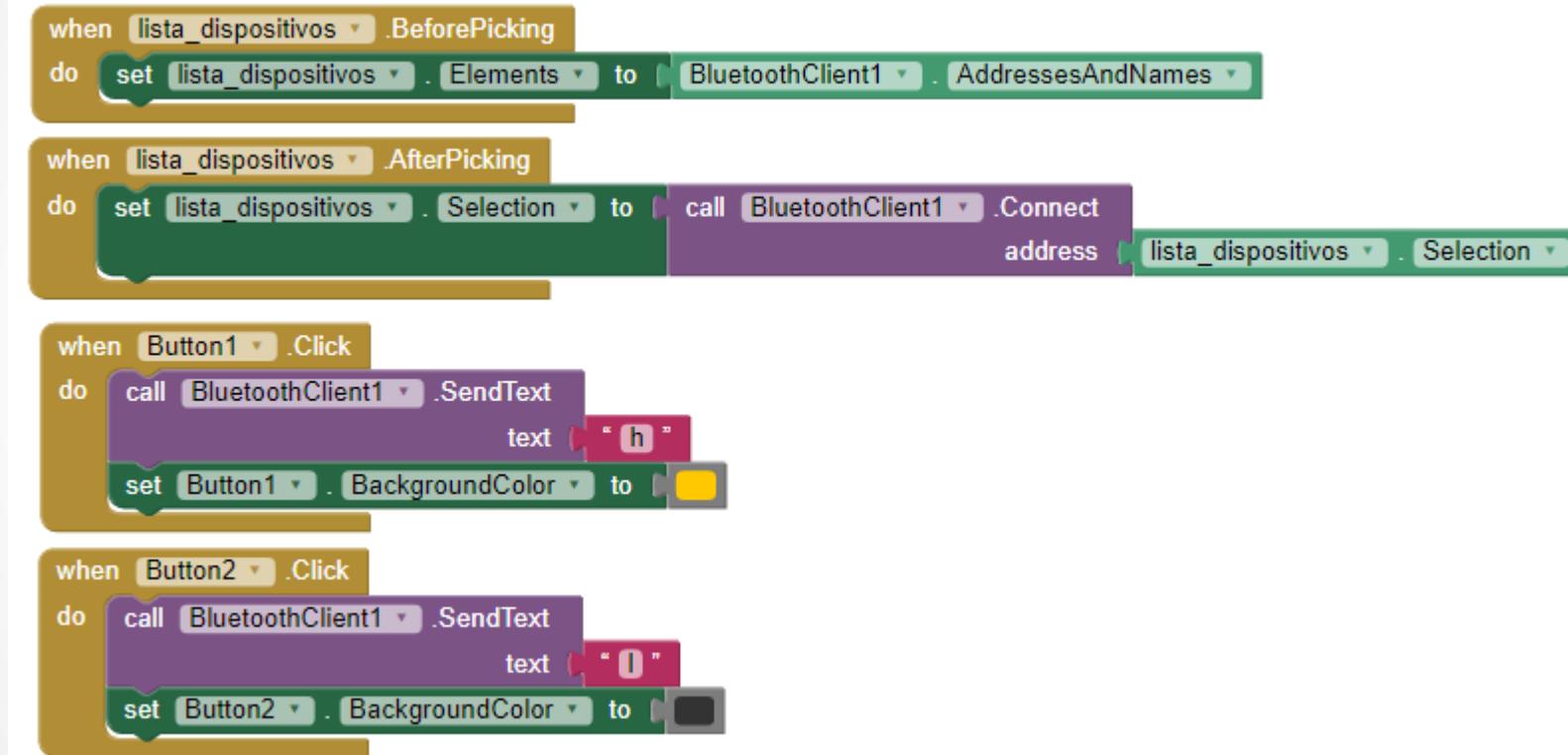
¿Cómo lo realizamos?

4. En la paleta buscamos la sección de conectividad y arrastramos cliente bluetooth a nuestra app (ver gráficos).





CÓDIGO POR BLOQUES EN APP INVENTOR

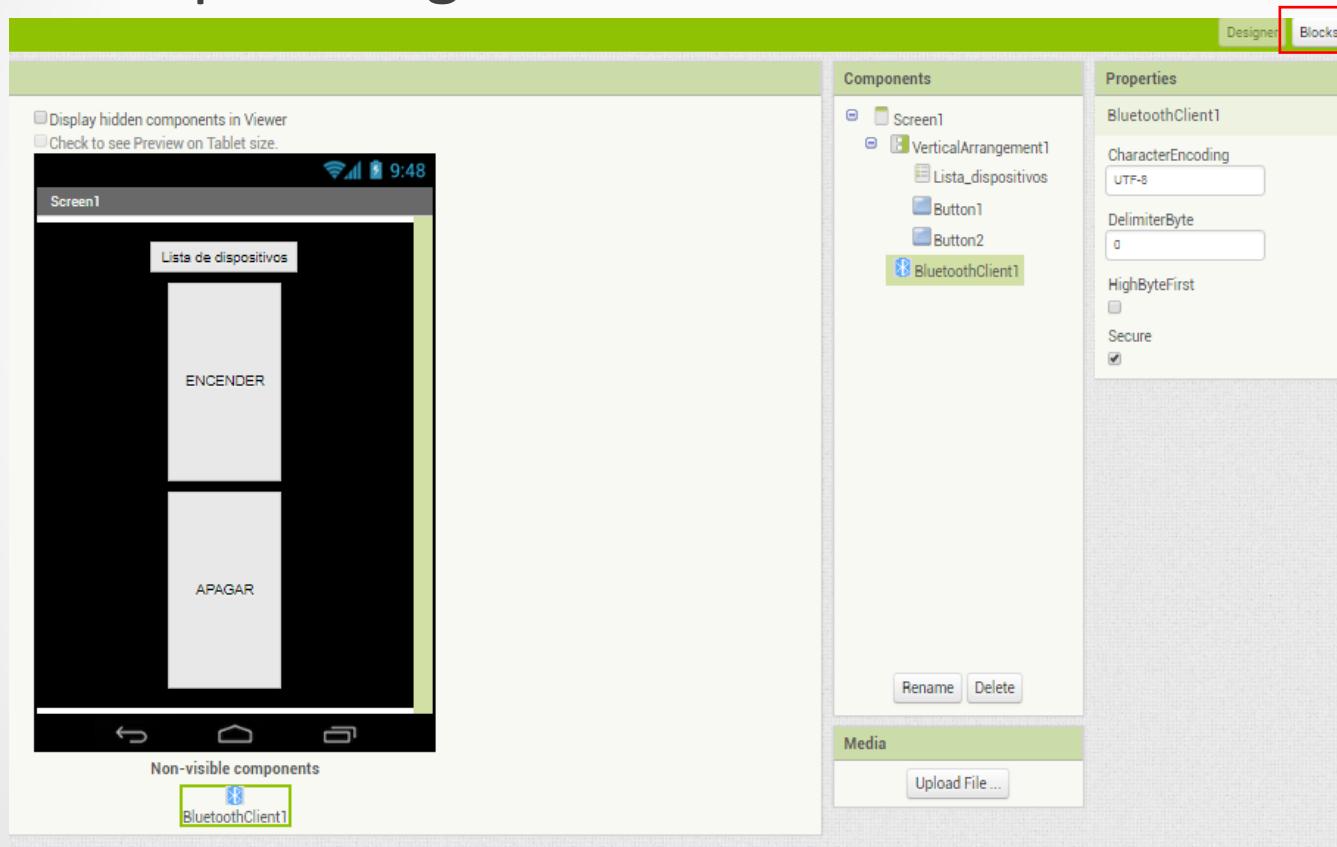


TUTORA: ANGELA JAZMIN MIRANDA FLORES



¿Cómo lo realizamos?

5. Vamos a bloques luego de terminar el diseño





¿Cómo lo realizamos?

5. Vamos a la lista de selección y arrastramos al editor la opción antes de seleccionar (ver gráficos).

The screenshot shows the MIT App Inventor interface. On the left, the 'Blocks' palette is open, displaying categories like Built-in, Screen1, and Any component. In the center, the 'Viewer' workspace contains several event blocks for a component named 'Lista_dispositivos'. One specific block, 'when Lista_dispositivos .BeforePicking do []', is highlighted with a red border.

Blocks palette:

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - VerticalArrangement1
 - Lista_dispositivos
 - Button1
 - Button2
- Any component

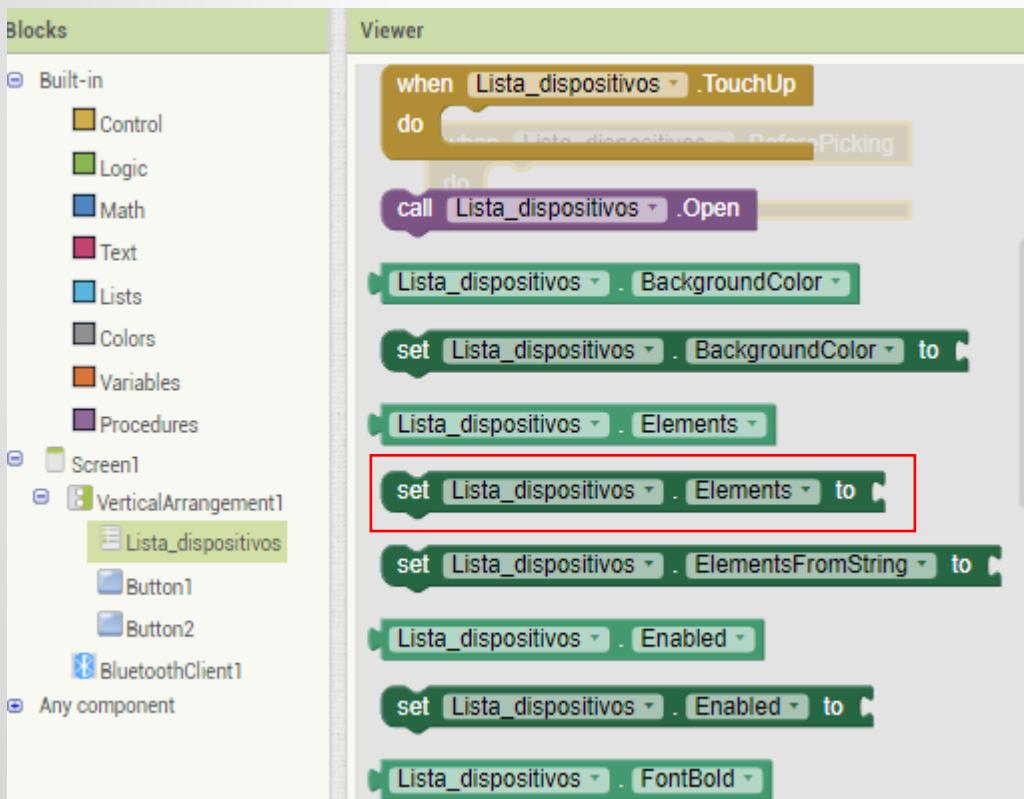
Viewer workspace:

```
when Lista_dispositivos .BeforePicking
do [ ]
```



¿Cómo lo realizamos?

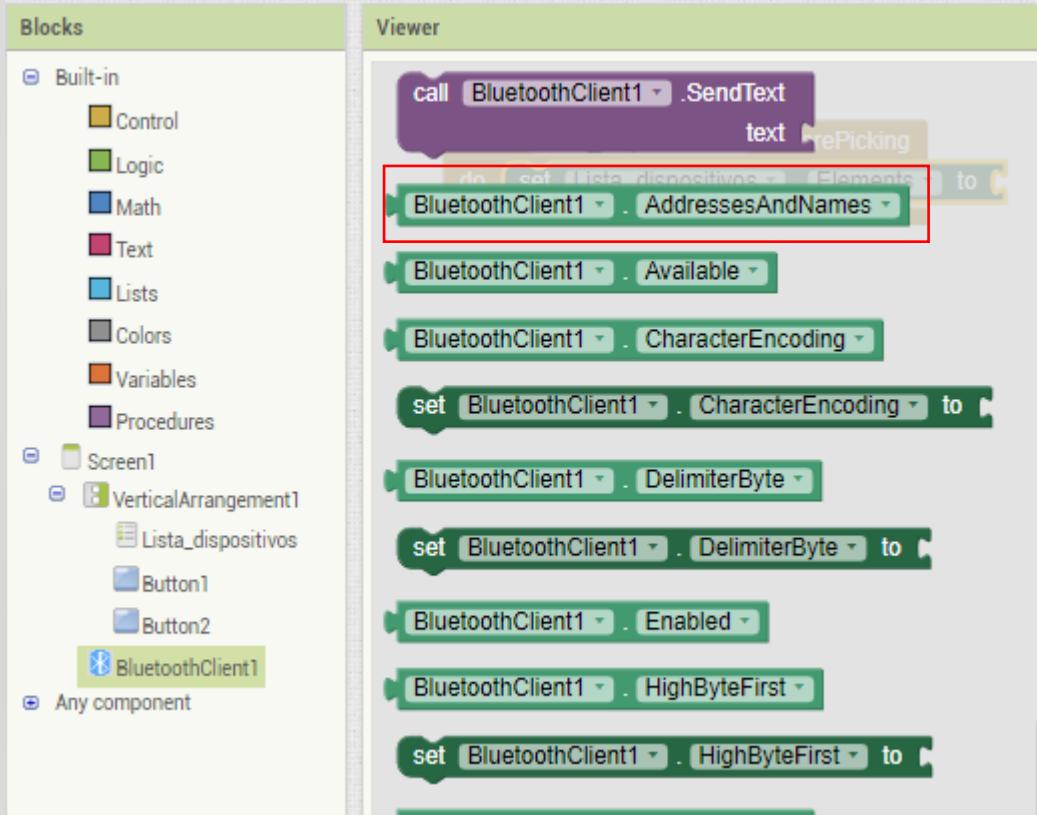
6. Vamos a la lista de selección y arrastramos al editor la opción cambiar los elementos con (ver gráficos).





¿Cómo lo realizamos?

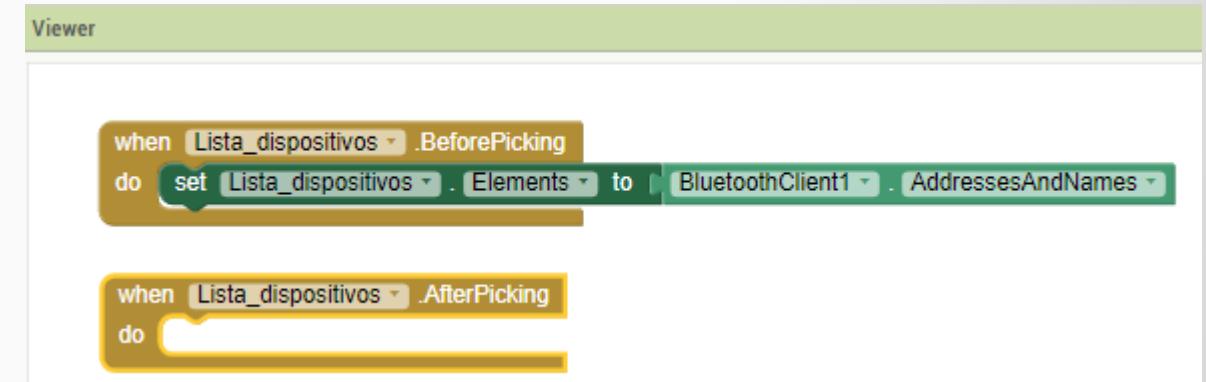
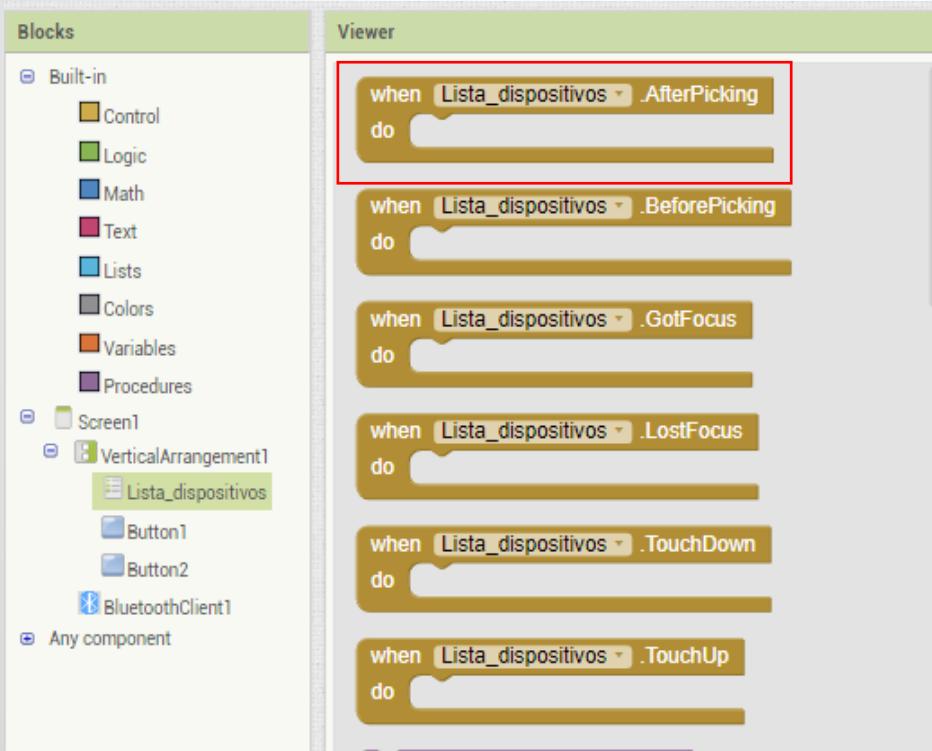
7. Vamos a cliente bluetooth y arrastramos al editor la opción dirección y nombres (ver gráficos).





¿Cómo lo realizamos?

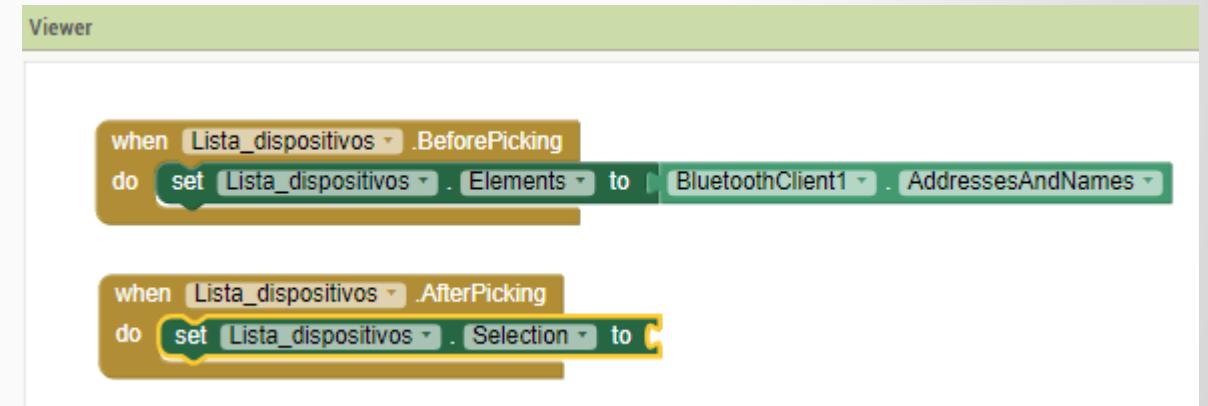
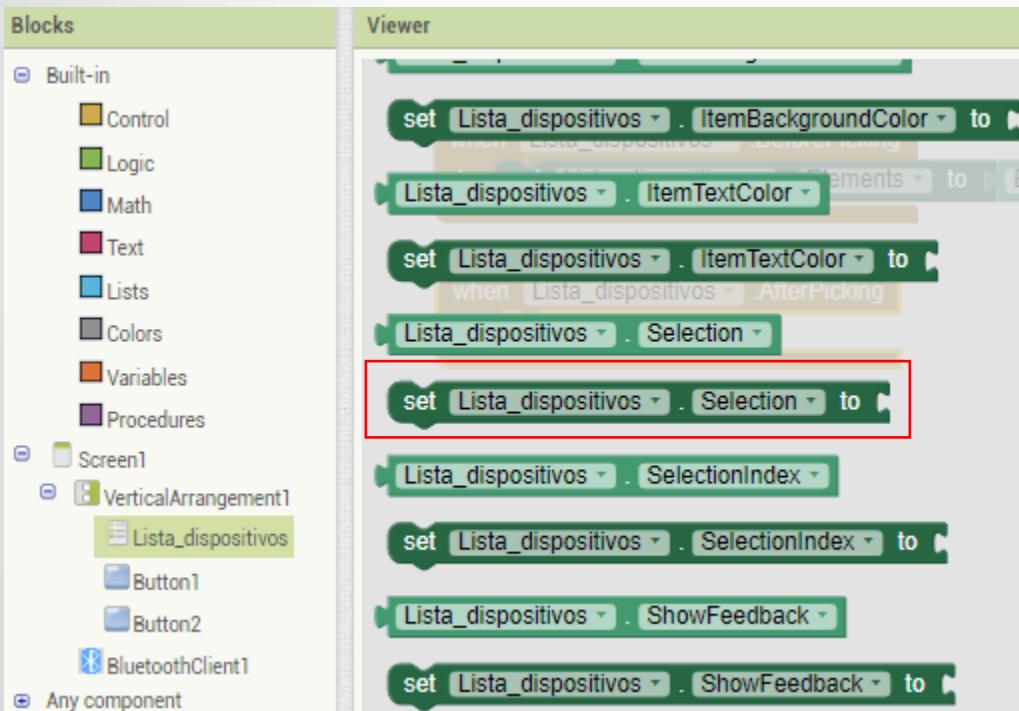
8. Vamos a la lista de selección y arrastramos al editor la opción después de seleccionar (ver gráficos).





¿Cómo lo realizamos?

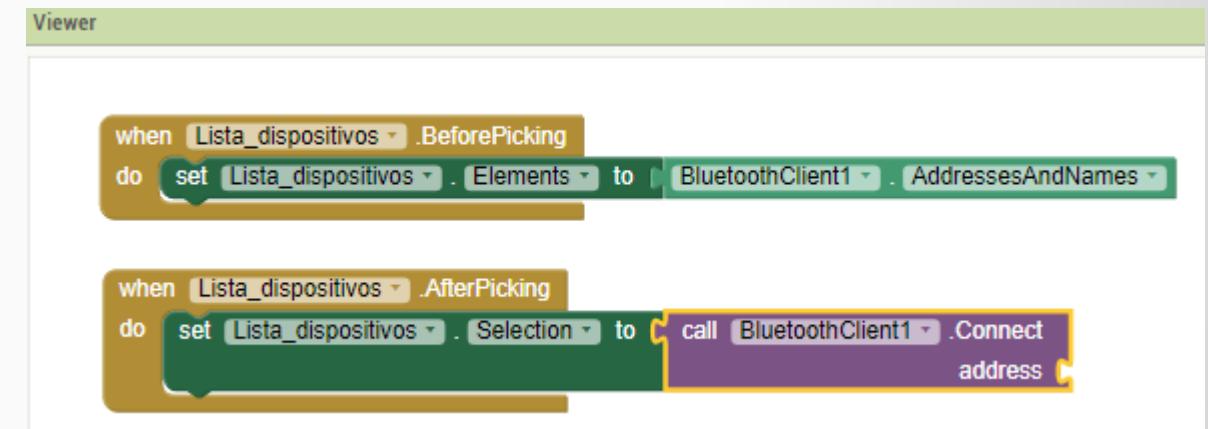
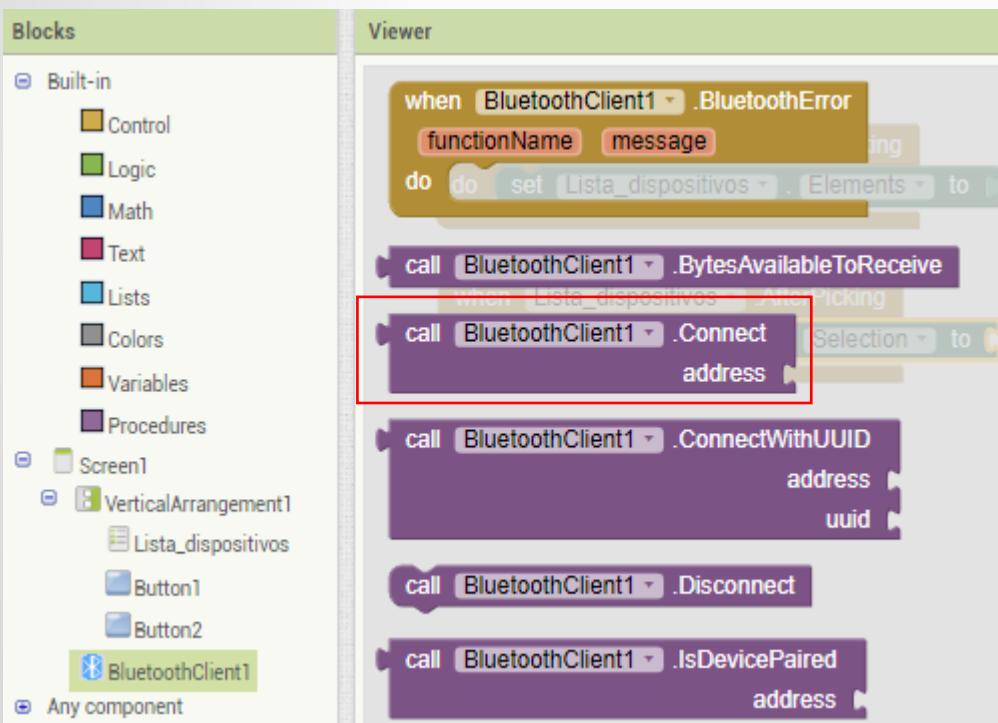
9. Vamos a la lista de selección y arrastramos al editor la opción cambiar lista con el elemento seleccionado (ver gráficos).





¿Cómo lo realizamos?

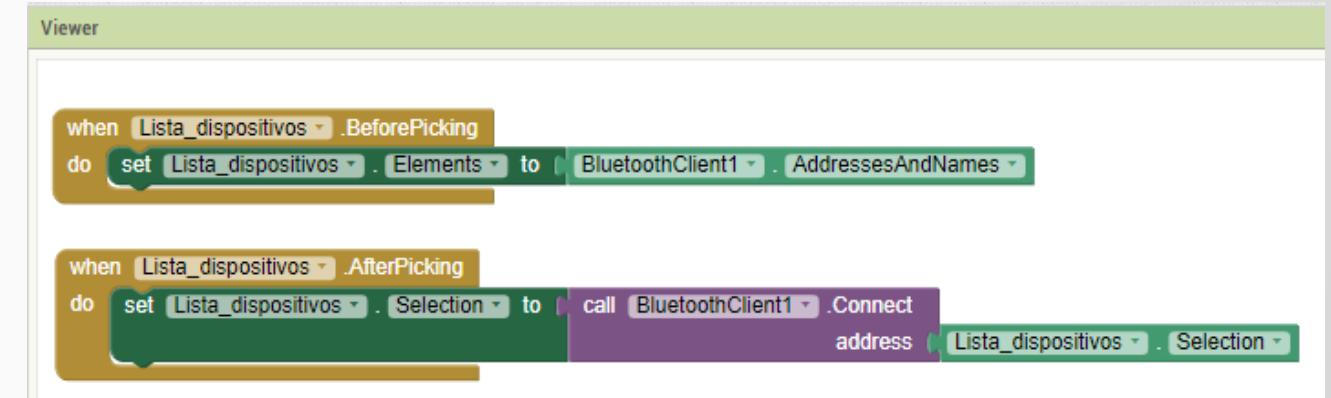
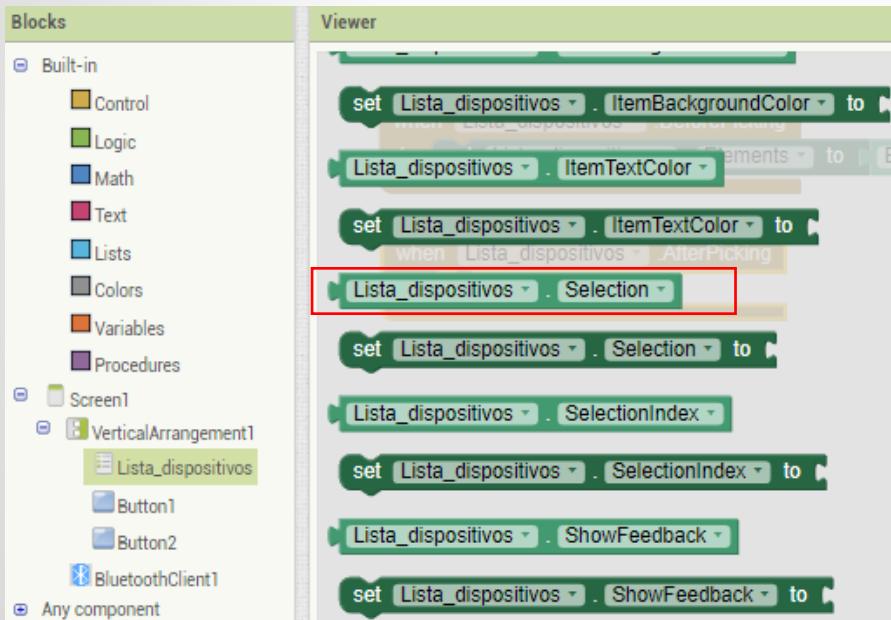
10. Vamos a cliente bluetooth y arrastramos al editor la opción conectar dirección (ver gráficos).





¿Cómo lo realizamos?

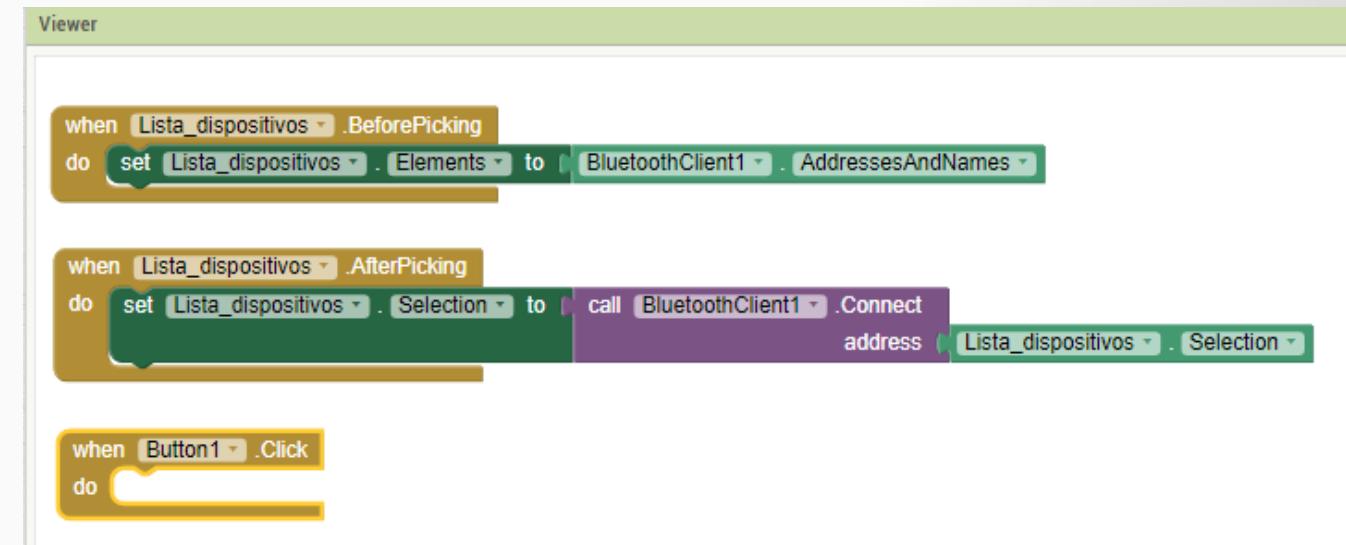
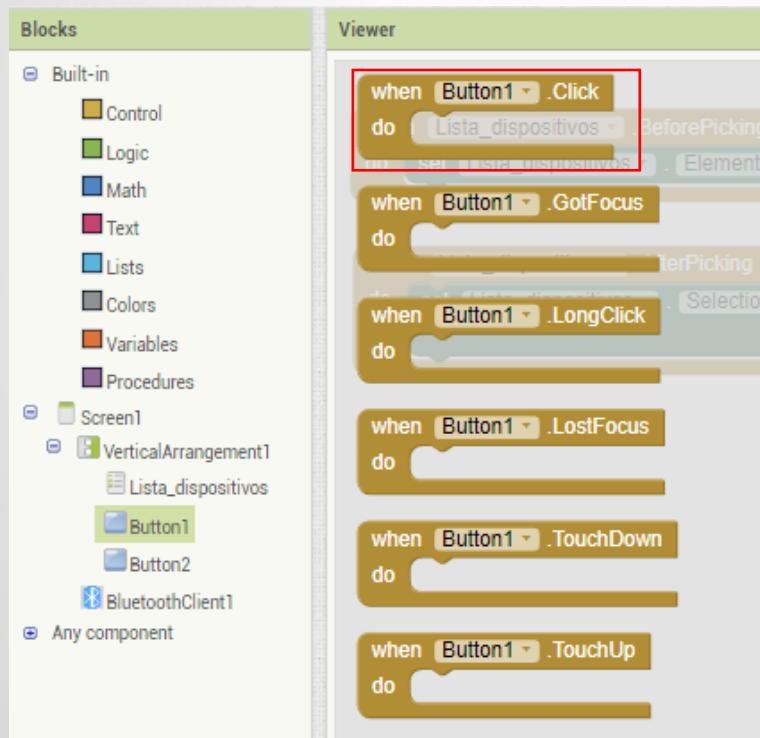
11. Vamos a la lista de selección y arrastramos al editor la opción elemento seleccionado (ver gráficos).





¿Cómo lo realizamos?

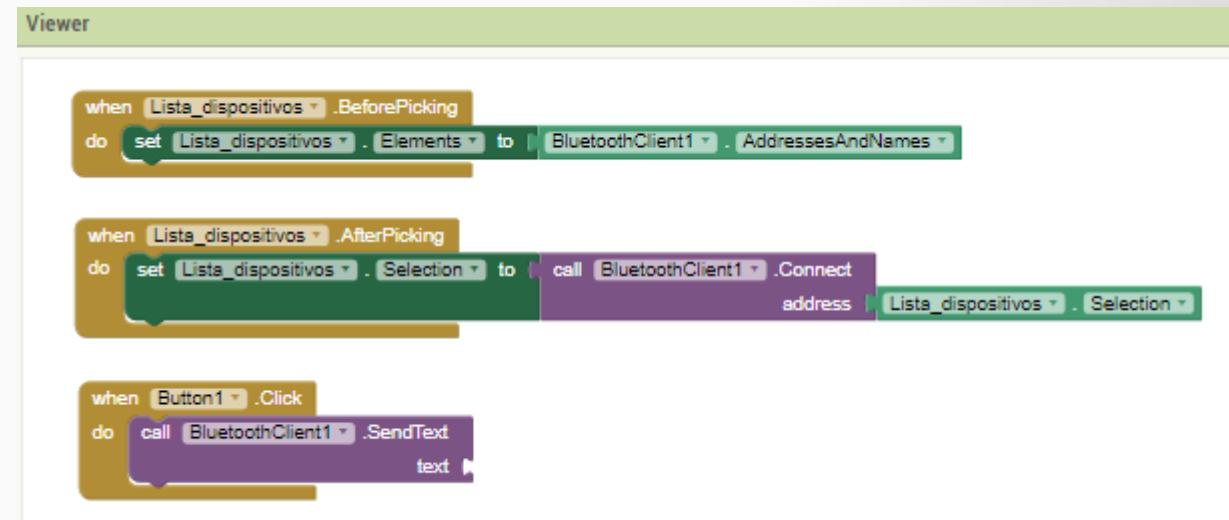
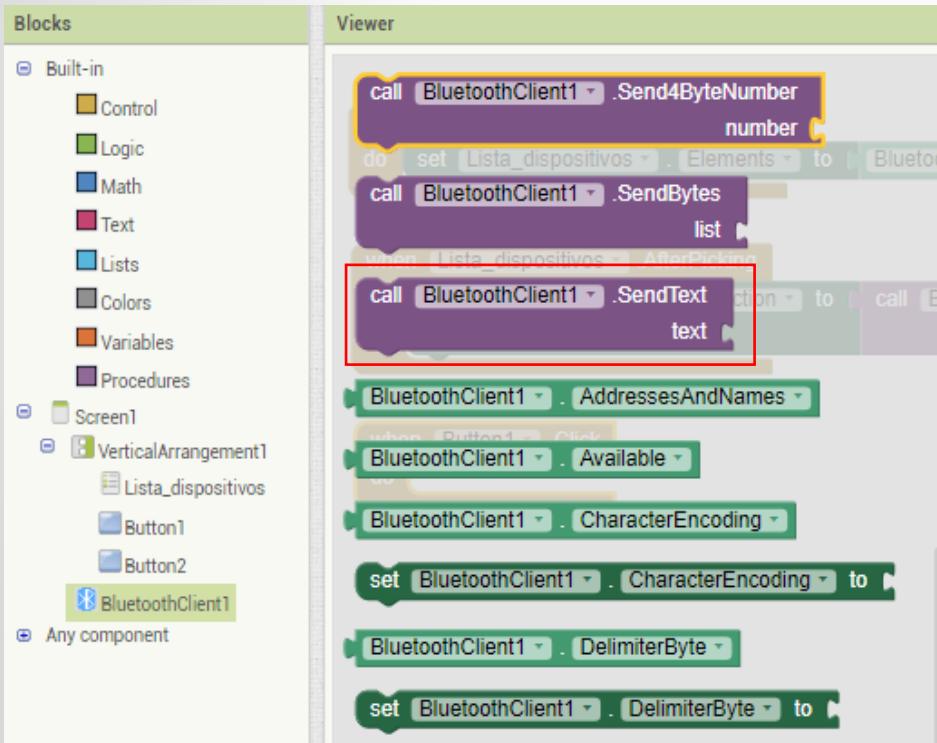
12. Hacemos click sobre el botón de nuestra app, elegimos la opción cuando se presione y lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

13. Hacemos click sobre cliente bluetooth, elegimos la opción mandar texto y lo arrastramos a nuestro editor (ver gráficos).





¿Cómo lo realizamos?

14. Hacemos click sobre el botón de nuestra app, elegimos la opción cuando se presione y lo arrastramos a nuestro editor (ver gráficos).

The image shows the Scratch script editor with three scripts:

- BeforePicking:** A control script that sets the list `Lista_dispositivos` to the elements of `BluetoothClient1`'s `AddressesAndNames`.
- AfterPicking:** A control script that sets the list `Lista_dispositivos` to the selection of `BluetoothClient1`'s `Connect` method, passing the address from the selection.
- Button1 Click:** A control script that calls `BluetoothClient1`'s `SendText` method with the text "h".



¿Cómo lo realizamos?

15. Hacemos click sobre el botón de nuestra app, elegimos la opción cambiar fondo y lo arrastramos a nuestro editor (ver gráficos).

The image shows the Scratch script editor with two scripts for the 'Button1' hat:

Script 1 (when Button1 .TouchDown):

- when Button1 .TouchDown
- do
- set Lista_dispositivos . Elements to

Script 2 (when Button1 .TouchUp):

- when Button1 .TouchUp
- do
- set Lista_dispositivos . AfterPicking to
- Button1 . BackgroundColor to [blue v]

Script 3 (when Button1 .Click):

- when Button1 .Click
- do
- call BluetoothClient1 .Connect [address] [Lista_dispositivos . Selection]
- call BluetoothClient1 .SendText [text] ["h"]
- set Button1 . BackgroundColor to [yellow v]



¿Cómo lo realizamos?

16. Hacemos click colores, elegimos un color en nuestro ejemplo es amarillo y lo arrastramos a nuestro editor (ver gráficos).

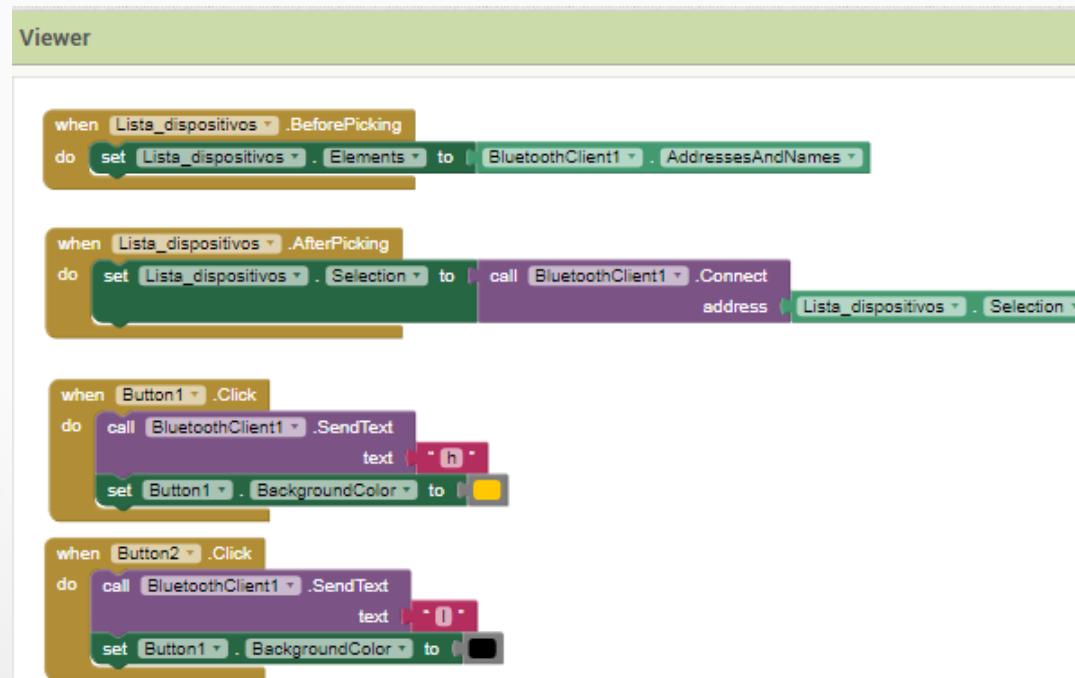
The image shows the Scratch script editor interface. On the left is the 'Blocks' palette with categories like Built-in, Colors (highlighted), and Screen1. On the right is the 'Viewer' where three scripts are visible:

- when [Lista_dispositivos v] .BeforePicking**: A control script that sets the list variable to a list of Bluetooth addresses.
- when [Lista_dispositivos v] .AfterPicking**: A control script that connects to the selected device.
- when [Button1 v] .Click**: A control script that sends the letter 'h' via Bluetooth and changes the button's background color to yellow.



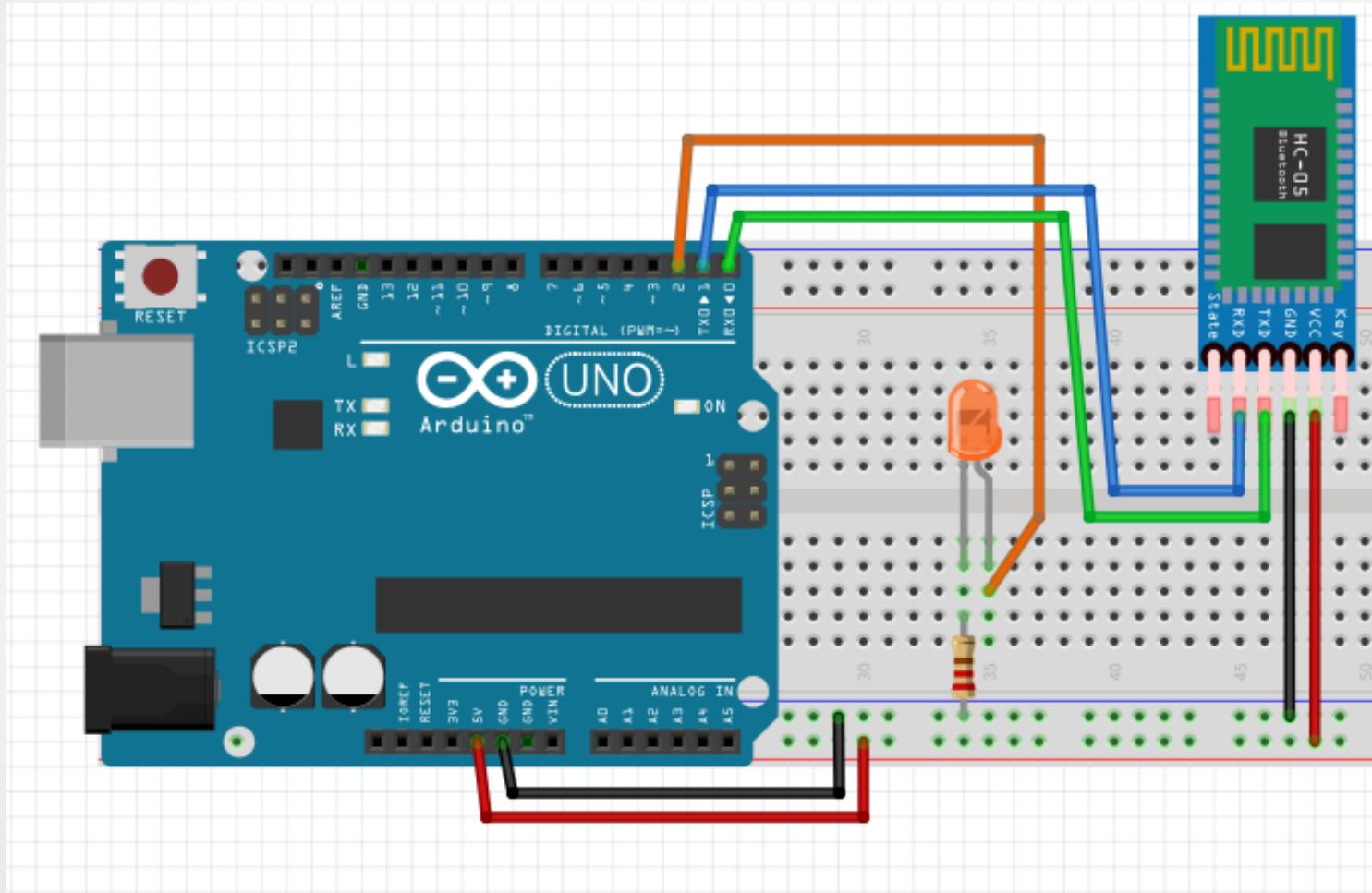
¿Cómo lo realizamos?

17. Repetimos los pasos 12 al 16 para hacer el código del segundo botón o duplicamos el bloque del botón y cambiamos los valores(ver gráfico).





Conexión del bluetooth



TUTORA: ANGELA JAZMIN MIRANDA FLORES



Código en arduino

```
//AUTORA:Angela Miranda Flores
//CONTACTO angiejazminmiranda@gmail.com
char estado=' ';
void setup() {
    Serial.begin(9600);
    pinMode(2,OUTPUT);
}
void loop() {
    if(Serial.available()>0){
        estado = Serial.read();
    }
    if(estado== 'h' ){
        digitalWrite(2,HIGH);
    }
    if(estado== 'l' ){
        digitalWrite(2,LOW);
    }
}
```

TUTORA: ANGELA JAZMIN MIRANDA FLORES

CONECTANDO LA APLICACIÓN POR BLUETOOTH



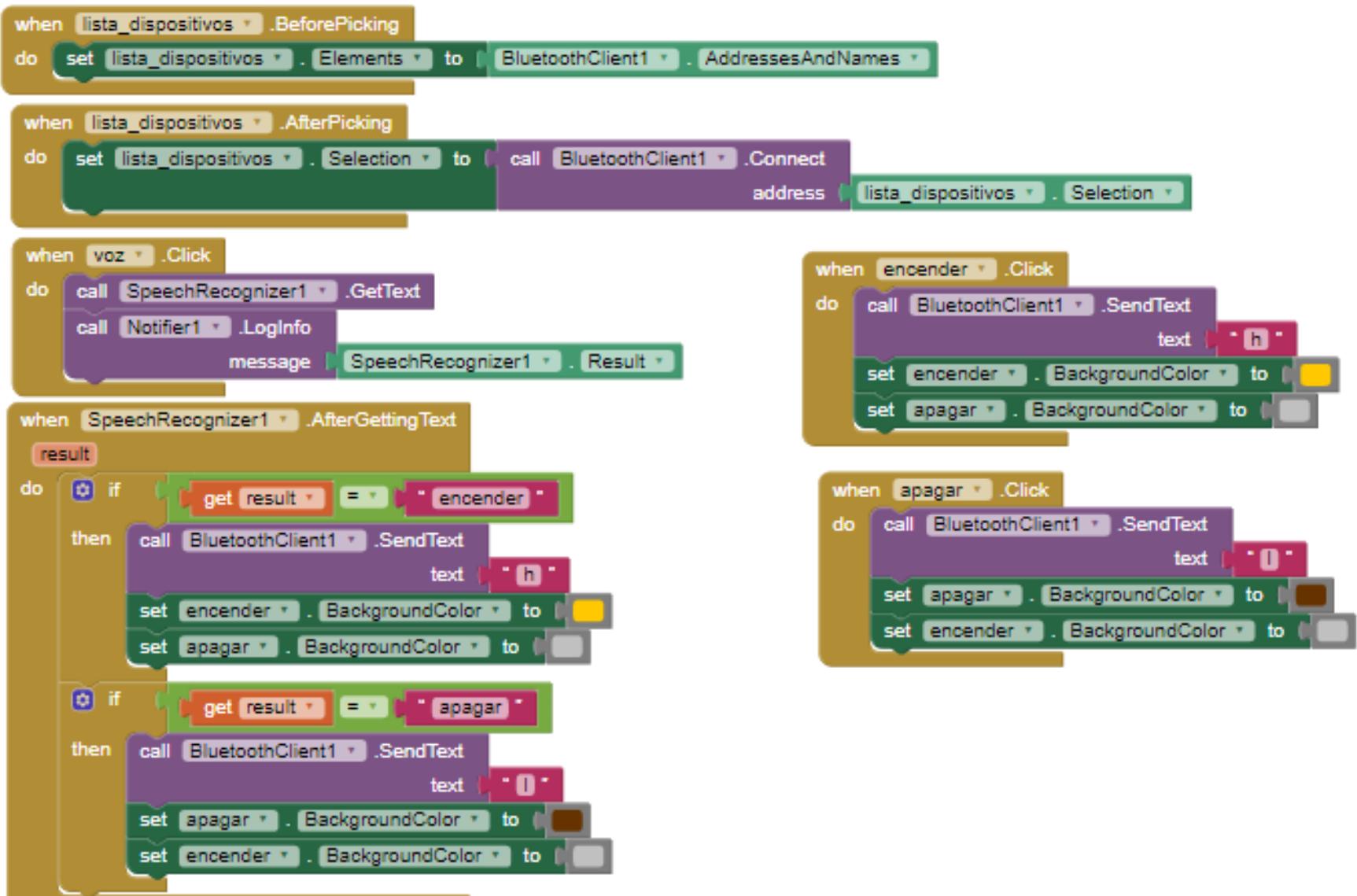
EJERCICIO DE APLICACIÓN

Con lo aprendido en clase, realizar una aplicación que conecte el reconocimiento de voz mediante bluetooth a arduino.

La aplicación debe tener una lista de selección y 3 botones, uno será para el reconocimiento de voz, el segundo botón para encender y el tercer botón para apagar un led (La aplicación debe quedar como en la imagen); el código para arduino es el mismo del anterior ejercicio de aplicación.



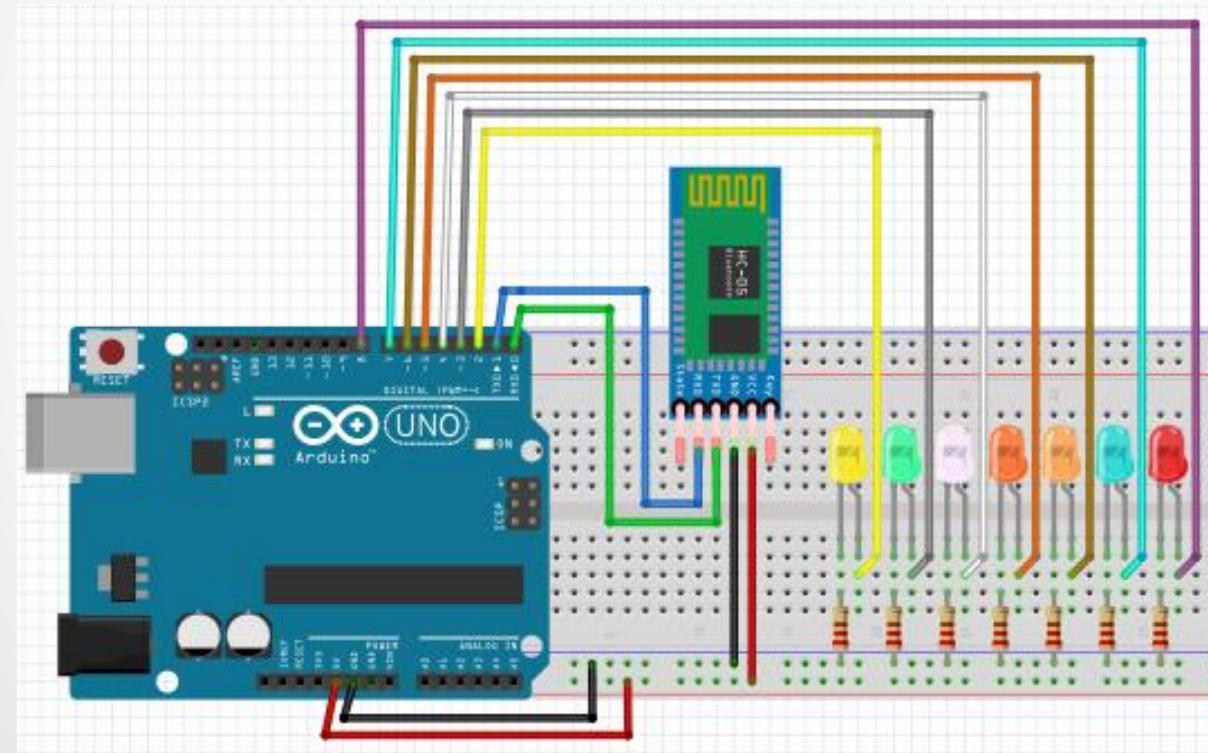
CÓDIGO POR BLOQUES EN APP INVENTOR



TUTORA: ANGELA JAZMIN MIRANDA FLORES

EJERCICIO DE APLICACIÓN

CREAR UNA APLICACIÓN EN APP INVENTOR QUE ENCIENDA Y APAGUE 7 LEDS SOLO CON RECONOCIMIENTO DE VOZ.



TUTORA: ANGELA JAZMIN MIRANDA FLORES