Proyecto 2:

Controla tu casa domótica Placa Zum Core 2.0

Descripción del proyecto:

Con esta guía aprenderás a crear una aplicación para tu tablet o móvil que controle la casa domótica que has construido.

Nivel de dificultad: *Medio*

Tiempo estimado: *4 horas*

Materiales:

- Casa domótica
- Bitbloq
- App Inventor
- Dispositivo Android (tablet o móvil)

¿Qué es App Inventor?

App Inventor es un entorno de desarrollo de software online que permite crear aplicaciones para Android. El usuario puede, de forma visual y a partir de un conjunto de herramientas básicas, ir enlazando una serie de bloques para crear la aplicación. El sistema es gratuito, pero hace falta una cuenta de Google para acceder a él.

Primeros pasos

Accede a los siguientes enlaces para conocer cómo instalar esta aplicación y sus principales elementos.

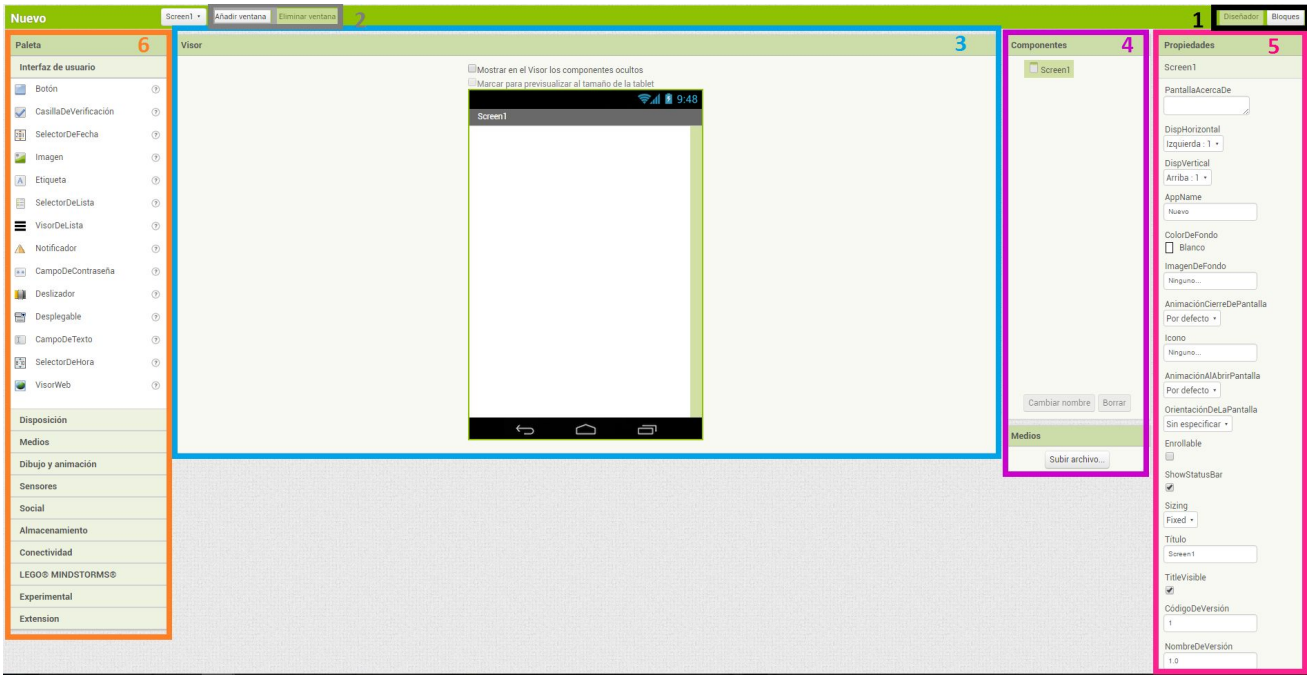
- Instalación en el dispositivo para utilizar AI Companion:
  - https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3&hl=es
- Primeros pasos:
  - http://appinventor.mit.edu/explore/get-started.html
  - http://diwo.bq.com/instalando-app-inventor-primeros-pasos-y-hola-mundo/
- Documentación extra (tutoriales, cómo conectar el programa con un dispositivo…):
  - http://appinventor.mit.edu/explore/library.html

Controla tu casa domótica

Antes de comenzar a crear la aplicación de la casa domótica, es necesario que realicemos con los alumnos una serie de actividades que les ayuden a aprender y conocer esta aplicación.

Para ello, iremos a la página del programa (http://ai2.appinventor.mit.edu/) y nos registraremos (utilizando una cuenta de gmail).

Una vez que hemos iniciado sesión, deberemos decidir en qué idioma trabajar con los alumnos (castellano o inglés, siendo éste último recomendado para que los alumnos se familiaricen con el lenguaje de programación). Tras esto, le daremos a *Comenzar un proyecto nuevo* y les explicaremos las partes del programa.



1. Diseñador/Bloques: Podemos identificar dos partes principales en una aplicación:

- Diseñador o interfaz: en la que se creará la estética de la aplicación.
- Bloques o programación: donde se programará mediante bloques qué va a hacer la aplicación.

2. Ventanas: La aplicación puede diseñarse en una sola pantalla o puede tener varias. Mediante estos botones, se podrán añadir o eliminar pantallas.

3. Visor: Es la pantalla de visualización de la aplicación. Aquí diseñaremos la interfaz.

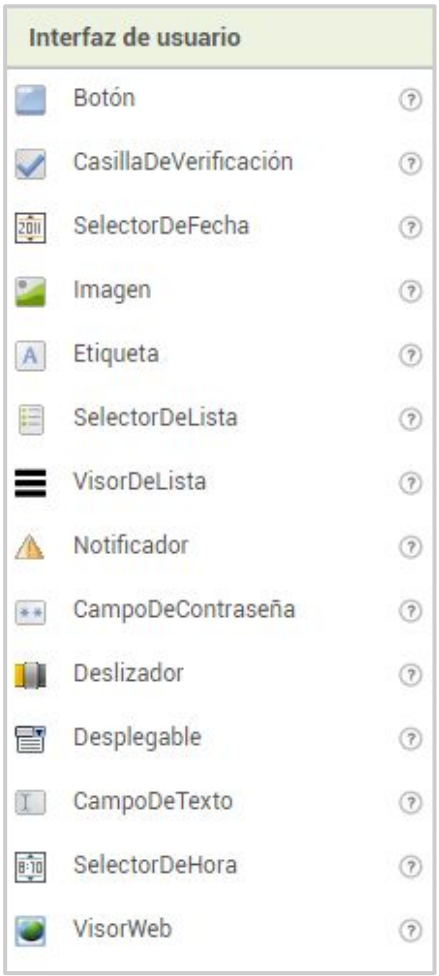
4. Componentes: Aquí será donde se muestran todos los componentes o elementos que vamos añadiendo a la pantalla.

5. Propiedades: Esta ventana te permitirá cambiar algunas de las propiedades de los elementos (color, tamaño de texto, etc.).

6. Paleta: En esta sección encontraremos los elementos o componentes que queremos añadirle a nuestra aplicación.

Podemos encontrar los siguientes elementos con las siguientes funciones:

- **Interfaz de usuario:** son elementos que se utilizan para crear la interfaz, es decir, para diseñar la pantalla de la aplicación.
    - Botón (*Button*): detecta cuándo se hace clic sobre él.
    - Casilla de verificación (*CheckBox*): permite activar o desactivar una casilla, generándose un evento al cambiar su estado.



- Selector de fecha (*DatePicker*): es un botón que, al pulsarlo, abre una ventana de diálogo que permite al usuario seleccionar una fecha.
- Imagen (*Image*): permite mostrar fotos. Se puede seleccionar la foto que se desea mostrar en la opción *Medios*, pinchando en *Propiedades* → *Foto* y seleccionando el nombre de la foto deseada. Además, si se quiere ir cambiando de foto a lo largo de la aplicación, existen bloques para ello.
- Etiqueta (*Label*): sirve para mostrar un texto. Utilizando la pestaña propiedades, o mediante bloques, se podrá cambiar algunas de sus características, como la apariencia (color, tamaño, tipo de letra...), su ubicación, etc.
- Selector de lista (*ListPicker*): es un botón que, al pulsarlo, muestra una lista de textos entre los cuales el usuario puede elegir. Para definir los textos se deberá especificar en *Propiedades* → *ElementosDesdeCadena* añadiendo las opciones separadas mediante comas.
- Visor de lista (*ListView*): componente visible que permite definir una lista de textos para que se muestren en la pantalla.
- Notificador (*Notifier*): muestra cuadros con alertas, mensajes y alertas temporales.
- Campo de contraseña (*PasswordTextBox*): es un cuadro para escribir contraseñas. Es como el componente *CampoDeTexto*, pero no se muestran los caracteres mientras el usuario los está escribiendo.
- Deslizador (*Slider*): es una barra de progreso con un marcador que puede desplazarse. Al tirar del marcador y arrastralo a la izquierda o a la derecha éste cambia de posición.
- Desplegable (*Spinner*): es un componente desplegable que muestra una ventana emergente que contiene una lista de elementos.
- Campo de texto (*TextBox*): es un campo que permite al usuario introducir texto.
- Selector de hora (*TimePicker*): es un botón que cuando se pulsa muestra una ventana que permite al usuario definir una hora.
- Visor web (*WebViewer*): Es un componente para enlazar páginas web. Se podrá definir la url en *Propiedades* → *UrlInicial*.

- **Disposición:** sirve para colocar los elementos o componentes de la *interfaz de usuario* en el *visor*. Hay varios tipos dependiendo de la disposición: horizontal (para colocar los elementos de izquierda a derecha), vertical (para colocar los elementos uno debajo de otro), tabular (para hacer tablas cuyo número de filas y/o columnas se puede cambiar en *Propiedades*).
- **Medios:** son componentes no visibles que sirven para añadir a la aplicación material multimedia (fotos, vídeos, reconocimiento de voz…).
- **Dibujo y animación:** permite incluir animaciones e interaccionar con la aplicación creando dibujos.
- **Sensores:** permite utilizar los sensores incluidos en el dispositivo, como por ejemplo, el sensor de proximidad, acelerómetro, etc.
- **Social:** para compartir información de la aplicación con otras aplicaciones o contactos.
- **Almacenamiento:** crear bases de datos o guardar datos en un archivo.
- **Conectividad:** está destinada a abrir otras aplicaciones, como la cámara, o para conectarse con otros dispositivos. El elemento más utilizado es el Bluetooth, ya que es el que conectará la placa controladora del kit de robótica con la aplicación Android.

Tras esto, vamos a la pestaña *Bloques* y mostramos su estructura, explicando las diferentes secciones (Control, Lógica…).

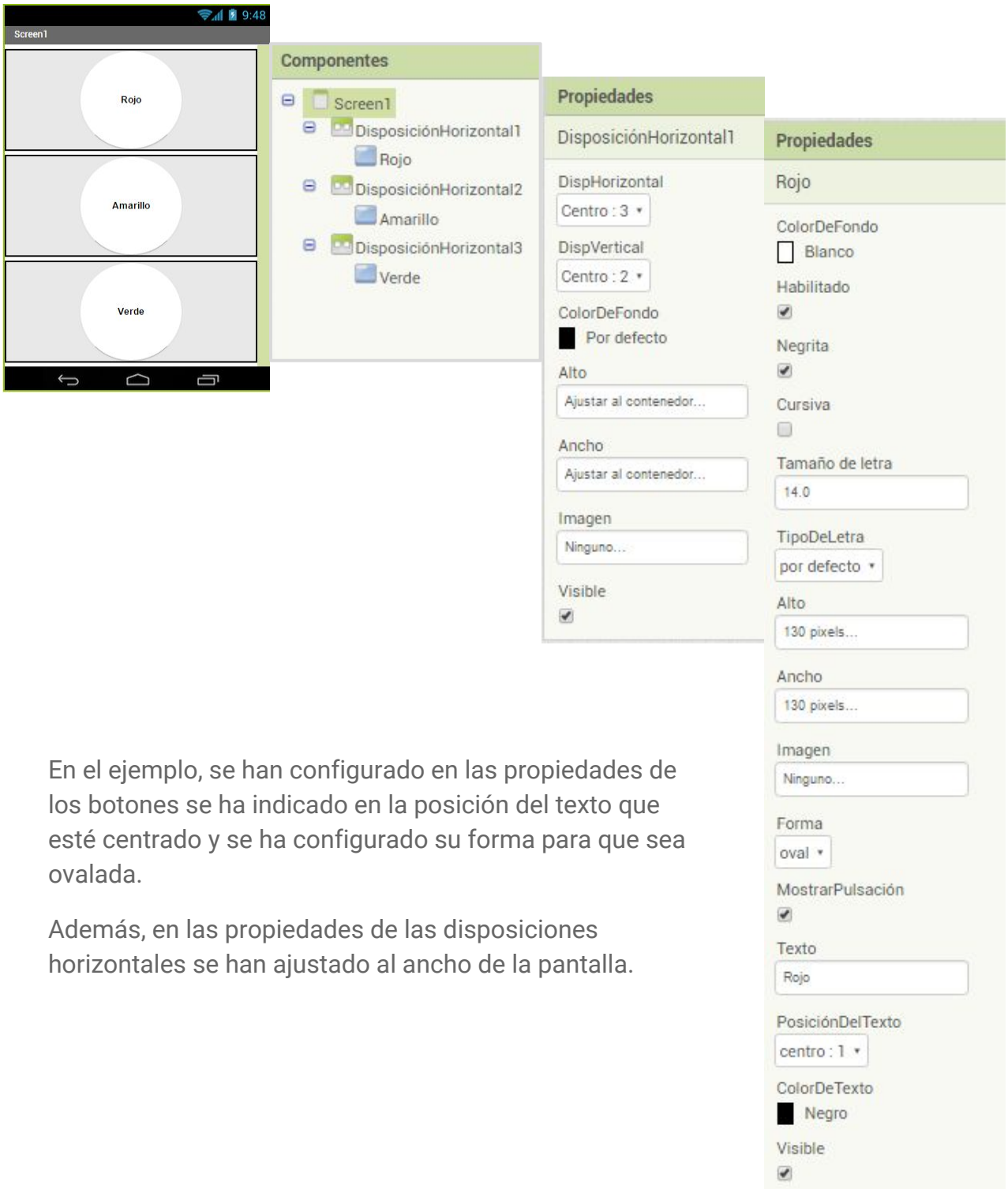
Para comenzar a realizar un programa, solo tenemos que seleccionar la sección que corresponda, pinchar en el bloque deseado y arrastrarlo al Visor.

Es importante explicar que mientras que Bitbloq es un programa secuencial, es decir, la programación se lee de arriba a abajo, Appinventor es por eventos, es decir, podemos tener varias acciones que sucedan a la vez.

Una vez que hemos explicado la interfaz, programaremos junto a los alumnos un semáforo, con el fin de crear una aplicación sencilla que permita aprender algunos aspectos básicos de este programa.

Mi primera aplicación

Para empezar, pediremos a los alumnos que creen la interfaz de su aplicación. Las luces de los semáforos serán botones y para colocarlos, se deberán utilizar los elementos que se encuentran en *Disposición*. Un ejemplo podría ser el siguiente:

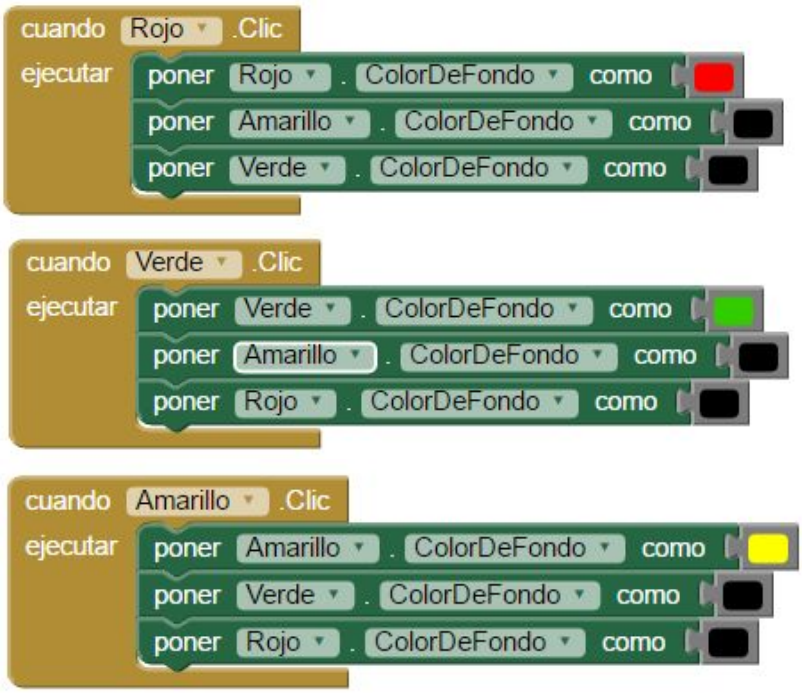


En el ejemplo, se han configurado en las propiedades de los botones se ha indicado en la posición del texto que esté centrado y se ha configurado su forma para que sea ovalada.

Además, en las propiedades de las disposiciones horizontales se han ajustado al ancho de la pantalla.

Una vez creada la interfaz deberán programar el funcionamiento de la aplicación. Programarán que cuando se pulse un botón, éste se muestre con su color correspondiente y el resto de botones se apaguen, es decir, se pongan en negro.

La programación que deberán crear es la siguiente:

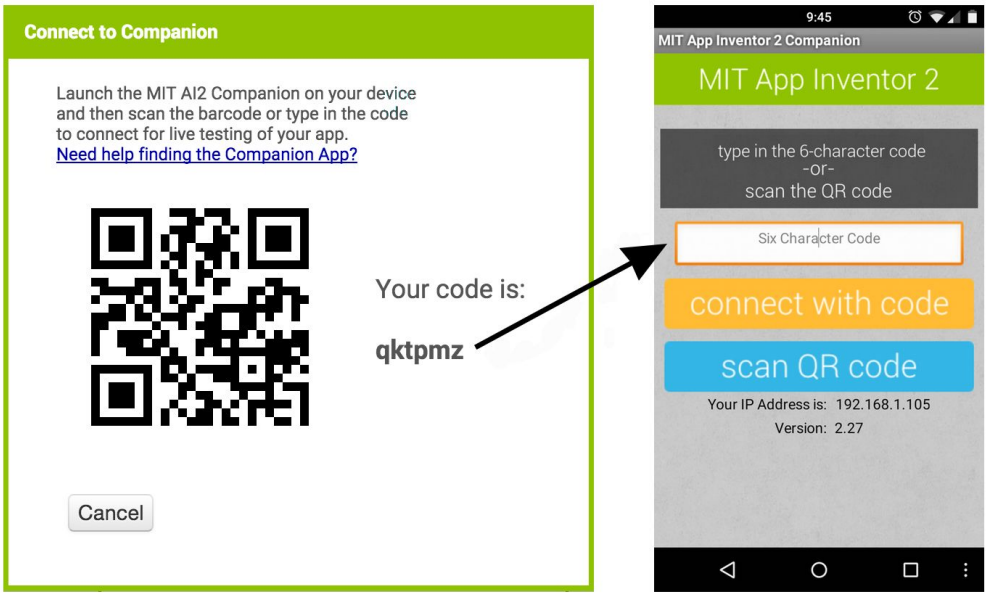


Para que sea más sencillo este programa, podemos pedir a los alumnos que comiencen realizando el semáforo de peatones con dos luces (una de color verde y otra de color rojo) y una vez que hayan aprendido, intenten hacer ellos solos el de coches, añadiendo la luz amarilla.

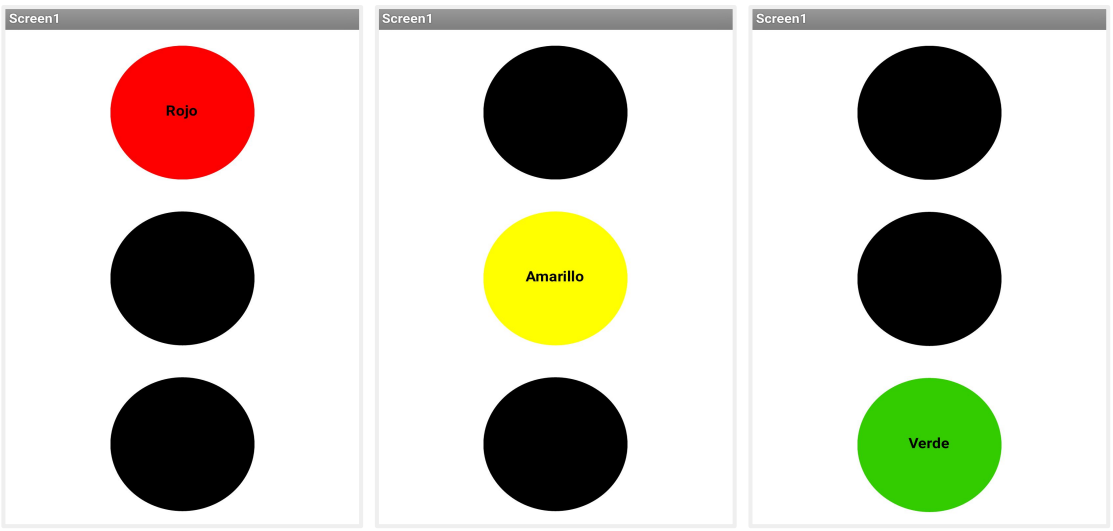
¿Cómo visualizar la aplicación en nuestro dispositivo?

Para visualizar la aplicación deberemos conectar el programa con el móvil. Se podrán utilizar tres opciones: conectarlo por USB, un emulador o AI Companion. Éste último es el más recomendado, ya que los demás necesitan realizar algún tipo de instalación y en el caso del USB, depende del dispositivo que se utilice. Para utilizar esta opción, tanto el dispositivo móvil o tablet y el ordenador deberán estar conectados a la misma red wifi.

Una vez cumplido este requisito, para conectarlo deberemos hacer clic en *Conectar → AI Companion*. Tras esto, nos saldrá un código QR, que deberemos introducir en la aplicación de App Inventor del móvil (MIT AI2 Companion). Para introducirlo, tendremos dos opciones: teclear el código manualmente y hacer clic en *connect with code* o pinchando en *scan QR code* y capturando el código que se muestra.



La aplicación en la pantalla del dispositivo será algo similar a lo siguiente:



Controlando la farola inteligente

Una vez que han creado su primera aplicación, comenzaremos con la integración de App Inventor y el kit de robótica. Para ello, crearemos la programación de la farola inteligente de la casa domótica de manera que se pueda encender o apagar el LED desde nuestro dispositivo Android.

Comenzaremos abriendo un nuevo proyecto y crearemos la interfaz de la aplicación y la programación en App Inventor.

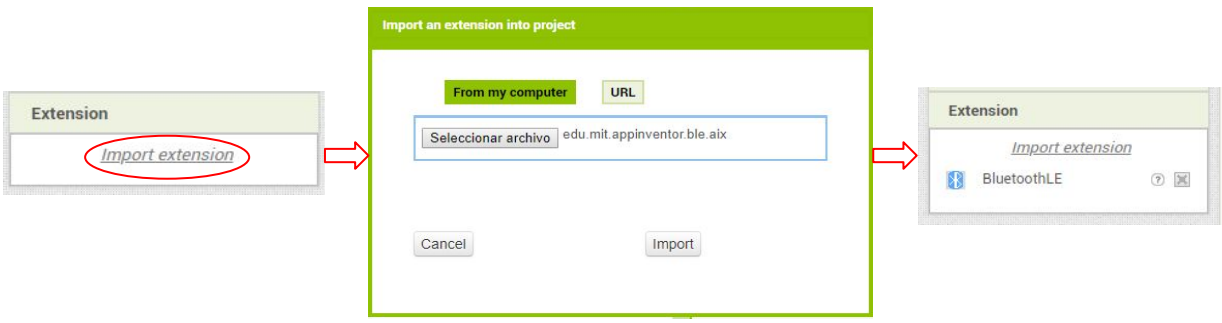
Para vincular por Bluetooth un dispositivo Android con la placa Zum Core 2.0, se debe importar en primer lugar la extensión correspondiente al BluetoothLE. Comenzaremos descargando el archivo de la web de App Inventor http://appinventor.mit.edu/extensions/

Supported:

Name	Description	Author	Download .aix File	Source Code
BluetoothLE	Adds as Bluetooth Low Energy functionality to your applications. See BluetoothLE Documentation and Resources for more information.	MIT App Inventor	BluetoothLE.aix	BluetoothLE-Source.zip

Note: The BluetoothLE component extension, was in part made possible by a grant given by the University Program Office at Intel Corporation.

Una vez descargado el archivo, haremos clic en *Import extension*, en el apartado *Extension*, e importaremos desde nuestro ordenador el archivo descargado (edu.mit.appinventor.ble.aix).

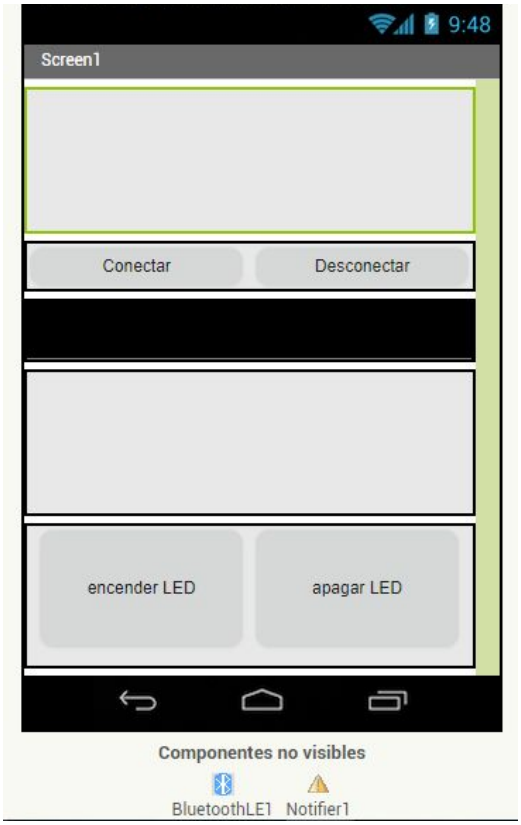


A continuación diseñaremos la interfaz, que tendrá los siguientes elementos:

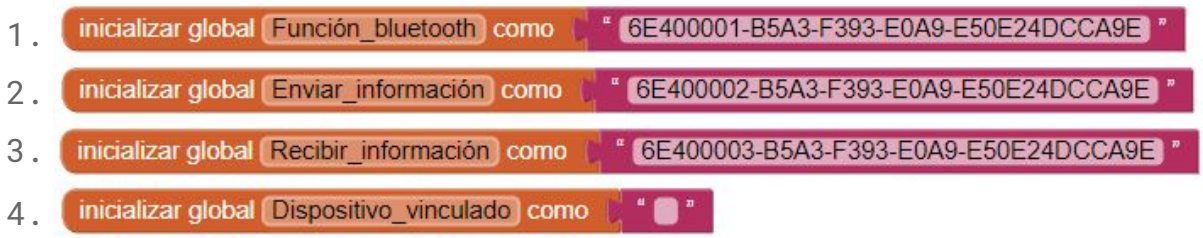
- Dos *botones*: uno para buscar los dispositivos Bluetooth disponibles, al que llamaremos “Conectar”, y otro para desconectar el dispositivo.
- Un *VisorDeLista*, que mostrará el listado de dispositivos Bluetooth detectados. Esto nos permitirá seleccionar el Bluetooth correspondiente a nuestra placa controladora y conectarla con el dispositivo.
- Dos *botones* más para encender y apagar el LED.

Además, en la interfaz, tendremos que añadir el componente *BluetoothLE* (anteriormente añadido en el apartado *Extensión*) para que se realice la conexión entre la aplicación y la placa controladora, y el *Notificador* (en *Interfaz de usuario*) que nos permitirá mostrar mensajes o alertas. Ambos elementos, no serán visibles en la interfaz de la aplicación.

Un ejemplo sencillo es el siguiente:



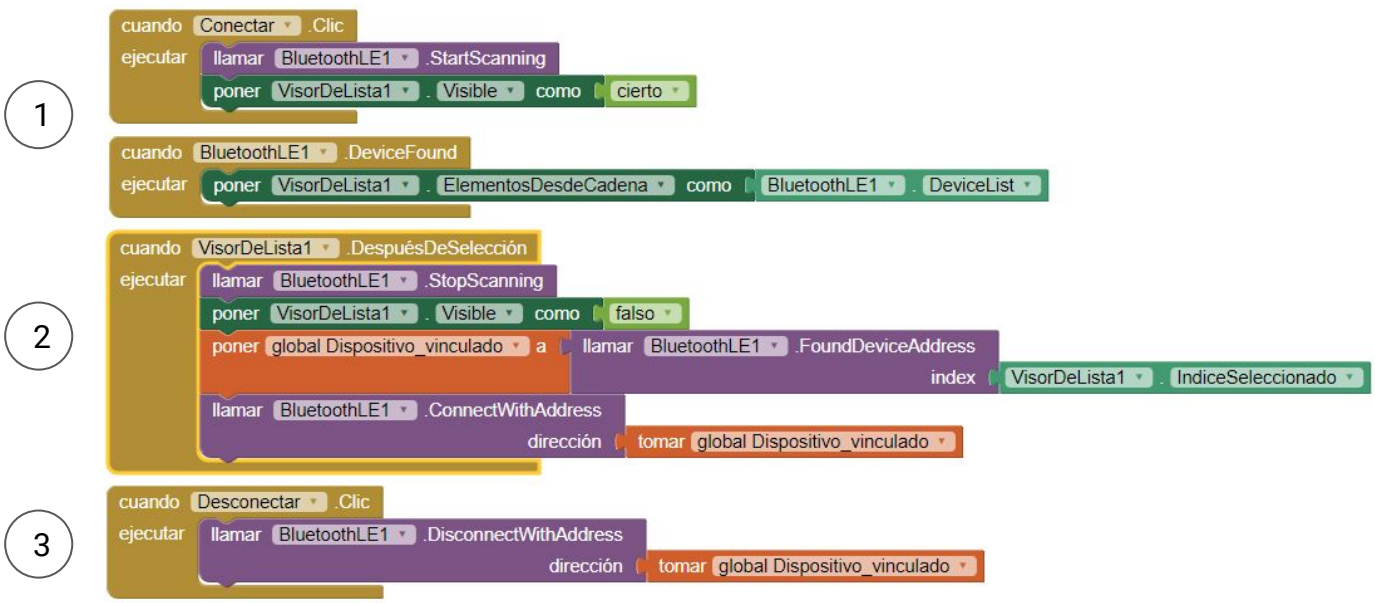
A continuación, programaremos qué funciones queremos que tenga cada botón. Para ello, en el apartado *Bloques* comenzaremos poniendo los bloques correspondientes para la conexión del Bluetooth con la placa controladora. Recomendamos que estos bloques, al ser siempre los mismos, se muestren a los alumnos.



Primero se añaden cuatro variables para guardar la siguiente información:

1. El código que determina la función del *Bluetooth*: enviar y recibir información: 6E400001-B5A3-F393-E0A9-E50E24DCCA9E
2. El código para que el *Bluetooth* envíe información: 6E400002-B5A3-F393-E0A9-E50E24DCCA9E
3. El código para que el *Bluetooth* reciba información: 6E400003-B5A3-F393-E0A9-E50E24DCCA9E
4. El nombre del dispositivo al que se vaya a conectar.

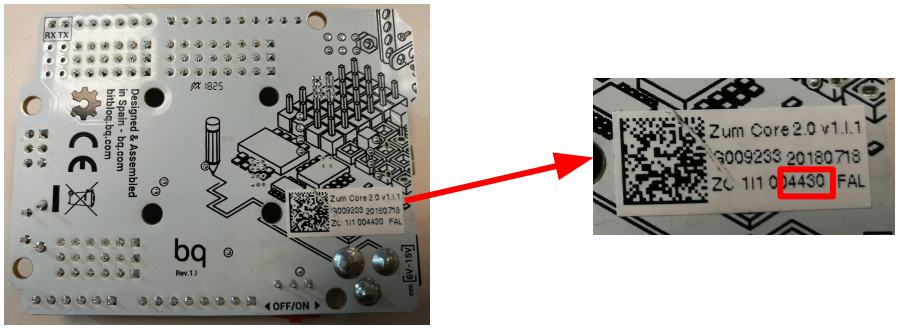
A continuación pondremos el resto de bloques:



La función de cada una de estas programaciones es la siguiente:

1. Al hacer clic en el botón *Conectar* se activa el *Bluetooth* y se muestran en el *VisorDeLista* los dispositivos disponibles.

   Es importante saber que el nombre de nuestra placa controladora se corresponde con los 4 últimos números que aparecen impresos en la pegatina de la parte trasera de la misma.



2. Tras seleccionar un dispositivo de la lista, se esconde el *VisorDeLista,* se guarda el nombre del dispositivo en la variable *Dispositivo_vinculado* y se establece la conexión *Bluetooth*.

3. Al hacer clic sobre el botón *Desconectar,* el *Bluetooth* se desconecta del dispositivo al que se ha vinculado.

Con estos dos últimos grupos de bloques se muestran las alertas o avisos en la aplicación, informando de que se ha realizado la conexión Bluetooth o que se ha desconectado.

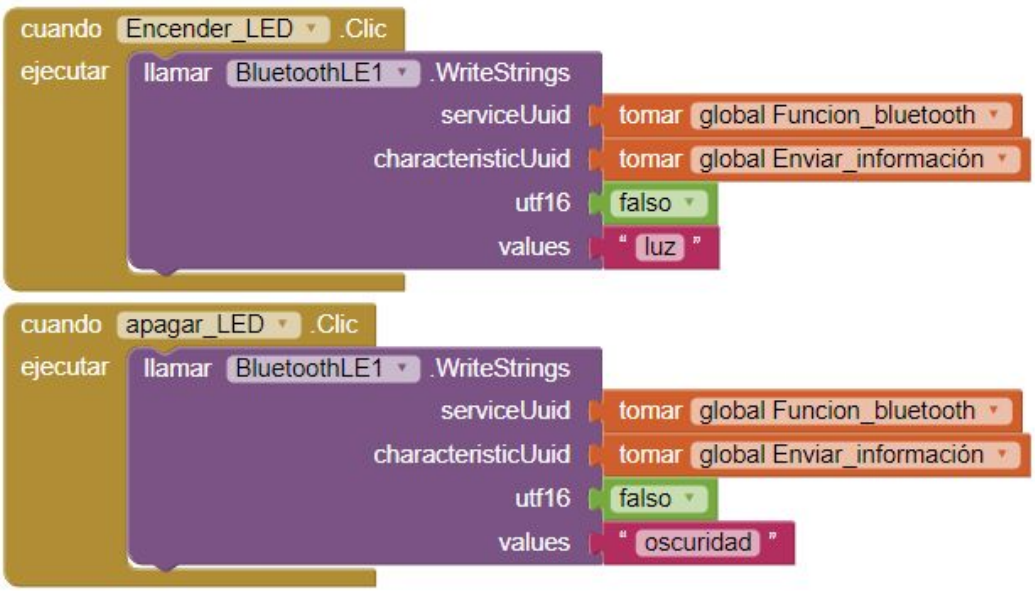
En vez de un *Notificador*, también podríamos poner una *Etiqueta* para mostrar en pantalla el estado de conexión o una *Imagen* que vaya cambiando.

Una vez que tienen la conexión, se deberá programar la función de los otros dos botones, para que apaguen y enciendan un LED.

Comenzaremos programando que se encienda un LED con un botón. Para ello, programaremos que al hacer clic sobre el botón, llamaremos al Bluetooth y le indicaremos:

- el tipo de servicio que se va a realizar (variable *Función_bluetooth*)
- la característica concreta del servicio (variable *Enviar_información*)
- el tipo de codificación (falso, correspondiente a UTF-8)
- la lista de valores para escribir en el dispositivo (el mensaje para la placa controladora, ej. “luz”)

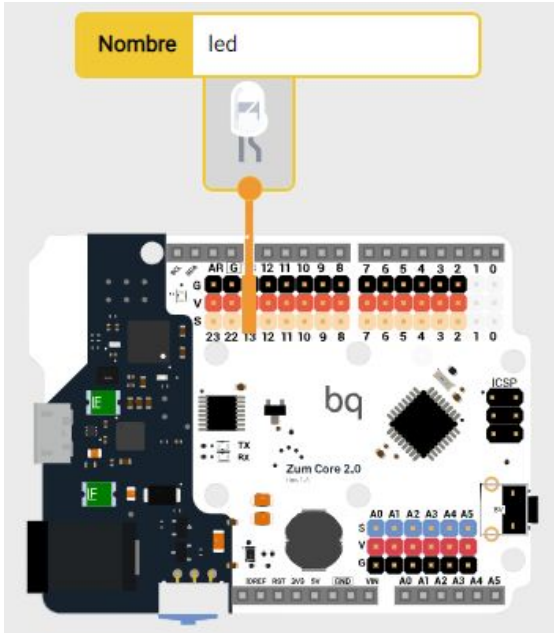
La programación sería similar a la siguiente:



Programación en Bitbloq

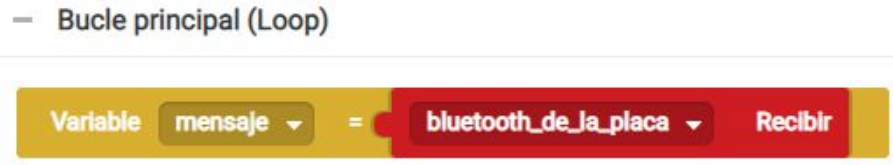
Por último, abriremos Bitbloq y programaremos que cuando la aplicación le mande los mensajes que hemos indicado en ella, se encienda o apague el LED.

Para ello, primero conectaremos en la parte de *Hardware* los componentes que vamos a utilizar.

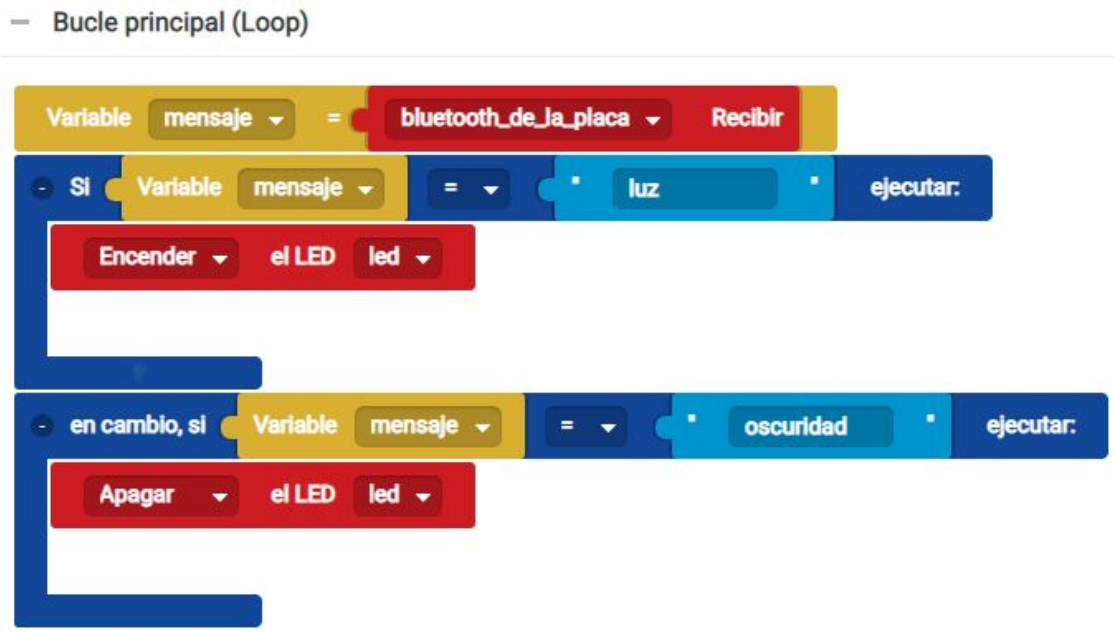


A continuación, iremos a la pestaña *Software* y crearemos en el apartado *Variables globales, funciones y clases*, una variable a la que llamaremos “mensaje” que sea igual a un texto vacío. Esta variable guardará los mensajes que son enviados por la aplicación.

Tras esto, en la parte *Bucle principal (Loop)* guardaremos en la variable que hemos creado anteriormente lo que se reciba por el bluetooth. Esto le indicará al programa que debe almacenar los mensajes que reciba por bluetooth, es decir, los mensajes que se envíen desde la aplicación.



Debajo de este bloque, comenzaremos a programar qué tiene que hacer el programa cuando reciba los mensajes de la aplicación. Para ello, indicaremos que, si la variable mensaje es igual a “luz” (que es el mensaje que pusimos en App Inventor) se encienda el LED. Añadiremos también el otro mensaje, es decir, que si la variable es igual a “oscuridad” se apague el LED.



Hay que tener mucho cuidado a la hora de poner el mensaje, ya que éste deberá ser igual que el que enviamos desde la aplicación (signos de puntuación, mayúsculas, minúsculas…).

¿Cómo instalar la aplicación que he creado en mi dispositivo?

Para poder comprobar que todo funciona correctamente, deberemos cargar el programa a la placa controladora, abrir App Inventor y conectar la aplicación con la placa.

Una vez que hemos comprobado que el programa es correcto, podemos generar la aplicación (archivo .apk) desde App Inventor → *Generar*. Aquí dará dos opciones, una que guarda el archivo .apk en el ordenador, que deberemos pasar al dispositivo Android y ejecutarlo, y otra que genera un QR que podemos capturar con el móvil para que se descargue la aplicación y luego ejecutarla.

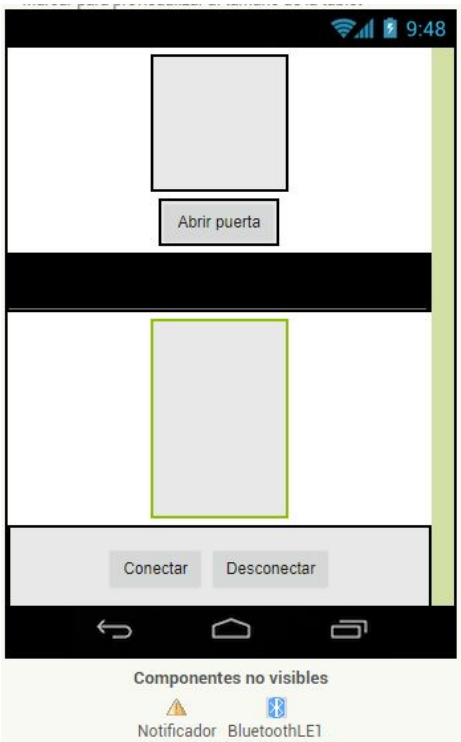
Al tener instalada la aplicación en el dispositivo, ya no hará falta estar conectado al mismo wifi, ni conectarlo por AI Companion. Sólo bastará con conectar el dispositivo a la placa controladora por Bluetooth.

Una vez tengamos esta aplicación, ya podremos controlar la farola de nuestra casa. Podemos personalizar los mensajes de la aplicación que hemos creado adaptándola a la farola como se muestra en el archivo “RetoTech_P2_Farola_ZumCore2.aia”.

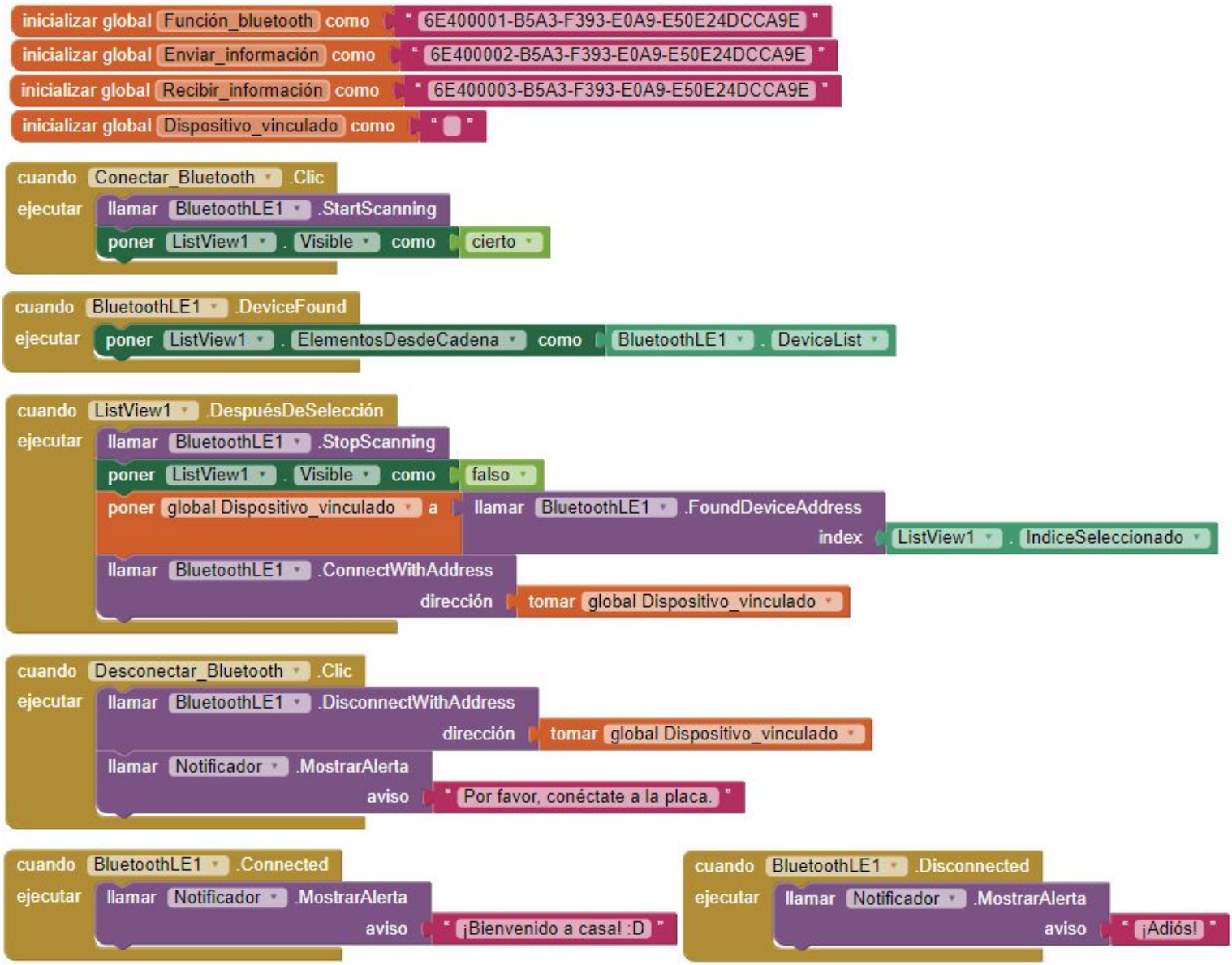
Controlando la puerta automática

A continuación crearemos la aplicación que controle la puerta del garaje. Para ello, empezaremos creando una aplicación sencilla, en la que tras pulsar un botón, se abra la puerta del garaje solo cuando haya un coche en la puerta. Si pulsamos el botón y la placa controladora detecta que no hay coche, la puerta no se abrirá.

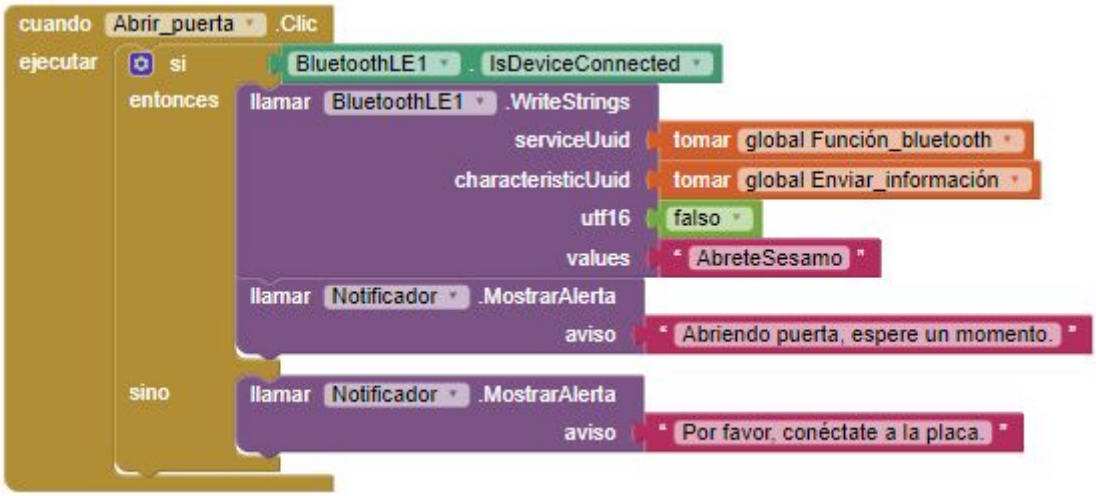
Comenzaremos diseñando la interfaz. Ésta deberá contar, al igual que en el programa anterior, con los botones *Conectar* y *Desconectar y el visor de lista* , para realizar la conexión entre el dispositivo y la placa controladora. Además, se deberá añadir un botón para abrir la puerta y los elementos no visibles *BluetoothLE* y *Notificador*.



A continuación, crearemos la programación. Para ello, colocaremos los bloques necesarios para la conexión del Bluetooth.


inicializar global Función_bluetooth como " 6E400001-B5A3-F393-E0A9-E50E24DCCA9E "
inicializar global Enviar_información como " 6E400002-B5A3-F393-E0A9-E50E24DCCA9E "
inicializar global Recibir_información como " 6E400003-B5A3-F393-E0A9-E50E24DCCA9E "
inicializar global Dispositivo_vinculado como " "
cuando Conectar_Bluetooth .Clic
ejecutar llamar BluetoothLE1 .StartScanning
poner ListView1 . Visible como cierto
cuando BluetoothLE1 .DeviceFound
ejecutar poner ListView1 . ElementosDesdeCadena como BluetoothLE1 . DeviceList
cuando ListView1 .DespuésDeSelección
ejecutar llamar BluetoothLE1 .StopScanning
poner ListView1 . Visible como falso
poner global Dispositivo_vinculado a llamar BluetoothLE1 .FoundDeviceAddress
index ListView1 . IndiceSeleccionado
llamar BluetoothLE1 .ConnectWithAddress
dirección tomar global Dispositivo_vinculado
cuando Desconectar_Bluetooth .Clic
ejecutar llamar BluetoothLE1 .DisconnectWithAddress
dirección tomar global Dispositivo_vinculado
llamar Notificador .MostrarAlerta
aviso " Por favor, conéctate a la placa. "
cuando BluetoothLE1 .Connected
ejecutar llamar Notificador .MostrarAlerta
aviso " ¡Bienvenido a casa! :D "
cuando BluetoothLE1 .Disconnected
ejecutar llamar Notificador .MostrarAlerta
aviso " ¡Adiós! "

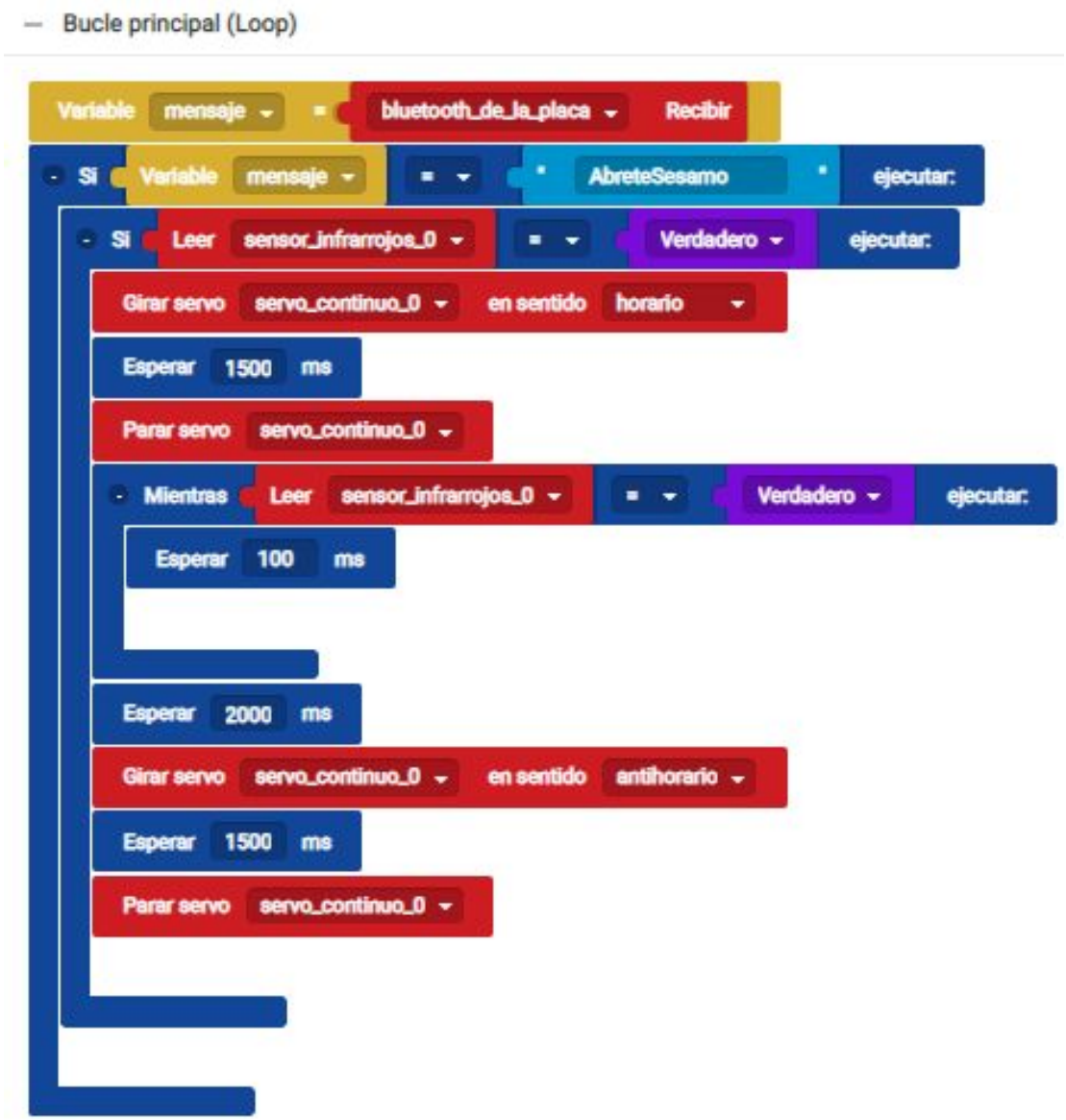
Para completar la aplicación, tendremos que programar qué tiene que hacer el botón de *Abrir puerta*. En este caso, al hacer clic sobre él, llamará al Bluetooth para indicarle el tipo de servicio que va a realizar y la característica del servicio, el tipo de codificación y la lista de valores, que es el mensaje que queremos enviar a la placa para que ejecute que se abra la puerta. Podemos añadir un aviso para informar al usuario de que la puerta se está abriendo. En el caso de que al pulsar el botón *Abrir puerta* el Bluetooth no esté conectado, se deberá programar que aparezca un aviso al usuario de que tiene que conectarse a la placa.



Una vez que ya tenemos nuestra aplicación, pasaremos a programar con Bitbloq qué tiene que hacer nuestra placa controladora. Para ello, abriremos el programa que creamos para la apertura de la puerta automática en el primer proyecto RetoTech.

En la parte de *Hardware,* conectaremos los componentes que vamos a utilizar, en este caso, el servo de rotación continua o miniservo y el sensor de infrarrojos.

En *Software*, tendremos que declarar en *Variables globales, funciones y clases* una variable que sea igual a un texto vacío. Tras esto, iremos a la parte de *Bucle principal (Loop)* y encima de la programación igualaremos la variable que hemos creado a lo que reciba del bluetooth. Por último, incluiremos la programación de la puerta dentro de un condicional que compruebe que si el mensaje recibido es el mismo (en nuestro caso era "AbreteSesamo"), se ejecute la programación de apertura de la puerta.



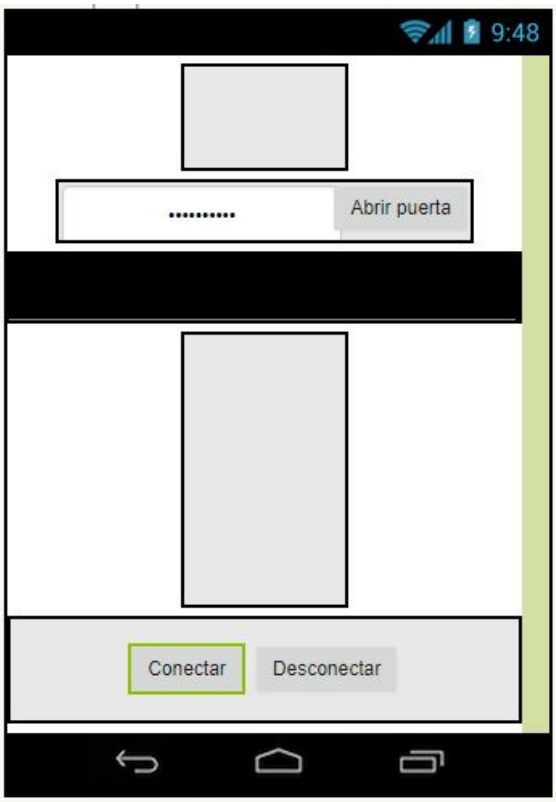
Para poder comprobar que todo funciona, al igual que se hizo con la aplicación anterior, deberemos cargar el programa a la placa controladora, abrir la aplicación de App Inventor y conectar el dispositivo con la placa.

Podemos ver este programa en los archivos “RetoTech_P2_Puerta_Simple_ZumCore2.aia” y “RetoTech_P2_Puerta_Automatica_ZumCore2.bitbloq”

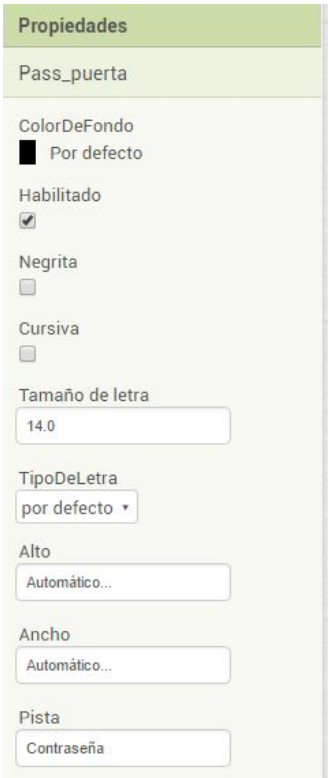
Como reto extra, podemos proponerles que intenten crear una contraseña para abrir la puerta, de esta forma no se abrirá hasta que no se haya introducido la contraseña correcta.

Para este reto, solo tendrán que cambiar la aplicación de App Inventor, puesto que la programación de la placa controladora será la misma.

Para programar ésto, primero tendrán que cambiar la interfaz de la aplicación y añadir un campo de contraseña para poder introducir la clave.

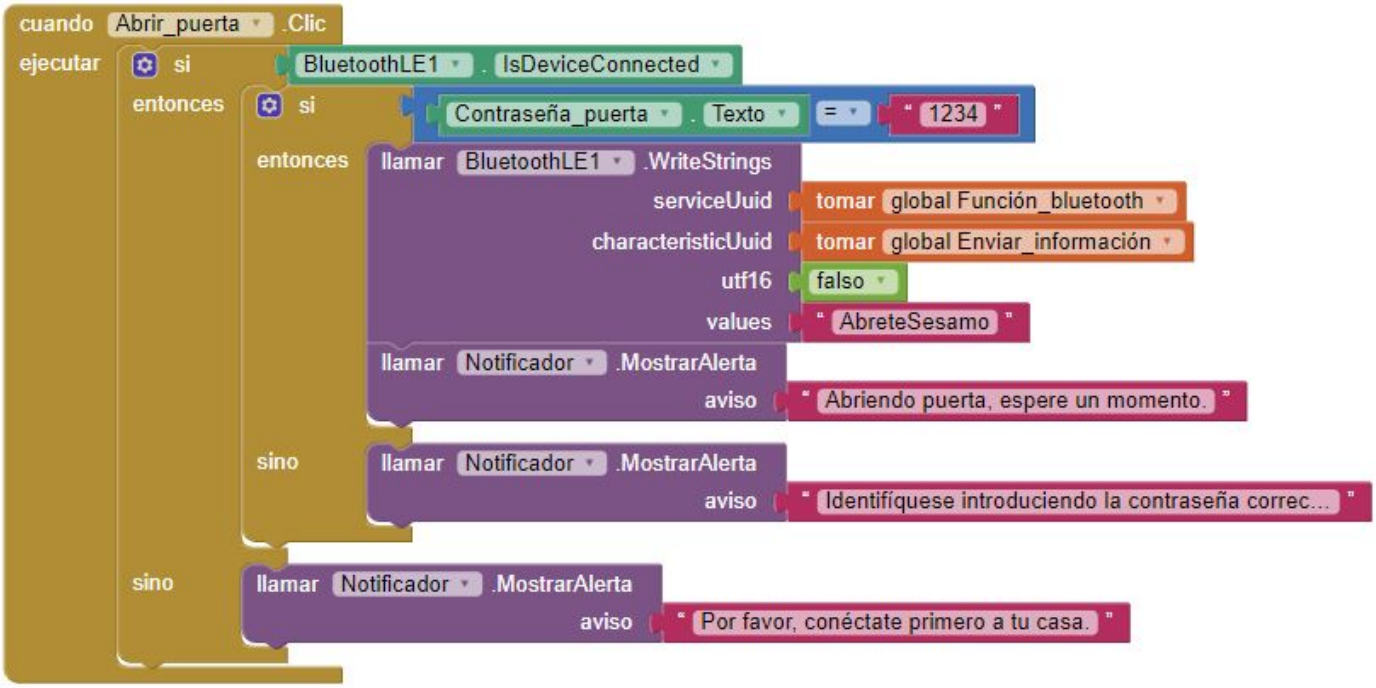


A continuación, deberán añadir dentro del campo de contraseña una pista para que el usuario sepa qué tiene que introducir en el recuadro. Para ello, seleccionamos el elemento y en *Propiedades* → *Pista* añadimos el texto "Contraseña", como se muestra en la imagen de la derecha.



En caso de que no se añada una pista, saldrá un campo de texto vacío.

Por último, tendrán que ir al apartado *Bloques* y modificar lo que hace el botón de abrir puerta, añadiendo un condicional que indique que si el texto de la contraseña introducida es igual a la contraseña que queramos, por ejemplo, "1234", abra la puerta, de lo contrario, que ponga un texto que indique que se introduzca la contraseña.



Modificando este bloque, ya tendrían su aplicación terminada. Sólo faltaría que prueben que todo funciona correctamente y mejorar el diseño de la aplicación (pueden poner color al botón, añadir un fondo, etc).

Podemos ver el programa de ejemplo en el archivo:
“RetoTech_P2_Puerta_con_Contraseña_ZumCore2.aia”

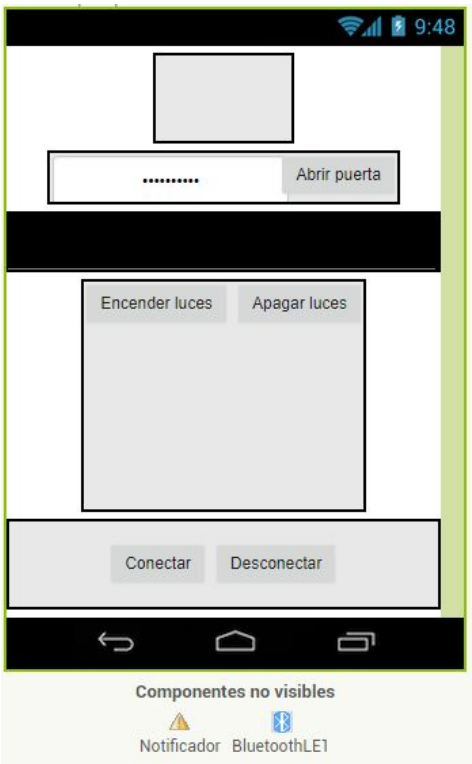
Creando la aplicación de la casa domótica

Para completar este proyecto tendrán que unir las dos aplicaciones en una, de manera que con una sola aplicación se pueda controlar la farola inteligente y la puerta automática.

Para ello, podrán hacer una copia de la programación de la puerta automática con contraseña que han creado anteriormente e ir añadiendo los elementos y programaciones de la farola inteligente.

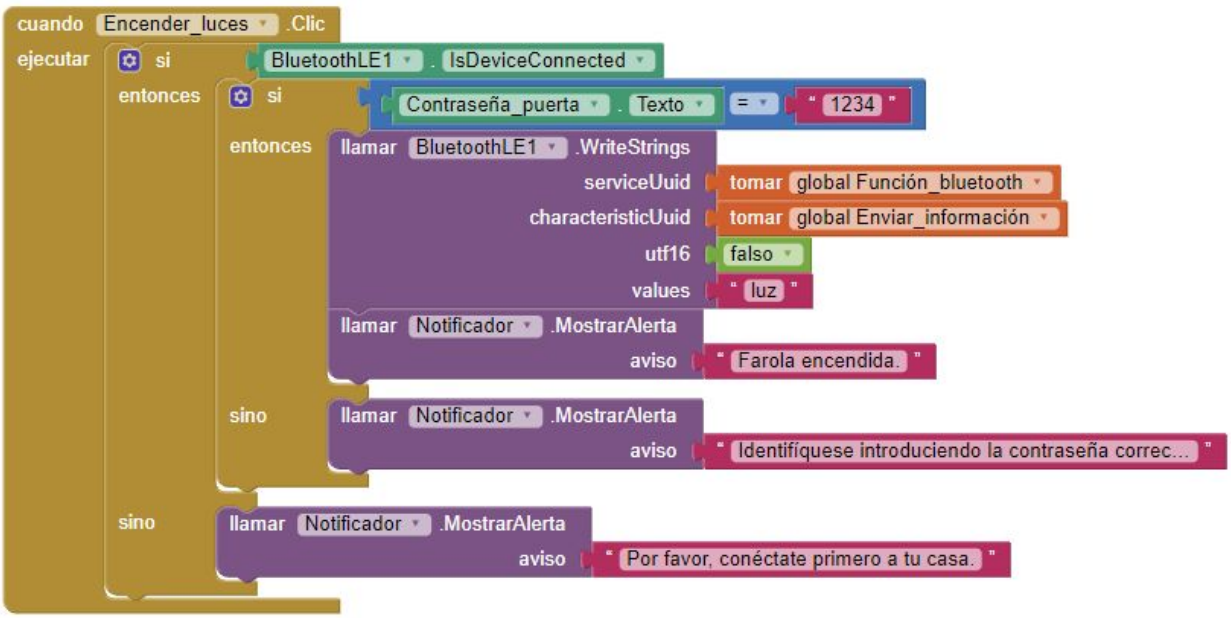
Empezaremos por la interfaz. Ésta deberá contar con 6 elementos:

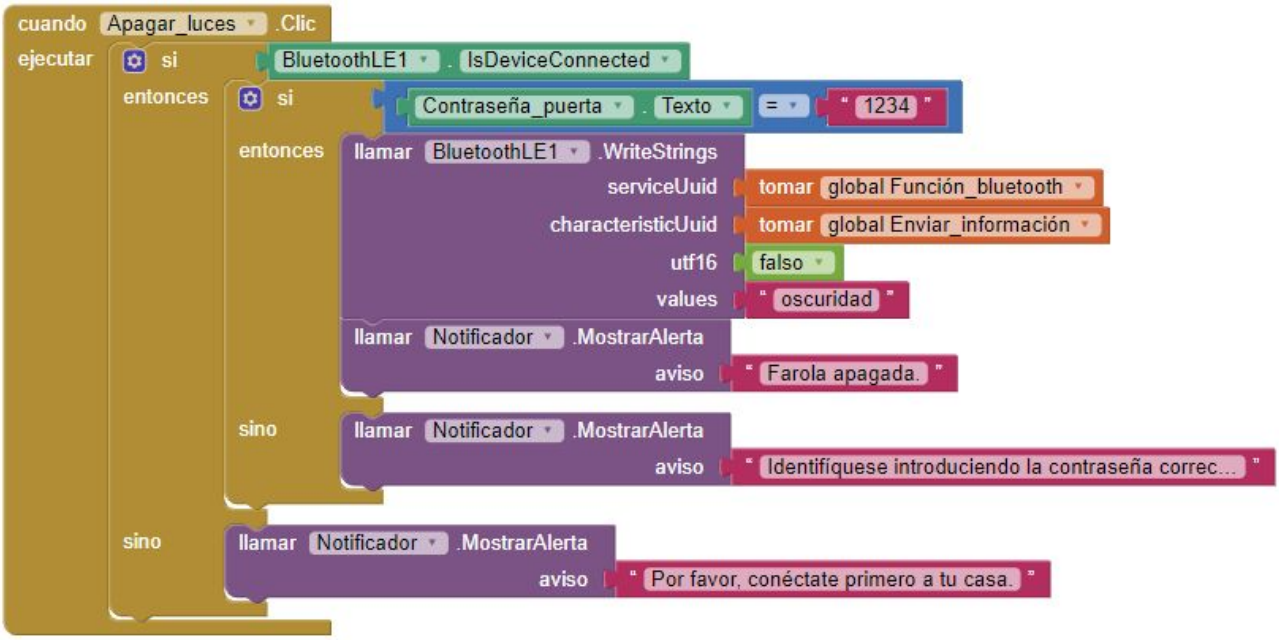
- **Visor de lista:** para mostrar el listado de dispositivos Bluetooth y conectar el dispositivo con la placa controladora.
- **Campo de contraseña:** para proteger mediante una contraseña la aplicación.
- **Cinco botones:** uno para conectar y otro para desconectar el Bluetooth, uno para encender la farola, otro para apagarla y un último para abrir la puerta automática.



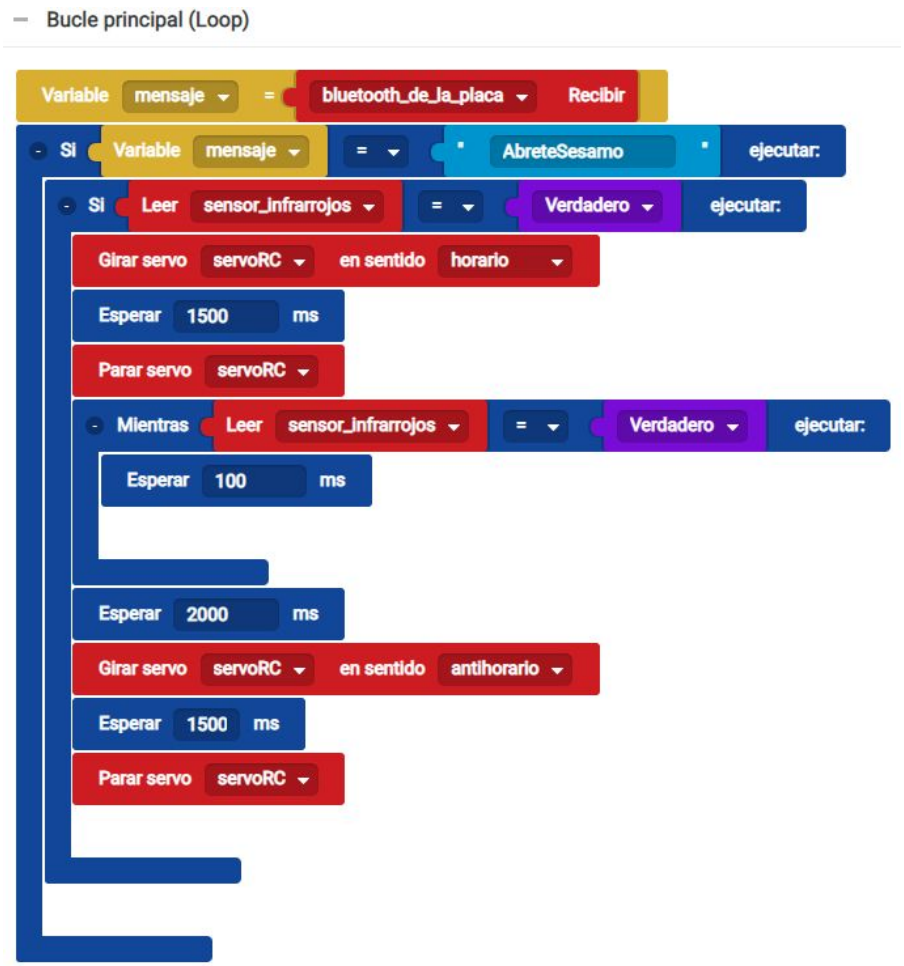
A continuación, procederán a crear la programación añadiendo la programación de la farola con el fin de que hasta que no se introduzca la contraseña de seguridad ésta no se encienda.

Para crearla, tendrán que añadir un condicional en el que indique que si la contraseña introducida es la correcta, se pueda pulsar el botón que enciende la farola, pero si no es la correcta, la aplicación le indique al usuario que introduzca la clave.





Posteriormente, deberán crear la programación en Bitbloq para que abra la puerta y encienda o apague la farola. Para ello, deberán juntar las dos programaciones que crearon anteriormente.

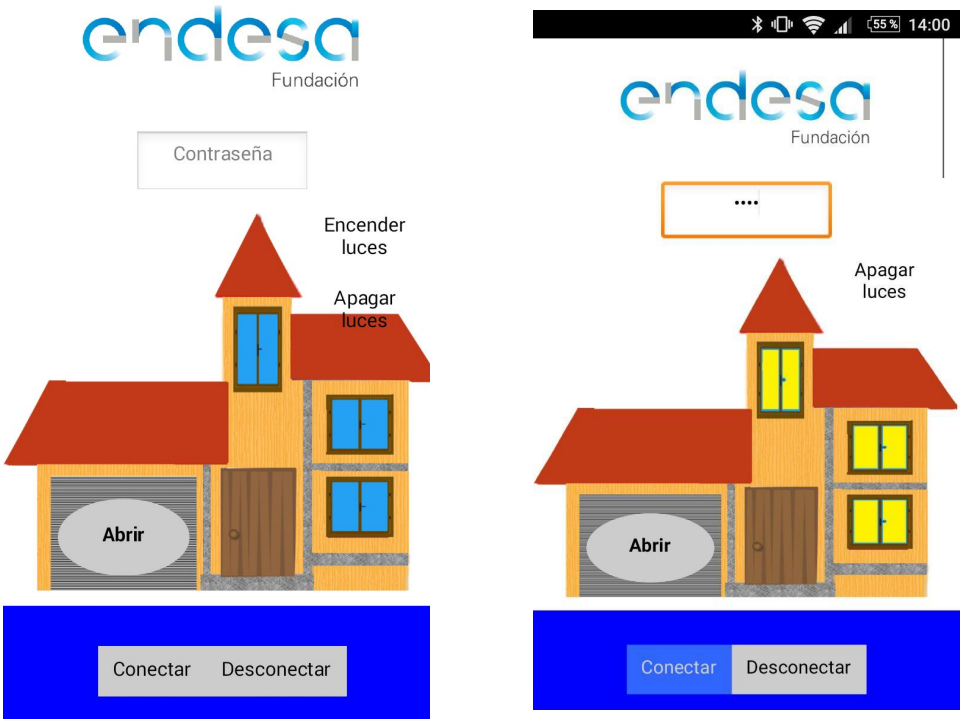


La programación completa se puede ver en el archivo "RetoTech_P2_Casa_simpleZumCore2.aia" y en "RetoTech_P2_Casa_simple_ZumCore2.bitbloq"

Personalizando la aplicación

Una vez programada la casa, deberemos mejorar la interfaz de la aplicación. Es muy importante que la interfaz tenga decoración, con el fin de personalizar y mejorar la apariencia de la aplicación. Puede ser interesante que sugiramos a nuestros alumnos incorporar elementos decorativos que mejoren el aspecto de sus aplicaciones, como pueden ser imágenes, colores, etc.

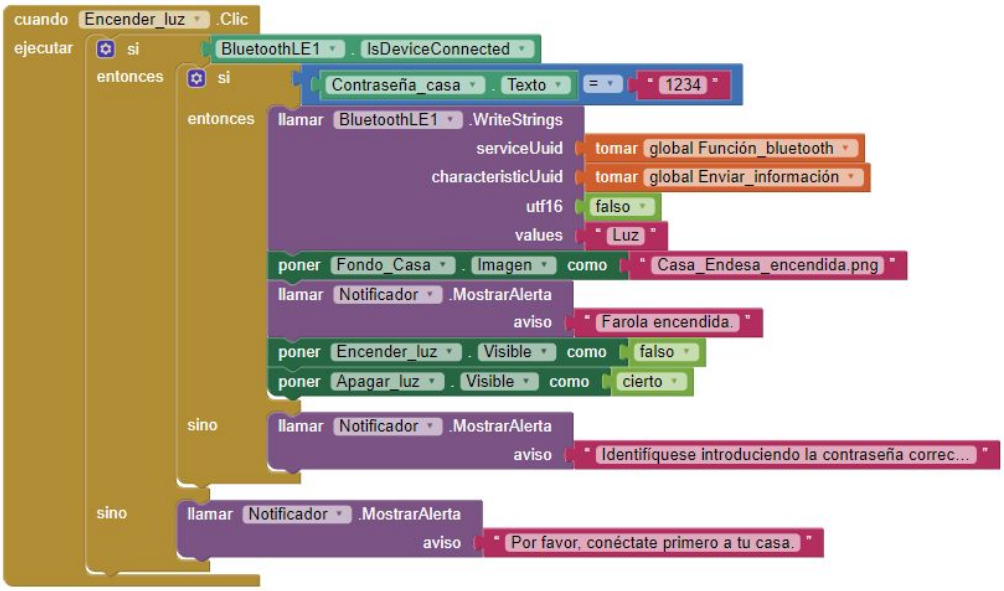
Un ejemplo de interfaz es el siguiente:

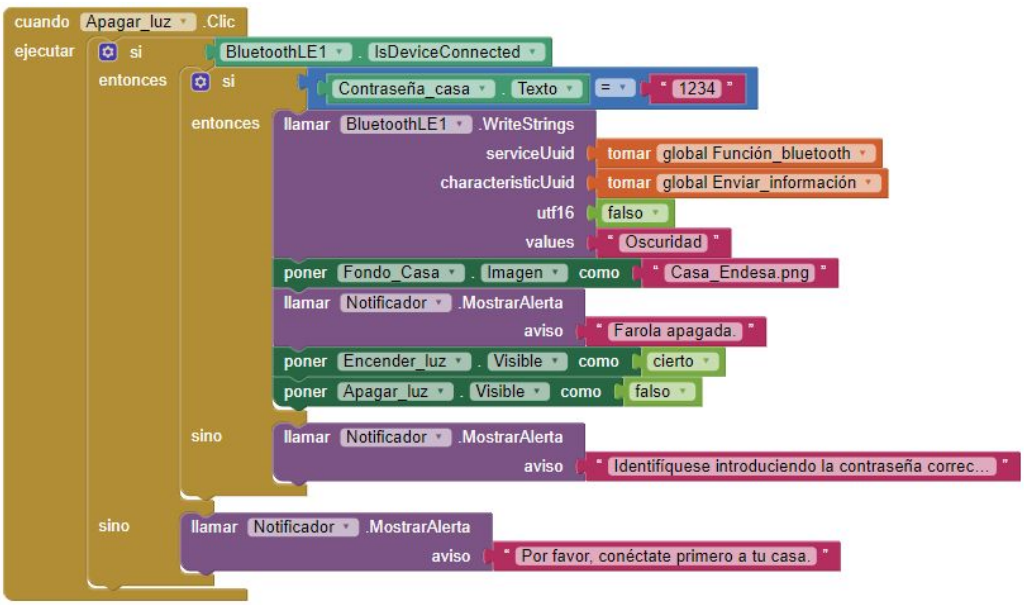


Para diseñar la interfaz se ha añadido una foto de la casa y un logo. Un ejemplo de las funcionalidades que se han añadido al programa son las siguientes:

- Se ha programado que cuando se pulse el botón de *Encender luz* se iluminen las ventanas. Para hacer esto, se han creado dos fondos y se ha programado que cuando se pulse en *Encender luz* se muestre el fondo con las ventanas de color amarillo, y si se pulsa en *Apagar luz* se cambie el fondo que tiene las ventanas sin iluminar.

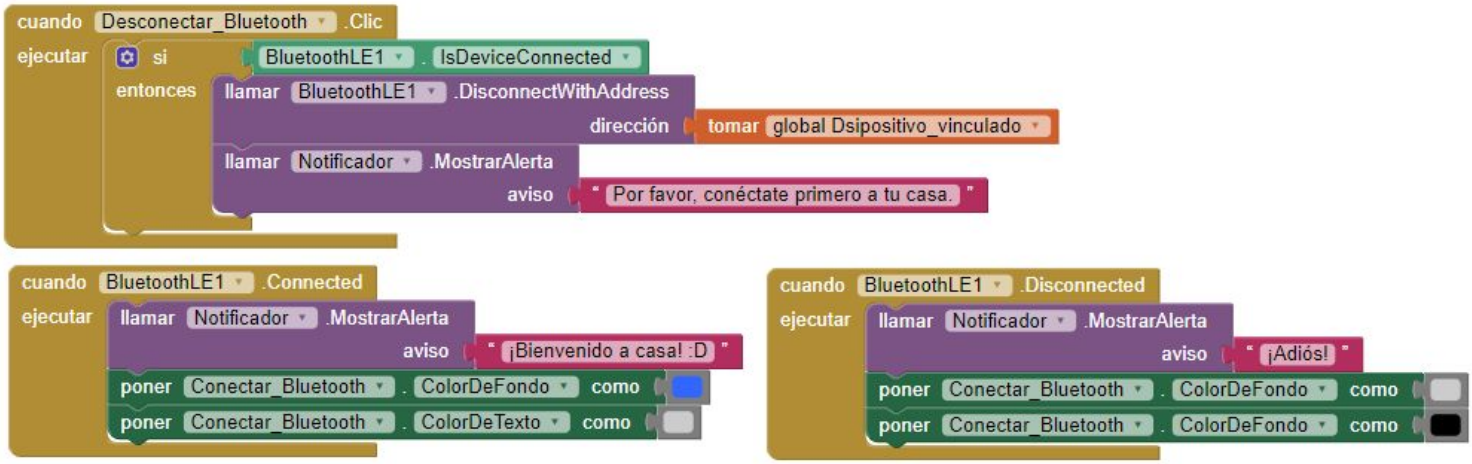
  Además, se ha añadido que al pulsar el botón de *Encender luz*, el botón de apagar se esconda y viceversa. Para ello, se han utilizado los bloques de lógica *falso* y *cierto*, de manera que cuando se pulse el botón *Encender luz*, éste se muestre visible (cierto) y el botón de *Apagar luz* se esconda (falso).





Tal y como se muestra en las imagenes de programación, para cambiar el fondo, se deberá añadir el bloque *Poner fondo imagen como* seguido del bloque texto con el nombre de la imagen y la extensión: "*Casa_Endesa.png*" o "*Casa_Endesa_encendida.png*".

- Se ha programado que cuando se conecta el dispositivo al Bluetooth cambie el botón de *Conectar* de gris a azul y se muestre el texto de color blanco. Para ello, tendrán que ir cambiando el color de los botones como se muestra a continuación:
- También se han añadido mejoras en el botón de *Desconectar*. Primero se ha puesto un condicional que comprueba si el Bluetooth está conectado, con el fin de que si lo está se desconecte. Además, se ha añadido que cambie de color el botón de *Desconectar* cuando se hace clic sobre él, poniendo el fondo gris y el texto negro.



Para que al rotar el dispositivo, la casa no se deforme se puede dejar fija la orientación de la pantalla. Para ello, habrá que seleccionar la pantalla y en *Propiedades* → *OrientacionDeLaPantalla* elegir cómo debe estar (horizontal, vertical…). En este caso, se debería seleccionar *Vertical*.

Podemos ver cómo programar y añadir una interfaz como la mostrada anteriormente, en el archivo "RetoTech_Casa_simple_interfaz_ZumCore2.aia".

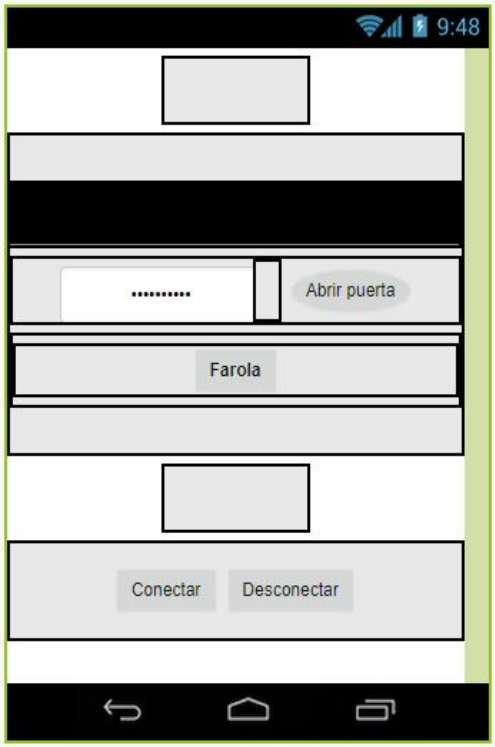
Añadiendo mejoras a nuestra aplicación

Una vez que han terminado el programa anterior, podemos plantear a los alumnos retos extras. Algunas mejoras que pueden añadir a su aplicación son las siguientes:

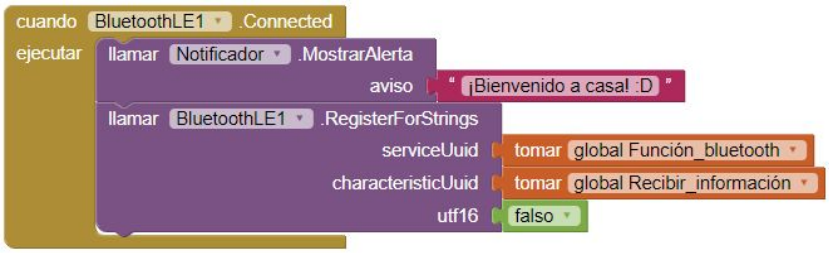
1. Añadir un sensor de luz o LDR, con el fin de que la placa le envíe a la aplicación la cantidad de luz que hay en el lugar y le pregunte al usuario si desea encender la farola. Si la cantidad de luz es alta, se mostrará un mensaje que indique que hay un gasto de energía excesivo.
2. Enviar desde la placa controladora un mensaje informando a la aplicación si está o no un coche encima del IR, y en función de eso, que ésta abra la puerta o avise al usuario que no hay coche.

Para crear la aplicación, primero tendrán que diseñar la interfaz. Ésta contará con los siguientes elementos:

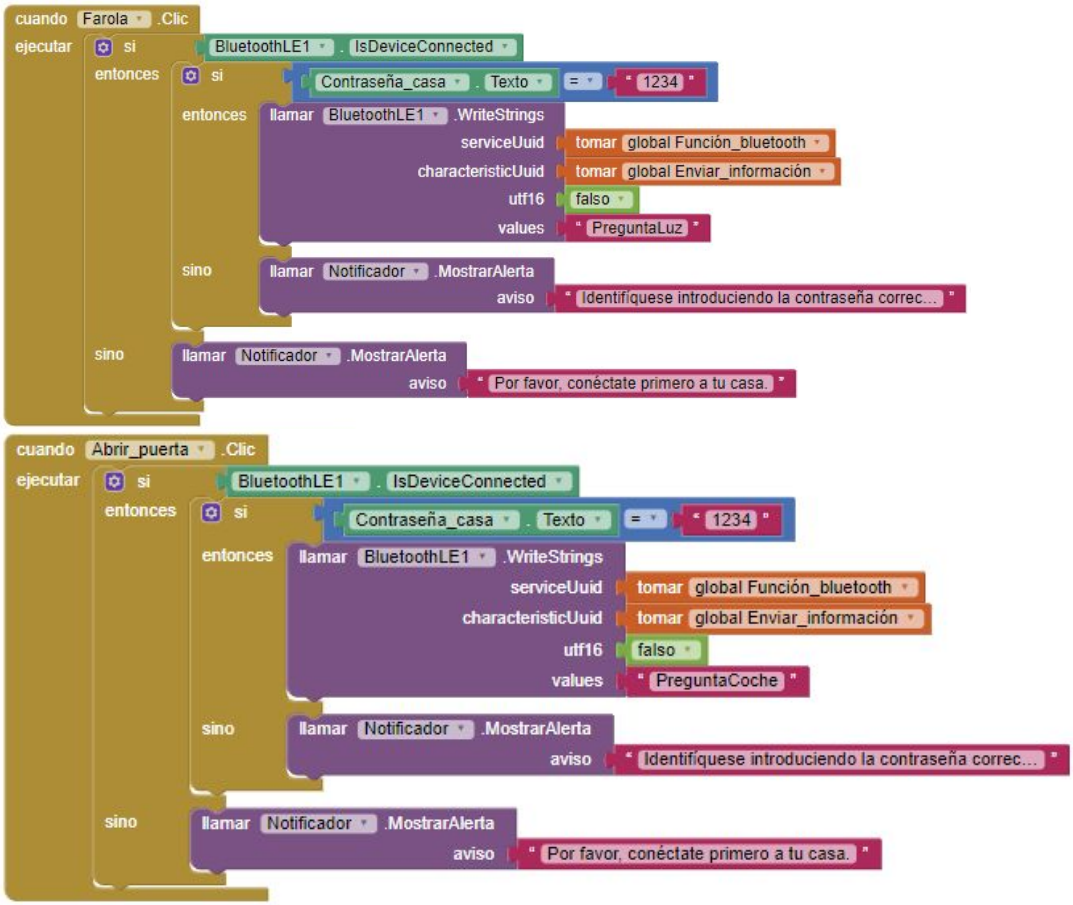
- **Campo de contraseña** para introducir la contraseña que permita manejar la aplicación.
- **Botón** para abrir la puerta del garaje.
- **Botón** que pregunte al usuario si quiere encender o apagar la farola.
- **Botone** que conecte el Bluetooth.
- **Botón** que desconecte el Bluetooth.
- ***VisorDeLista*** para elegir el Bluetooth y conectar el dispositivo con la placa controladora.

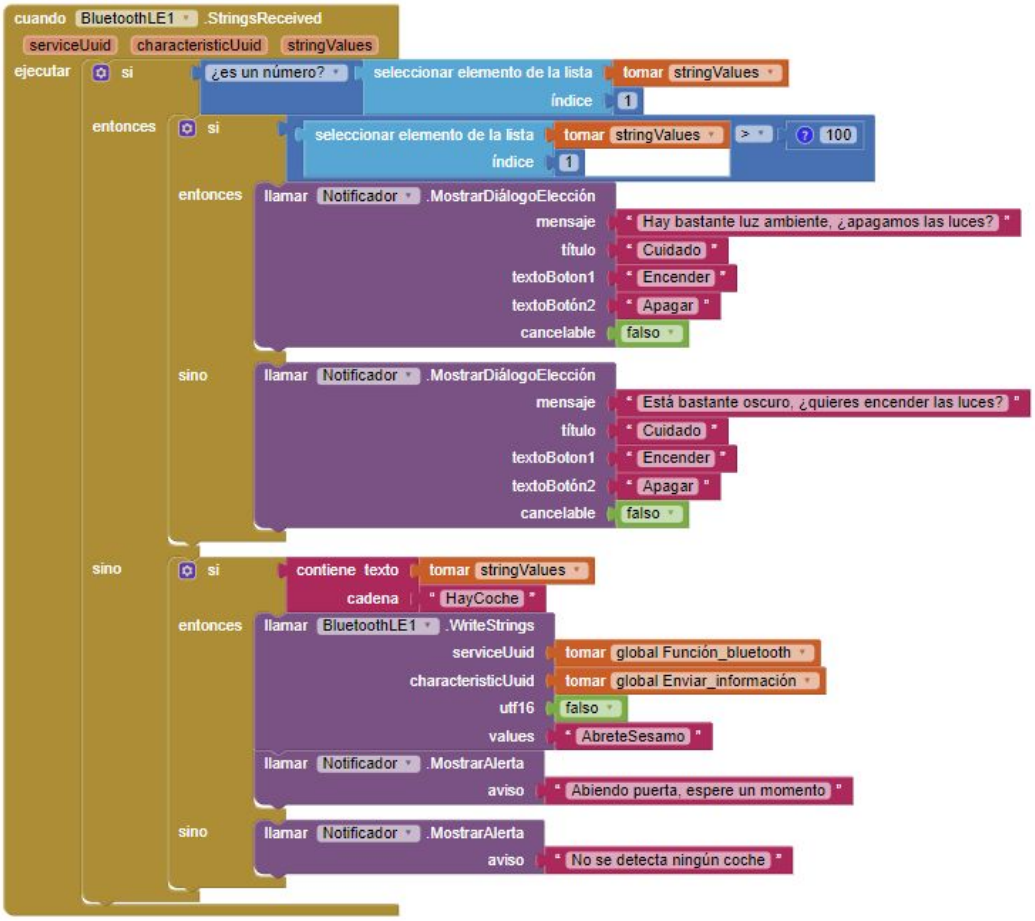


A continuación, deberán programar la aplicación. Para ello, primero habilitaremos la posibilidad de recibir información de la placa controladora a través del Bluetooth cuando éste se encuentre conectado, añadiendo el bloque *RegisterForStrings* en el evento *Bluetooth Conectado*.



A continuación, se deberán programar los botones de la Farola y Abrir_puerta.
En cuanto a la farola, le indicaremos al programa que cuando pulsemos el botón, le envíe el mensaje “PreguntaLuz” a la placa preguntándole por la cantidad de luz que hay.
En el caso de la puerta, le indicaremos al programa que cuando pulsemos el botón, le envíe el mensaje “PreguntaCoche” a la placa preguntándole si el sensor de infrarrojos está detectando un coche o no. En caso de que lo detectase, se enviará a la aplicación el mensaje “HayCoche” y en el caso contrario, el mensaje recibido será “Vacío”



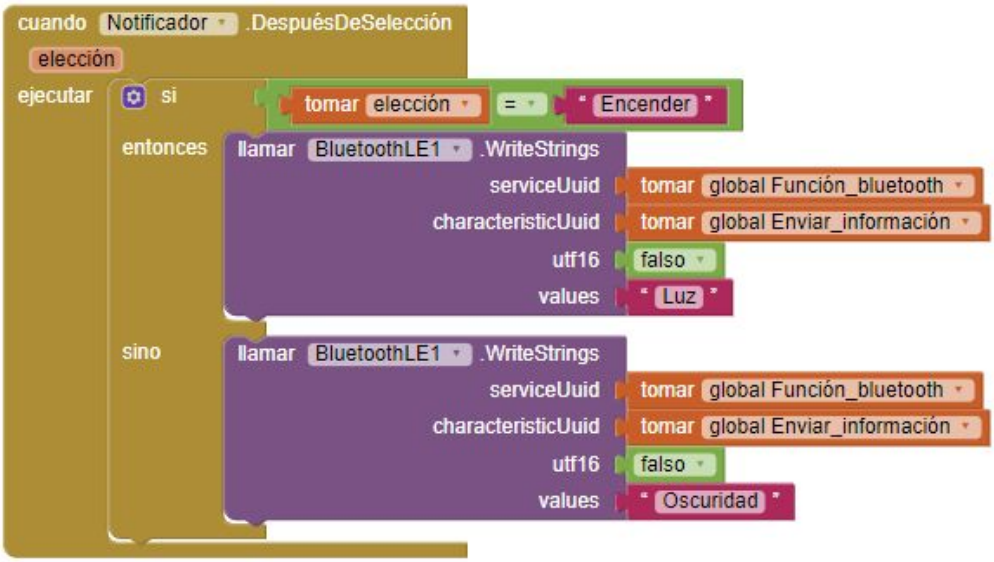


Cuando el Bluetooth reciba información, lo primero que hará será preguntar si el primer mensaje de la lista que recibe es un número. En caso de que así sea, lo que estamos recibiendo es la lectura del sensor de luz, de modo que llamaremos al Notificador para que, si el valor de luz es mayor de 100, le diga al usuario que hay bastante luz y que si quiere apagar las luces. Le dará dos opciones, *Encender* o *Apagar* la farola. Si la cantidad de luz es menor de 100, se le dirá al usuario que hay poca luz y que si quiere encender las luces, mostrándole igualmente las opciones de apagar o encender la farola.

La opción cancelable igual a *falso* indica que la pregunta que se le hace al usuario indicándole que si quiere encender o apagar la farola se debe contestar, es decir, que no se puede cancelar.

En caso que el valor recibido no corresponda con un número, preguntaremos si el mensaje contiene la palabra “HayCoche”. Si es así, mandaremos a la placa el mensaje “AbreteSesamo” que dará la orden de abrir la puerta, en caso contrario, aparecerá un mensaje en la pantalla informando al usuario que no se detecta ningún coche.

Ahora, indicaremos al programa qué tiene que hacer cuando pulsemos el botón farola y el usuario responda *Encender luces*. En este caso, mandaremos a la placa el mensaje *Luz* para que encienda el LED de la farola, y cuando reciba *Apagar*, mandaremos el mensaje *Oscuridad* para apagar el LED.



Continuaremos programando la placa controladora. Para ello, se deberá añadir un condicional que indique que si el mensaje recibido es “PreguntaLuz”, le envíe la cantidad de luz que está leyendo el LDR y otro condicional que compruebe que si el mensaje recibido por la aplicación es “PreguntaCoche”.

Se puede consultar la programación de todas estas mejoras en los archivos:
“RetoTech_P2_Casa_mejorada_ZumCore2.aia” y
“RetoTech_P2_Casa_mejorada_ZumCore2.bitbloq”

Temporalización sugerida:

1ª Sesión: Iniciación a App Inventor. Mi primera aplicación: el semáforo.

2ª Sesión: Control de la farola de la casa domótica. Encender y apagar un LED.

3ª Sesión: Control de la puerta automática. Apertura de la puerta con contraseña.

4ª Sesión: Control de la farola y la puerta con una sola aplicación. Introducción de mejoras.

//Contacto: Ayuda pedagógica:
retotech@fundacionendesa.org