Proyecto 2:
Controla tu casa domótica con **Bitbloq Apps**

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 Robotics
- Bitbloq Apps
- Dispositivo Android (tablet o móvil) o dispositivo Apple (solo para probar y previsualizar)

¿Qué es Bitbloq Apps?

Bitbloq Apps es una herramienta ubicada en Bitbloq, un entorno de desarrollo de software online y gratuito desarrollado por BQ Educación. Bitbloq Apps nos permite crear aplicaciones móviles para Android, y previsualizarlas desde iPhone, de forma visual y a partir de un conjunto de herramientas básicas con las que podemos enlazar una serie de bloques para crear la programación interna. De momento, **durante el curso 2022/2023 la creación de Apps móviles solo será posible con el Plan Docente de Bitbloq.**

Primeros pasos

Cuando estamos diseñando una aplicación, podemos visualizar en tiempo real cómo está quedando. Esto nos permite hacer pruebas sin necesidad de instalar la aplicación en el dispositivo.

Existen dos modos para probar la aplicación: *En tu dispositivo móvil* o *Previsualización web*.

Para probar la aplicación desde el dispositivo móvil, será necesaria la instalación de una app llamada Bitbloq Pocket disponible en las plataformas Google Play y Apple Store, dependiendo del sistema operativo del dispositivo que estemos utilizando.

Bitbloq pocket en la Play Store:

- https://play.google.com/store/apps/details?id=cc.bitbloq.mobile

Bitbloq pocket en la Apple Store:

- https://apps.apple.com/us/app/bitbloq-pocket/id1628061768

Importante: estos enlaces deben abrirse desde el dispositivo móvil, o bien buscar “Bitbloq Pocket” en el buscador de Google Play o Apple Store.

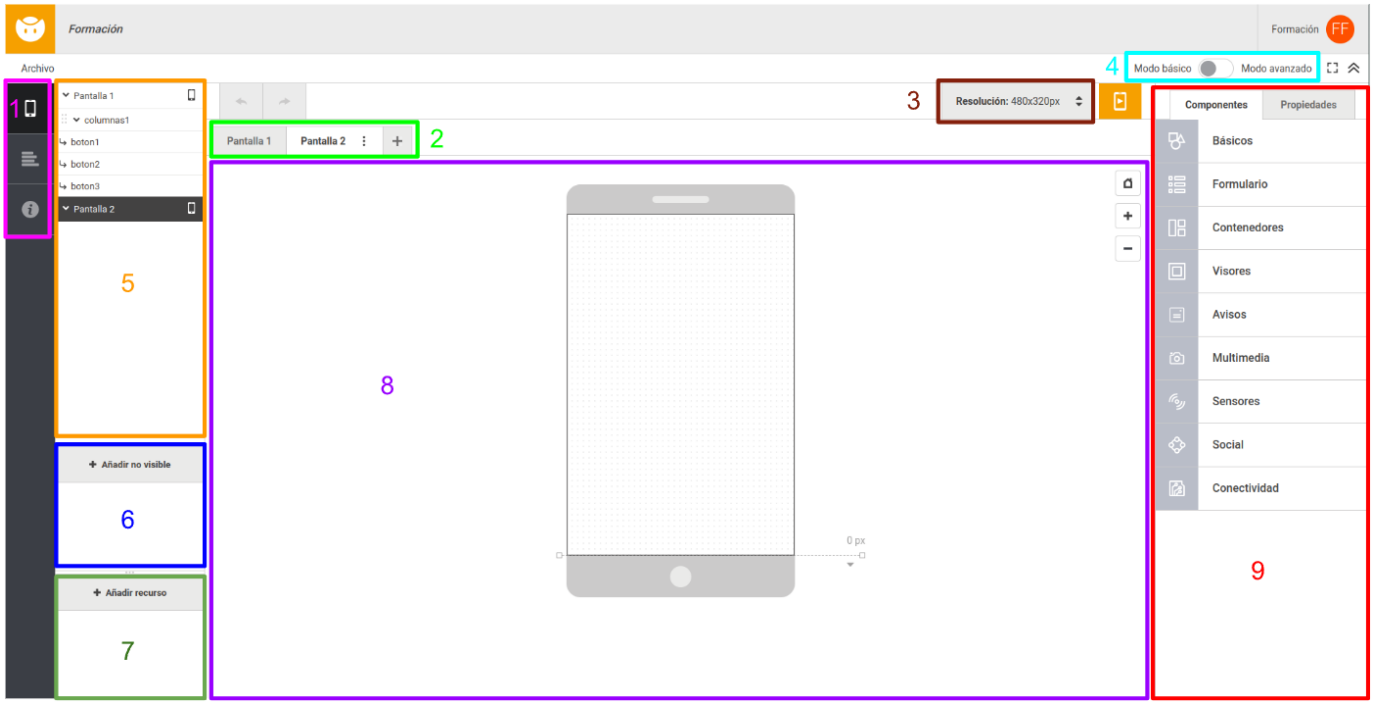
Controla tu casa domótica

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 la herramienta.

Para que los alumnos guarden sus proyectos sin necesidad de crearse una cuenta, podemos crear un aula y en ella un ejercicio en blanco de Bitbloq Apps para que puedan acceder a él. Es importante publicarlo para que puedan modificar este documento. En este documento se puede consultar cómo funciona Bitbloq Aulas.

Tras esto, les explicaremos las partes del programa.

Diseño



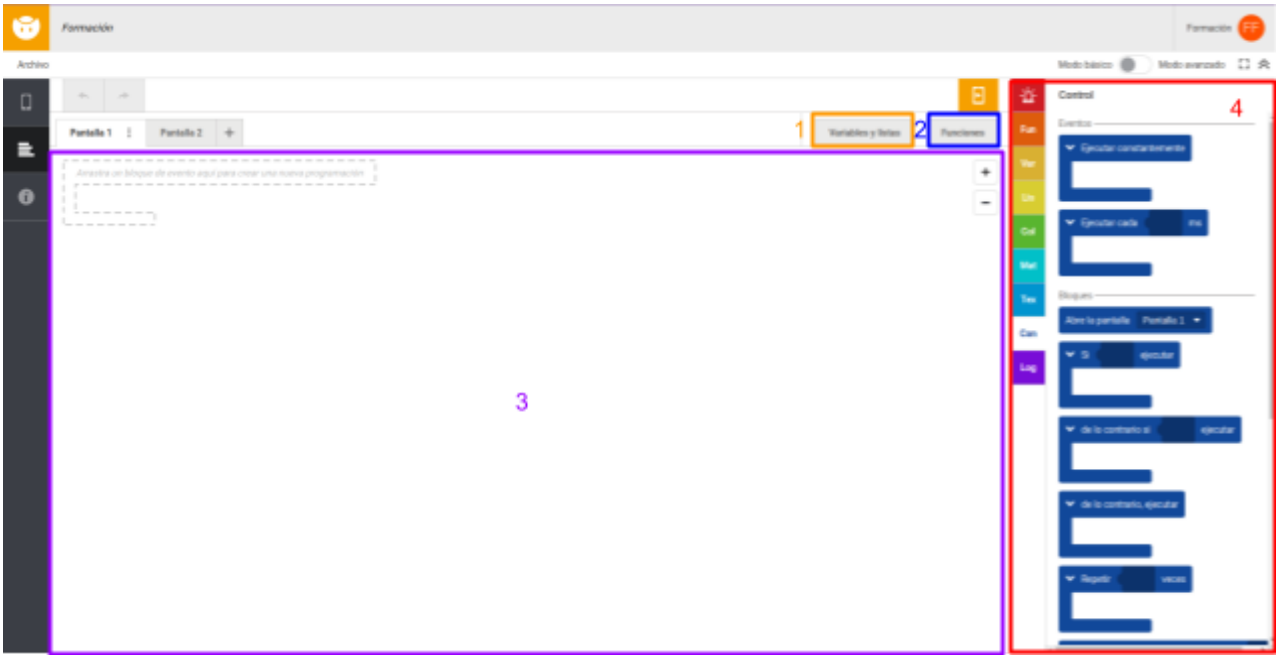
Bitbloq Apps se compone de los siguientes elementos:

1. ***Diseño/Bloques/Información del documento*** (interfaz/programación de la aplicación/información del documento): mediante estos botones podemos cambiar entre los tres apartados principales de la aplicación.
   - *Diseño*: en esta parte definiremos qué componentes del dispositivo usaremos y cómo va a ser visualmente la aplicación.
   - *Bloques*: en esta parte programaremos, mediante bloques, qué va a hacer la aplicación.

- *Información del documento*: en este apartado añadiremos la información del documento, además podremos añadir recursos para realizar el ejercicio.

2. ***Pantallas***: la aplicación puede diseñarse en una sola pantalla o puede tener varias. Mediante estos botones, podemos navegar entre pantallas o añadir más.
3. ***Resolución***: en esta opción se puede cambiar la resolución del dispositivo que se usa como guía para crear la aplicación.
4. ***Modo básico/avanzado***: en esta sección podemos cambiar el modo de uso de la herramienta. En el modo básico encontramos las opciones más sencillas para configurar y programar la aplicación y en el modo avanzado se amplían las opciones.
5. ***Panel de componentes visibles***: muestra el listado de los componentes que hayamos incluido en la aplicación dividido en pantallas. Podemos organizar los componentes para superponer unos a otros según cuál esté en la parte superior de este panel y ver qué componentes están dentro de contenedores.
6. ***Panel de componentes no visibles***: son componentes no visibles que sirven para que la aplicación use avisos, material multimedia, sensores del dispositivo, herramientas sociales y conectividad.
7. ***Panel de recursos***: en esta sección podemos subir imágenes, vídeos, etc. para incorporarlos en la aplicación.
8. ***Interfaz***: en esta área podemos colocar lo que queremos que tenga la aplicación, además podemos ampliar la pantalla añadiendo scroll.
9. ***Componentes/Propiedades***: en esta sección hay dos pestañas.
    a. En la pestaña componente encontramos los componentes que podemos añadir a nuestra aplicación clasificados en varias categorías (*Básicos*, *Formulario*, *Contenedores*, *Visores*, *Avisos*, *Multimedia*, *Sensores*, *Social* y *Conectividad*).
    b. En la pestaña propiedades podremos cambiar las propiedades de los componentes (color, tamaño del texto, ubicación, etc.). Las propiedades serán diferentes según el componente seleccionado.

Bloques



Aquí podemos programar lo que hay detrás de una aplicación para que funcione. Es importante distinguir entre el diseño de la aplicación y su programación.
Se compone de los siguientes elementos:

1. ***Variables y listas***: seleccionando esta pestaña podemos declarar y programar variables y listas arrastrando bloques al área de programación.
2. ***Funciones***: en esta pestaña podemos programar funciones arrastrando bloques al área de programación.
3. ***Área de programación***: en este espacio podemos colocar los bloques de programación arrastrándolos desde la zona *Bloques de programación*.
4. ***Bloques de programación***: en esta sección encontramos, clasificados en categorías, los bloques para programar la aplicación.

Veamos una pequeña descripción de cada categoría:

- ***Componentes***: Al añadir componentes en la interfaz del apartado *Diseño*, aparecen bloques específicos de esos componentes en esta categoría.
- ***Funciones***: para declarar y usar funciones.
- ***Variables***: para declarar y usar variables.
- ***Listas***: para crear y gestionar listas de elementos.
- ***Color***: para asignar y modificar colores.
- ***Matemáticas***: para trabajar con números y hacer operaciones matemáticas.
- ***Texto***: para usar y trabajar con cadenas de caracteres.
- ***Control***: bloques de control de programa: if, for, eventos, etc.
- ***Lógica***: para hacer comparaciones lógicas y matemáticas.

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 a la zona de programación.

Es importante explicar que mientras que Bitbloq Robotics es un programa secuencial, es decir, la programación se lee de arriba a abajo y hasta que no finaliza una acción no se ejecuta la siguiente, Bitbloq Apps es **por eventos**, es decir, **podemos tener varias acciones que sucedan a la vez.**

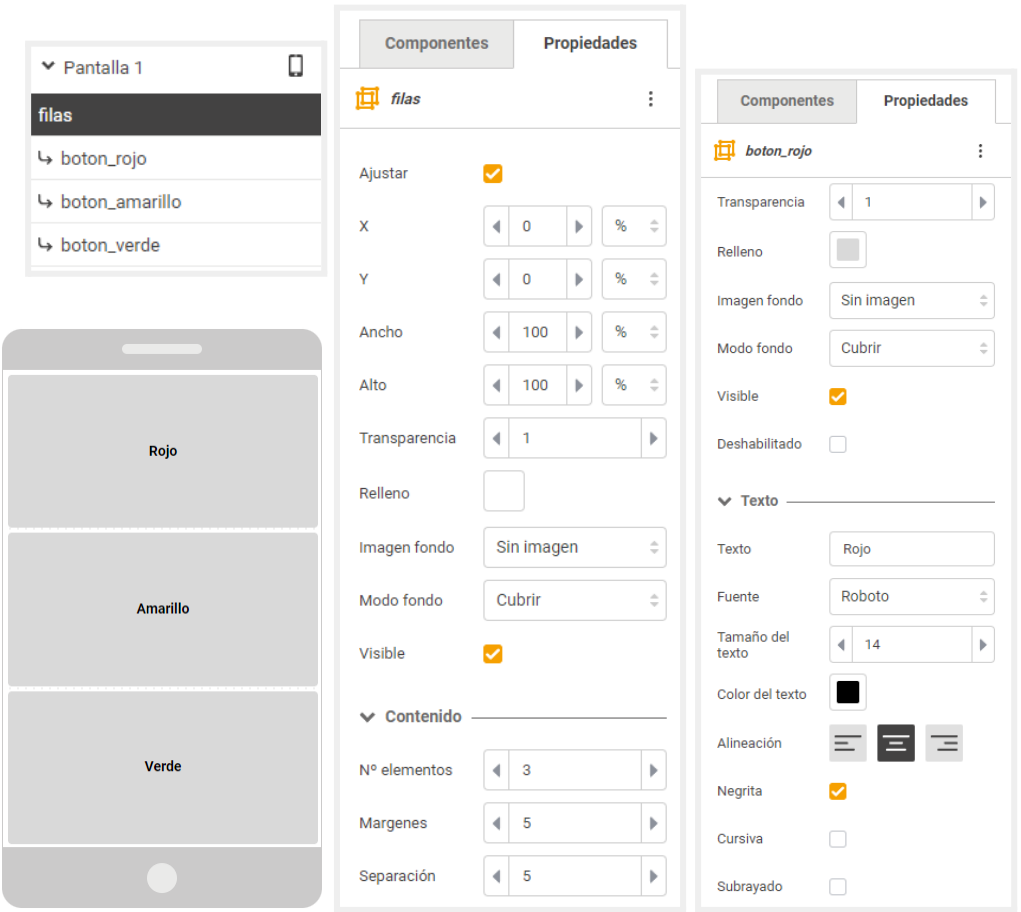
Los bloques de eventos son los primeros que podemos encontrar en la categoría componentes, y son los que iniciarán las instrucciones programadas.

Podemos organizarlos por columnas para mantener nuestra programación ordenada, sin que esto afecte al orden de ejecución de las instrucciones.

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:



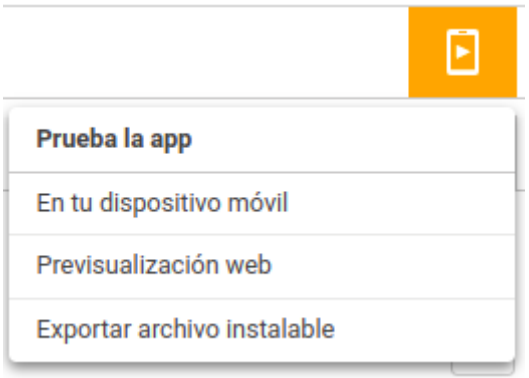
En el ejemplo, vemos que hemos colocado un contenedor con tres elementos, el cual hemos ajustado a la pantalla del dispositivo. En cada elemento hemos puesto un botón, al que hemos cambiado las propiedades de color de fondo y de texto. Como podemos ver en el árbol de componentes, los hemos **renombrado** todos, lo cual recomendamos hacer siempre para tener controlados los componentes que vamos a programar.

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 gris.**

La programación que deberán crear es la siguiente:

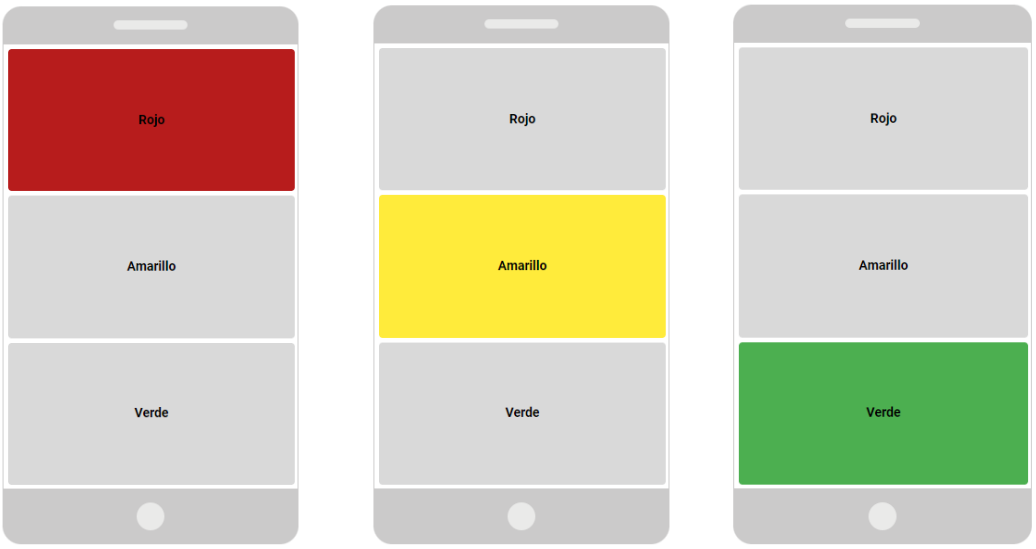
¿Cómo visualizar la aplicación en nuestro dispositivo?

Para visualizar la aplicación deberemos conectar el programa con el móvil. Se podrán utilizar tres opciones: previsualizar en el móvil (con Bitbloq Pocket), previsualizar en la web o bien crear una APK para instalar en el dispositivo (solo para Android).



Para previsualizarlo en el móvil, habrá que hacer clic sobre *En tu dispositivo móvil*, abrir Bitbloq Pocket y escanear el código QR o introducir el código de texto:

La aplicación será algo similar a lo siguiente:



Podemos ver este programa en el archivo *Reto _2_ App Semáforo.bitbloq*

Controlando la farola inteligente

Una vez que han creado su primera aplicación, comenzaremos con la integración de Bitbloq Apps 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 móvil.

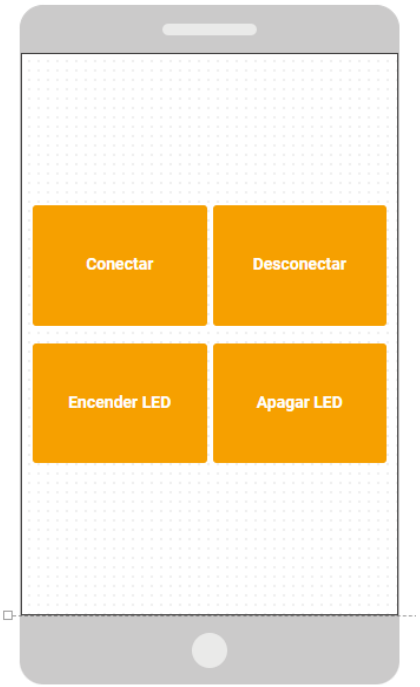
Programación en Bitbloq Apps

Comenzaremos abriendo un nuevo proyecto y crearemos la interfaz de la aplicación y la programación en Bitbloq Apps. La interfaz tendrá los siguientes elementos:

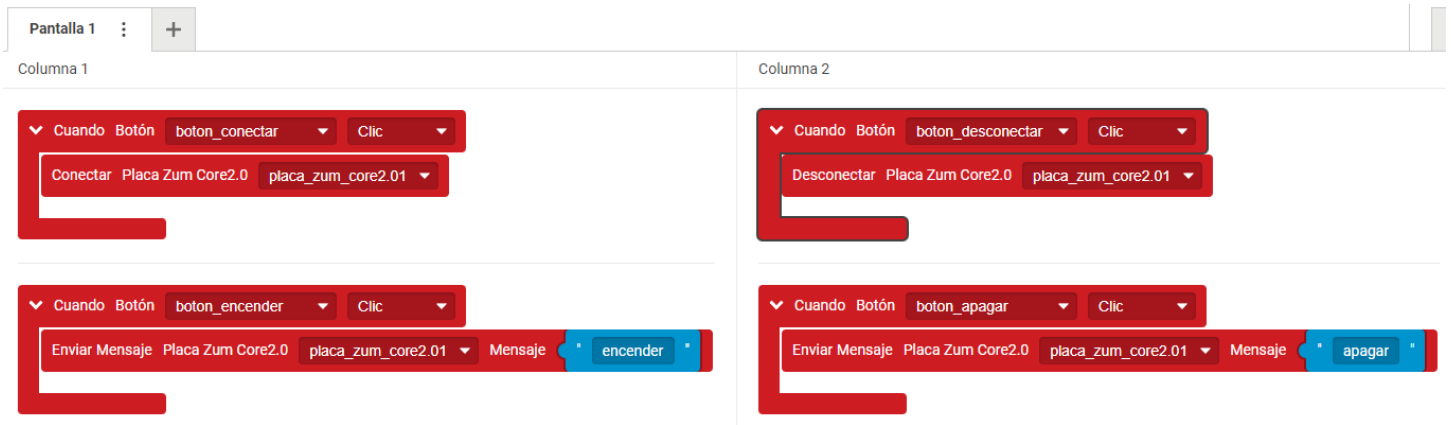
- Dos *botones*, uno para buscar las placas Zum Core 2.0 disponibles, al que llamaremos “Conectar”, y otro para desconectar el dispositivo.
- Dos *botones* más para encender y apagar el LED.

Además, en la interfaz tendremos que añadir, desde *Conectividad*, el componente de la placa correspondiente, en este caso el de la Zum Core 2.0. Este elemento no será visible en la interfaz de la aplicación.

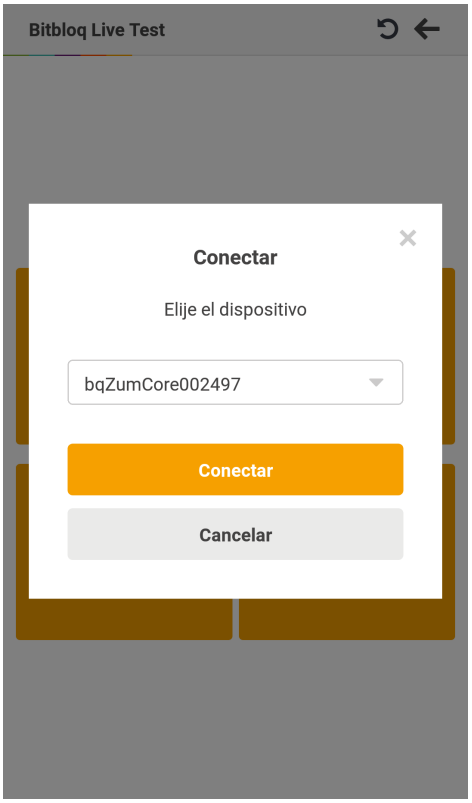
Un ejemplo sencillo es el siguiente:

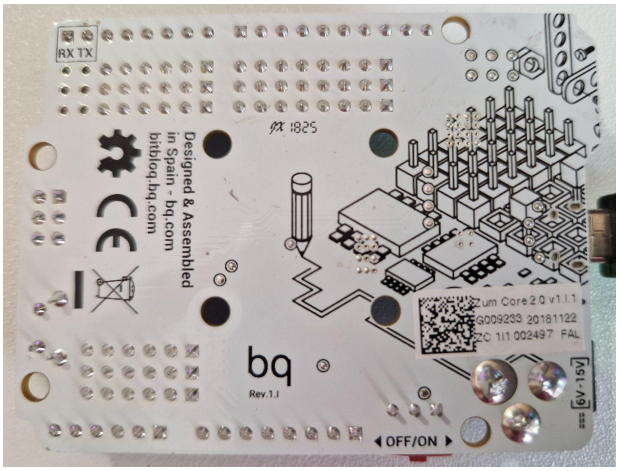


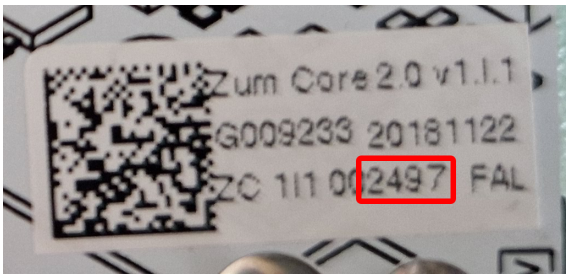
A continuación, programaremos qué funciones queremos que tenga cada botón. Para ello, en el apartado *Bloques* comenzaremos poniendo los bloques correspondientes para la conexión bluetooth con la placa controladora.



Con esta programación, al hacer clic en *Conectar* se abrirá un cuadro de diálogo con un desplegable que nos mostrará todas las placas Zum Core 2.0 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:







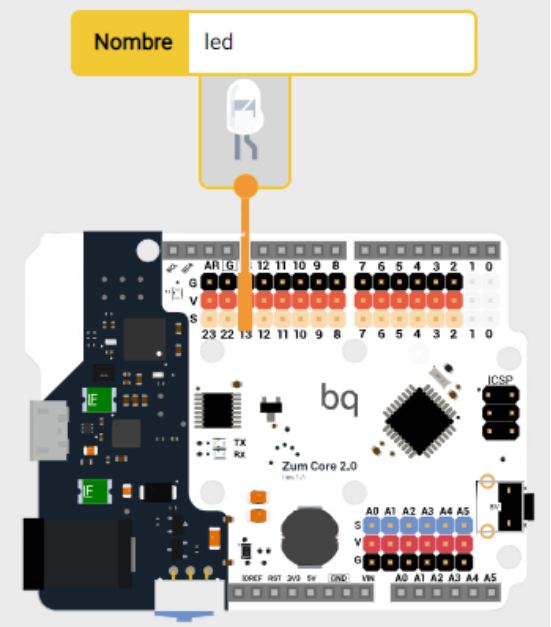
Tras seleccionar la placa correspondiente y hacer clic en *Conectar*, se nos notificará si la conexión se ha realizado con éxito. De ser así, un LED azul parpadeará en nuestra placa. También se nos notificará en caso de que la placa se desconecte.

Además, hemos programado que, al pulsar el botón *Encender* o *Apagar,* el programa mande un mensaje a nuestra placa. Será así, mediante mensajes, como podamos comunicarnos con ella para que ejecute las acciones que indiquemos en Bitbloq Robotics.

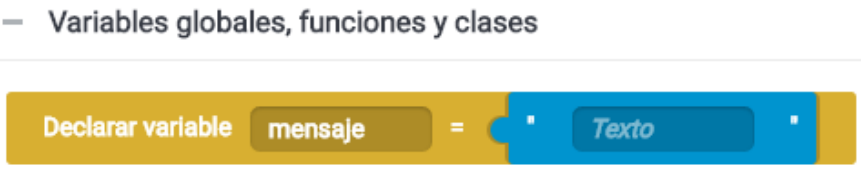
Programación en Bitbloq Robotics

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

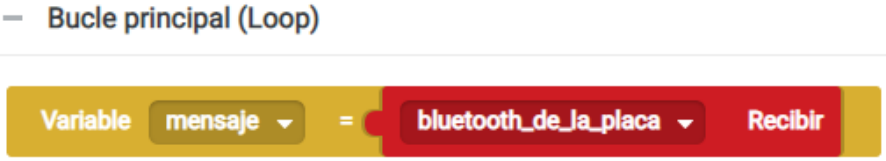
Para ello, primero conectaremos en la parte de *Hardware* los componentes que vamos a utilizar.



A continuación, iremos a la pestaña *Software* y crearemos en el apartado *Variables globales y funciones* una variable a la que llamaremos “mensaje” que sea igual a un texto vacío.



En esta variable guardaremos los mensajes que sean enviados por la aplicación. Para ello, en la parte *Bucle principal (Loop)* guardaremos en la variable 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. Indicaremos que, si la variable mensaje es igual a “encender” (que es el mensaje que pusimos en Bitbloq Apps) se encienda el LED. Añadiremos también el otro mensaje, es decir, que si la variable es igual a “apagar” se apague el LED.

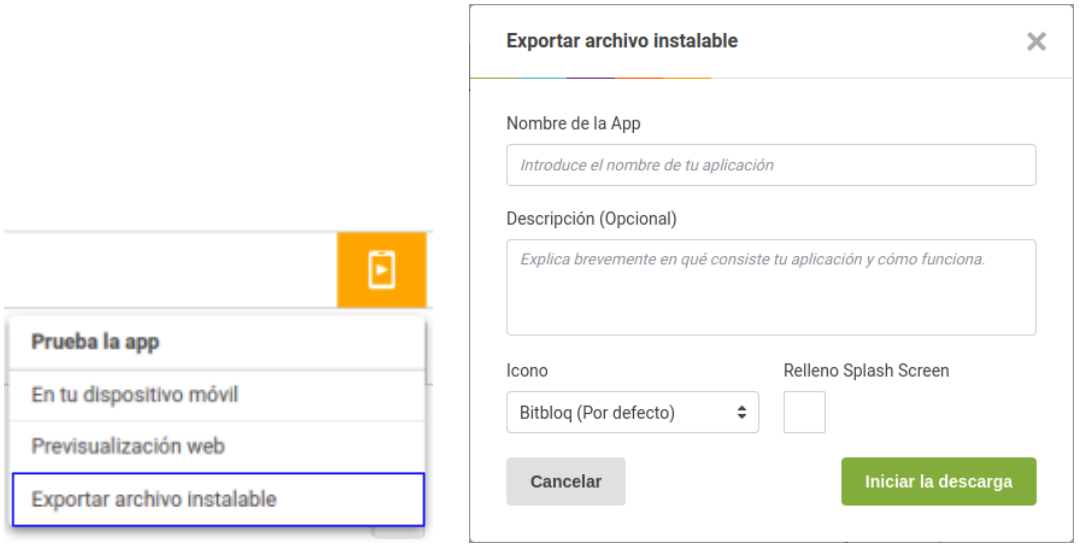
Hay que tener mucho cuidado a la hora de poner el mensaje, ya que éste deberá ser igual que el que enviamos desde la aplicación (signos de puntuación, mayúsculas, minúsculas…).

Para poder comprobar que todo funciona correctamente, cargamos el programa a la placa controladora, abrimos Bitbloq Apps y conectamos la aplicación con la placa. Al pulsar el botón *encender* debería encenderse el LED, y apagarse al pulsar el botón *apagar*.

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

Una vez que hemos terminado de modificar la aplicación a nuestro gusto, podemos compilarla y descargarla para instalarla en nuestro dispositivo.

Para descargar la aplicación creada pulsamos el botón *Prueba la app*  y seleccionamos la opción *Exportar archivo instalable*:



Rellenamos los datos que nos piden:

- *Nombre de la App*: este nombre será el de la aplicación, se verá cuando esté instalada en el dispositivo.
- *Descripción*: podemos añadir una descripción sobre lo que se puede hacer con la aplicación.
- *Icono*: podemos añadir un icono a la aplicación, se verá cuando esté instalada en el dispositivo.
- *Relleno Splash Screen*: podemos añadir color a la pantalla que sale al iniciar la aplicación en el dispositivo.

Pulsamos *Iniciar la descarga* para generar el archivo instalable.

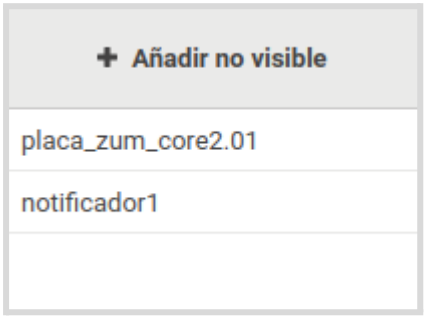
Una vez se ha generado el archivo nos da la opción de escanear un código QR para descargarlo en un dispositivo o descargarlo en el ordenador pulsando el botón *Descargar*.

Podemos ver estos programas en los archivos *Reto_2_farola_apps.bitbloq* y *Reto_2_farola_robotics.bitbloq*

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. Además, hasta que el coche no entre dentro del garaje, la puerta no se cerrará.**

Comenzaremos diseñando la interfaz. Ésta deberá contar, al igual que en el programa anterior, con los botones *Conectar* y *Desconectar* 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 *Placa Zum Core 2.0* y *Notificador.*



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

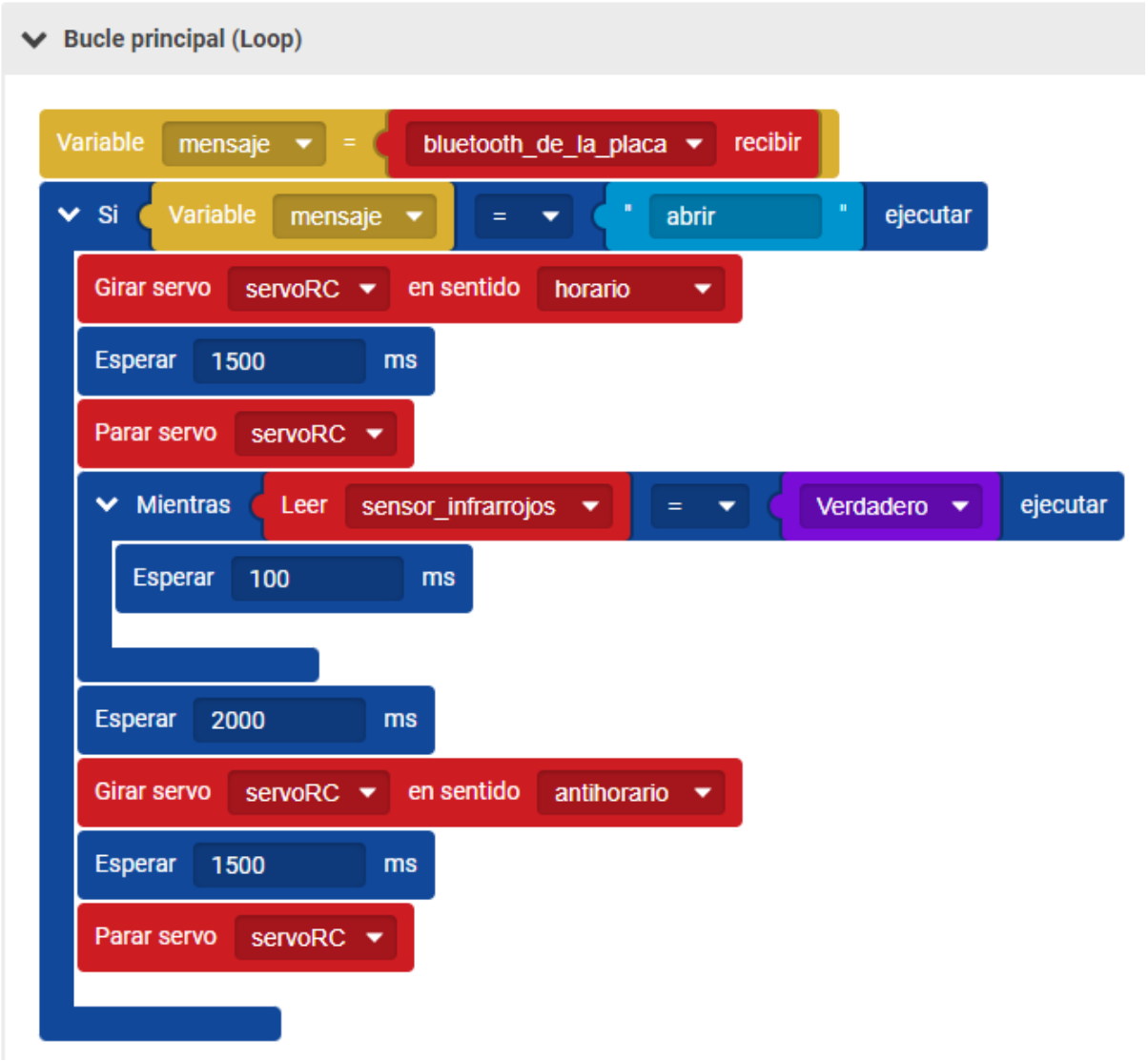
Para completar la aplicación, tendremos que programar qué tiene que hacer el botón de *Abrir puerta*. En este caso, al hacer clic sobre él, enviará un mensaje a la placa para que esta ejecute una acción, que tendremos que programar en Bitbloq Robotics. Podemos añadir un aviso para informar al usuario 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 Robotics 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 (para abrir la puerta) y el sensor de infrarrojos (para no cerrar la puerta *mientras* se detecta al coche).

En *Bloques*, tendremos que declarar en *Variables globales y funciones* una variable que sea igual a un texto vacío.

Tras esto, iremos a la parte de *Bucle principal (Loop)* y encima de la programación guardaremos en la variable que hemos creado lo que reciba del bluetooth. Por último, incluimos la programación de la puerta dentro de un condicional que compruebe que si el mensaje recibido es el mismo (en nuestro caso era “abrir”), 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 Bitbloq Apps y conectar el dispositivo con la placa.

Podemos ver este programa en los archivos *Reto_2_puerta_apps.bitbloq* y *Reto_2_puerta_robotics.bitbloq*.

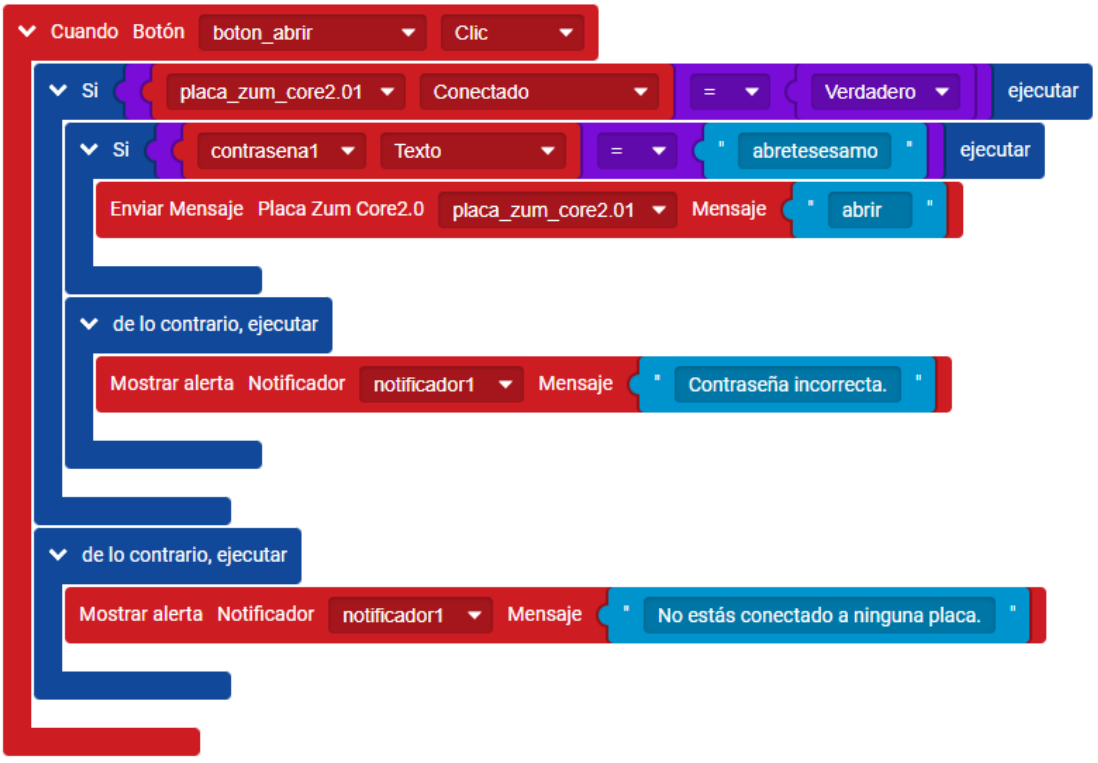
Como reto extra, podemos proponerles que intenten crear una contraseña para abrir la puerta, de esta forma no se abrirá hasta que no se haya introducido la contraseña correcta.

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

Modificaremos la interfaz de la aplicación y añadiremos un componente *Entrada de contraseña* para poder introducir la clave.

Por defecto, aparecerá la palabra “Contraseña” como pista de nuestro componente *Entrada de contraseña*. Podemos modificarlo en las propiedades del componente si así lo deseamos.

Por último, tendrán que ir al apartado *Bloques* y modificar lo que hace el botón de abrir puerta, añadiendo dentro un condicional que indique que **si el texto de la contraseña introducida es igual a la contraseña que queramos, por ejemplo, “abretesesamo”, abra la puerta, de lo contrario, que nuestro notificador nos muestre una alerta de que no es la contraseña correcta.**



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

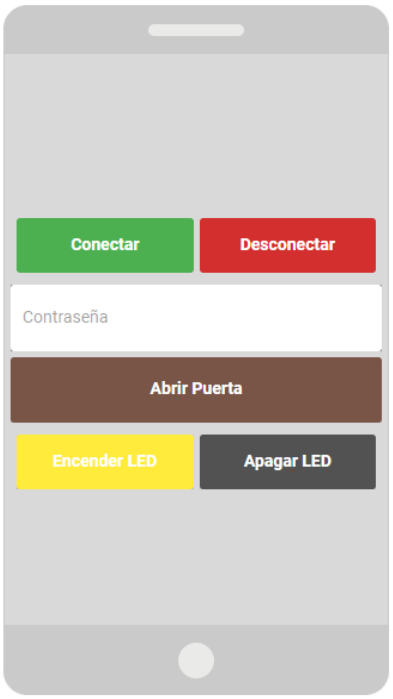
Podemos ver el programa de ejemplo en el archivo:
Reto_2_puerta_contraseña_apps.bitbloq

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 del archivo 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 8 elementos:

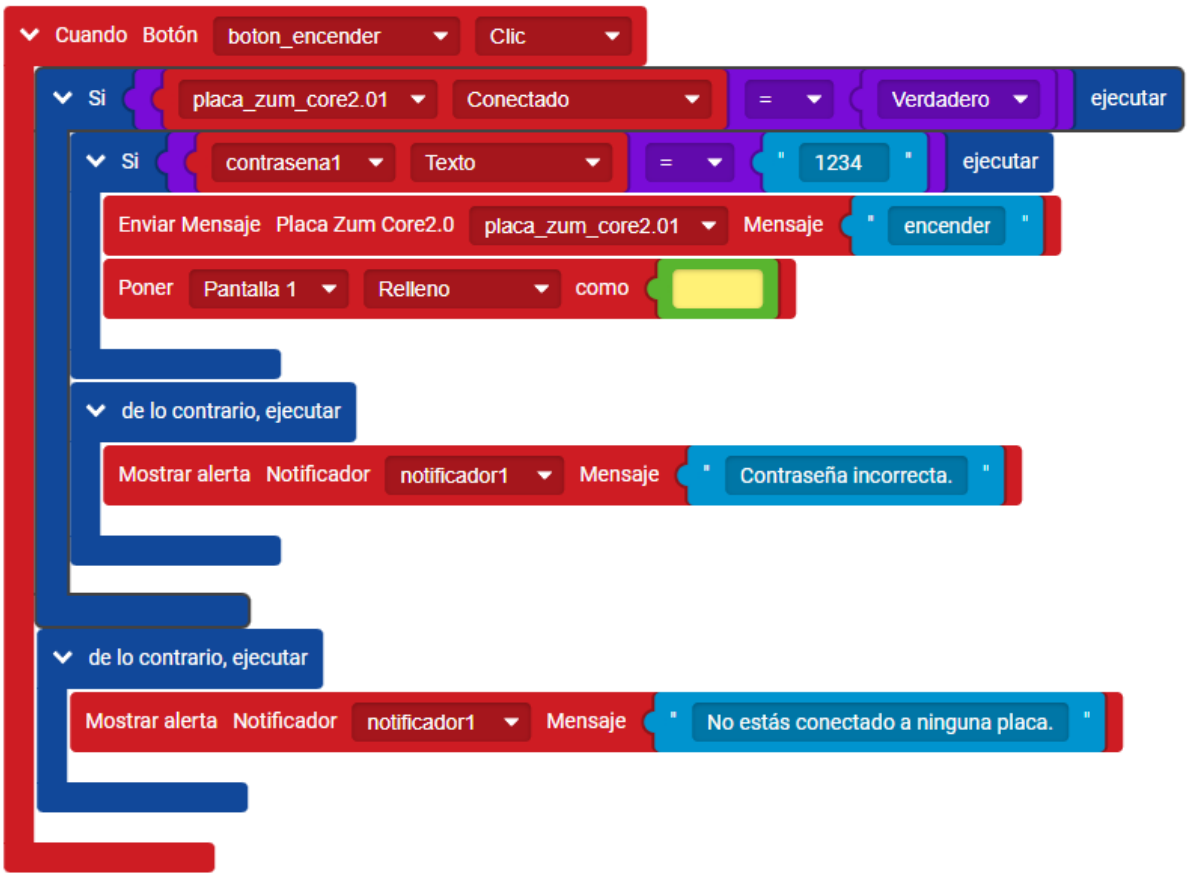


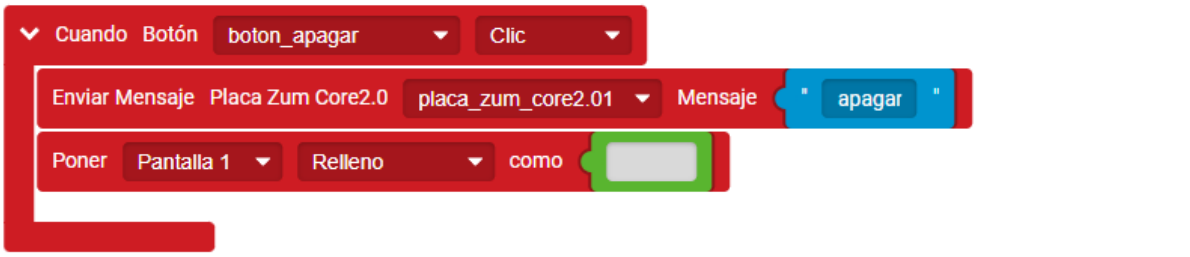
- **Entrada de contraseña:** para proteger mediante una contraseña la aplicación.
- **Cinco botones:** uno para conectar y otro para desconectar el Bluetooth, uno para encender la farola, otro para apagarla y un último para abrir la puerta automática.
- Dos elementos no visibles: el **notificador** y la placa **Zum Core 2.0**.

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.

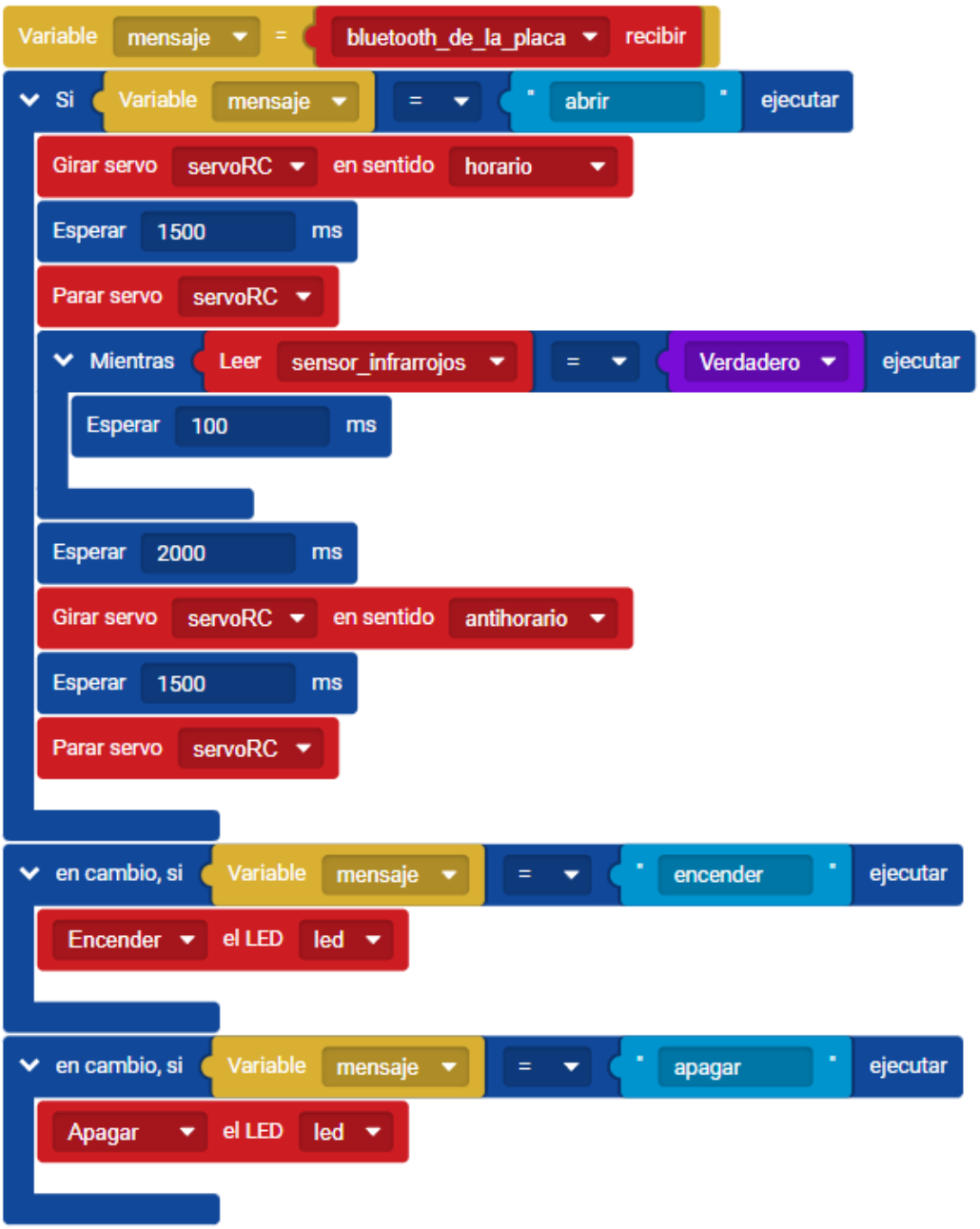
Para crearla, añadirán un condicional que indique que si la contraseña introducida es la correcta, se pueda pulsar el botón que enciende la farola, pero si no es la correcta, la aplicación le diga al usuario que introduzca la clave.

Además, hemos añadido el detalle de cambiar el color de relleno de la pantalla, dependiendo de si la luz está encendida o apagada. En Bitbloq Apps quedaría algo similar a lo siguiente:





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

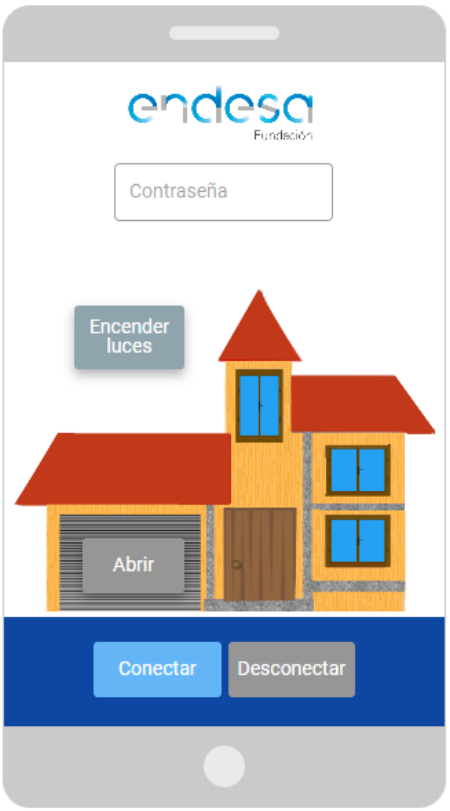


La programación completa se puede ver en el archivo *Reto_2_casa_domotica_sin_decoración.bitbloq* y en *Reto_2_casa_domotica_1_robotics.bitbloq*

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. Podemos sugerir a nuestros alumnos incorporar elementos decorativos como imágenes, colores, etc.

Un ejemplo de interfaz es el siguiente:



Para diseñar la interfaz se han añadido como recurso varias fotos de la casa (con luces encendidas o apagadas) 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 luces* se iluminen las ventanas. Para hacer esto, se han subido dos imagenes de la casa y se ha programado que cuando se pulse en *Encender luces* se muestre la imagen con las ventanas de color amarillo, y si se pulsa en *Apagar luces* se cambie a la imagen que tiene las ventanas sin iluminar.
- Además, se ha añadido que al pulsar el botón de *Encender luces*, el botón de apagar se esconda y viceversa. Para ello, se han utilizado los bloques de lógica *falso* y *verdadero*, de manera que cuando se pulse el botón *Encender luz*, éste se muestre visible (verdadero) y el botón de *Apagar luz* se esconda (falso).

Luces


Cuando Botón boton_encender Clic
Si placa_zum_core2.01 Conectado = Verdadero ejecutar
Si contrasena1 Texto = 1234 ejecutar
Enviar Mensaje Placa Zum Core2.0 placa_zum_core2.01 Mensaje encender
Poner Imagen imagen_casa Imagen fondo como Casa_Endesa_luz.png
Poner Botón boton_encender Mostrar como Falso
Poner Botón boton_apagar Mostrar como Verdadero
de lo contrario, ejecutar
Mostrar alerta Notificador notificador1 Mensaje Contraseña incorrecta
de lo contrario, ejecutar
Mostrar alerta Notificador notificador1 Mensaje No estás conectado a ninguna placa.

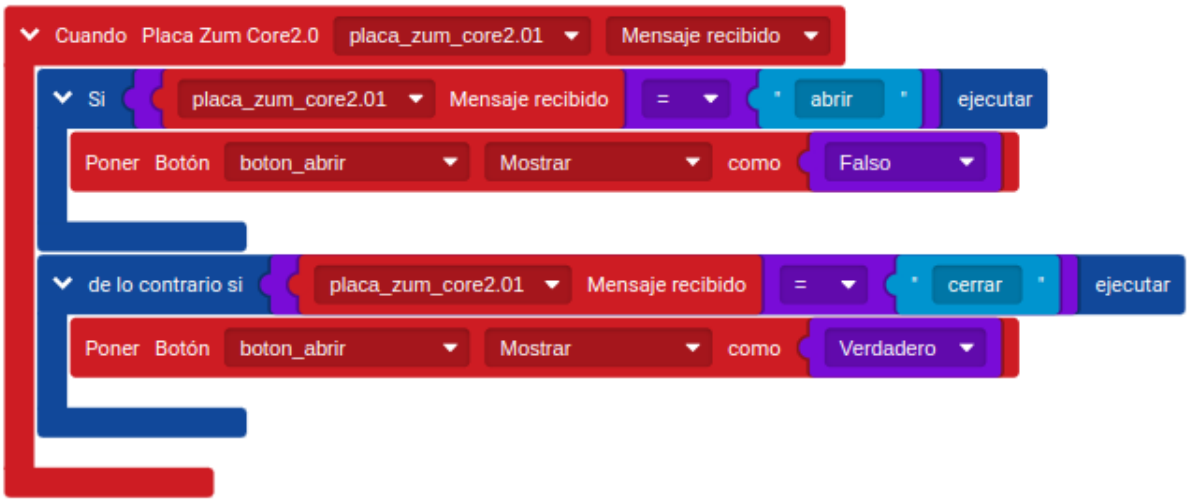

Cuando Botón boton_apagar Clic
Si placa_zum_core2.01 Conectado = Verdadero ejecutar
Si contrasena1 Texto = 1234 ejecutar
Enviar Mensaje Placa Zum Core2.0 placa_zum_core2.01 Mensaje apagar
Poner Imagen imagen_casa Imagen fondo como Casa_Endesa.png
Poner Botón boton_encender Mostrar como Verdadero
Poner Botón boton_apagar Mostrar como Falso
de lo contrario, ejecutar
Mostrar alerta Notificador notificador1 Mensaje Contraseña incorrecta
de lo contrario, ejecutar
Mostrar alerta Notificador notificador1 Mensaje No estás conectado a ninguna placa.

- Tal y como se muestra en las imagenes de programación, para cambiar el fondo de componente *imagen_casa*, se deberá añadir el bloque *Poner imagen - imagen fondo como* seguido del bloque de recurso que encontraremos en *Parámetros* con el nombre de la imagen y la extensión: "*Casa_Endesa.png*" o "*Casa_Endesa_luz.png*".
- Se ha programado que cuando la app se conecta a la placa cambie el botón de *Conectar* de azul a verde, para saber en qué momento la placa está conectada y cuándo no. Para ello, habrá que programar que al conectarse la placa el relleno del botón cambie, y que vuelva a azul cuando se desconecte:

- **Como funcionalidad extra**, hemos añadido **enviar un mensaje desde la placa a nuestra aplicación**, para que cuando la puerta esté abierta el botón no aparezca, y cuando esté cerrada aparezca para poder volver a abrirla. Cuando la puerta esté abierta mandaremos el mensaje *abrir* para que el *botón abrir* permanezca oculto hasta que la puerta vuelva a cerrarse. En ese momento se enviará el mensaje *cerrar* a nuestra app, y esto hará que el *botón abrir* vuelva a estar visible. Así quedaría la programación en Bitbloq Robotics:

Bucle principal (Loop)

En Bitbloq Apps, programaremos que dependiendo del mensaje que reciba de la placa se realice una acción u otra, de esta manera:



Podemos ver cómo programar y añadir una interfaz como la mostrada anteriormente, en el archivo *Reto_2_Casa_domotica_apps.bitbloq* y la programación de Bitbloq Robotics en *Reto_2_casa_domotica_2_robotics.bitbloq*.

Recepción de mensajes enviados por la placa

Una vez que han terminado el programa anterior, podemos plantear a los alumnos **retos extras**.

En varios casos, necesitaremos hacer la comunicación con la placa en el sentido inverso a como lo hemos hecho hasta ahora, es decir, en vez de enviar mensajes a la placa, **queremos recibir mensajes en nuestro dispositivo**, como ya hicimos anteriormente con el sensor infrarrojo.

Para ello, haremos un pequeño ejemplo, en un proyecto nuevo, en el que recibiremos los datos de la lectura de un sensor de luz conectado a la placa controladora.

Comenzaremos diseñando la interfaz, que ha de contener los siguientes elementos:

- *Botón* para conectar la placa.
- *Botón* para desconectar la placa.
- *Botón* para pedir que la placa controladora nos envíe el valor del sensor de luz.
- *Texto* donde explique que a la derecha se verá la luminosidad.
- *Texto* donde quedará reflejado el valor del sensor de luz.
- *Texto* que indica si la placa está o no conectada.
- Componente no visible *Placa Zum Core 2.0*.

Programación en Bitbloq Apps

Como siempre, primero programaremos el procedimiento de conexión con el bluetooth de la placa mediante dos botones:

Además, le diremos al componente *texto* que nos muestre el estado de nuestra placa para saber si está o no conectada:

A continuación, programaremos lo que va a realizar el botón *Leer LDR* al ser pulsado. Al igual que en casos anteriores, lo que hará este botón será mandar un mensaje desde la aplicación hasta la placa controladora, cuyo contenido, en este caso, será la palabra *luz*.

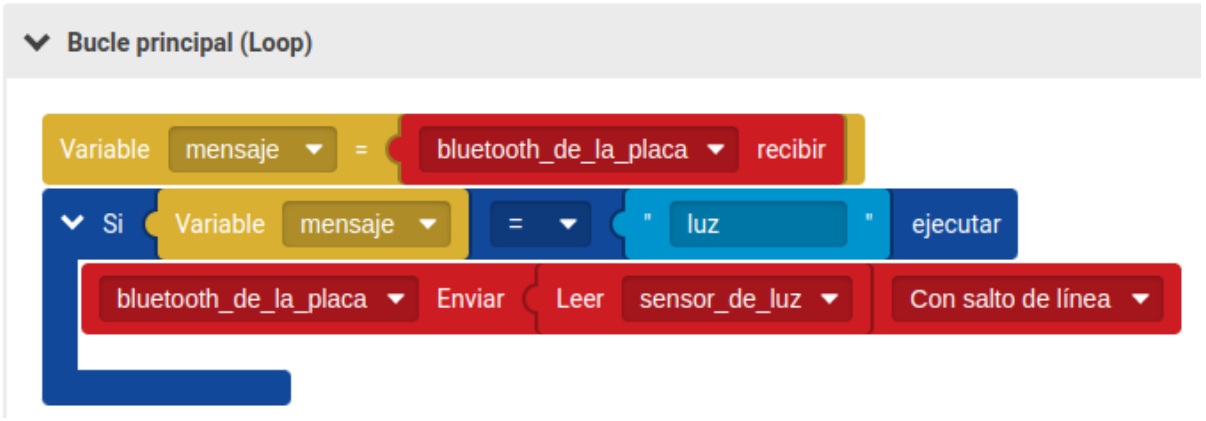
De esta forma, obtendremos el valor de la luminosidad medida por el LDR de nuestra placa en el momento en el que pulsamos el botón.

IMPORTANTE: para que la lectura de mensajes recibidos desde la placa funcione correctamente, las acciones asociadas a dichos mensajes deben estar dentro del evento *Cuando Placa Zum Core2.0 mensaje recibido*. Si de lo contrario se programan en secuencias de otros eventos, la lectura de los mismos no será instantánea.

Programación en Bitbloq Robotics

Primero, tenemos que declarar una variable de tipo texto en la zona de *Variables globales y funciones*.

A continuación, indicaremos en el bucle principal que la variable guarde todo lo que recibe el bluetooth de la placa (recibir mensajes), además de programar un condicional que lea el mensaje "luz" recibido desde la app y ejecute enviar los valores del sensor de luz.



Para poder comprobar que todo funciona, deberemos cargar el programa a la placa controladora, abrir Bitbloq Pocket y conectar el dispositivo a la placa.

Podemos ver cómo programar y añadir una interfaz como la mostrada anteriormente, en el archivo de Bitbloq Apps *Lectura LDR_boton_Apps.bitbloq*, y la programación de Bitbloq Robotics en *Lectur_LDR_Robotics.bitbloq*.

Añadiendo mejoras a nuestra aplicación

Una vez que hemos visto cómo poder mandar y cómo poder recibir mensajes entre el dispositivo y nuestra placa controladora, vamos a plantearles a nuestros alumnos dos mejoras para nuestra app:

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 envíe un mensaje a la aplicación que indique la cantidad de luz que hay en el lugar y 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. **Informar a la aplicación la presencia o no de un coche encima del IR** y, en función de eso, que ésta abra la puerta o avise al usuario de que no hay coche.

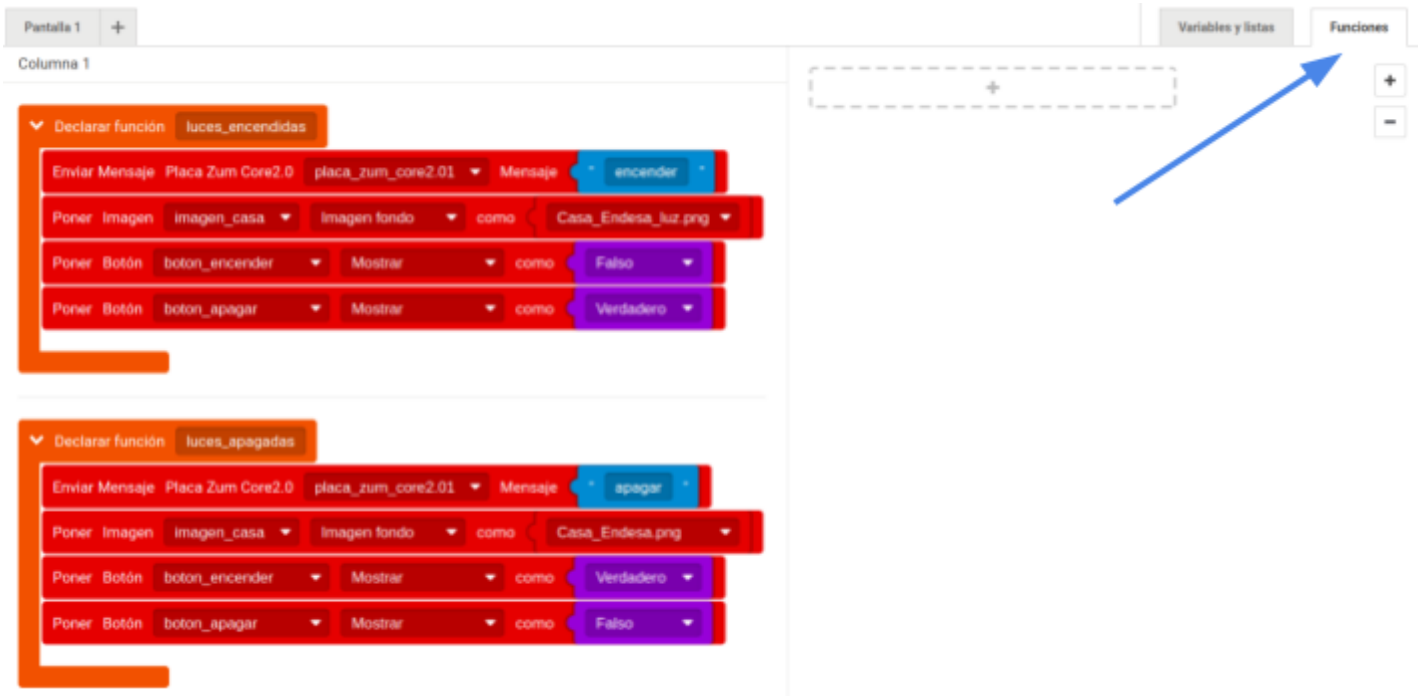
Para ello, podemos hacer una copia tanto de nuestro documento de Bitbloq Apps como del documento de Bitbloq Robotics y trabajar sobre los mismos.

1- Sensor de luz.

Programación en Bitbloq Apps

Con las mejoras que vamos a añadir ya vamos a utilizar bastantes bloques de programación, por lo que recomendamos mantener la programación ordenada para poder resolver más fácilmente posibles errores que vayan surgiendo.

Por ello, como en esta ocasión vamos a tener que repetir la secuencia de encender y apagar las luces en varios eventos, sugerimos declarar una **función** para cada secuencia, en el apartado de *funciones* de la parte superior derecha:



Además, tendremos que tener muy claro qué eventos son los que envían mensajes a la placa y, como ya advertimos anteriormente, recopilar todos los mensajes recibidos **desde la placa** en un evento ***Cuando Placa Zum Core2.0 mensaje recibido***.

Además, podemos hacer una lista de los mensajes que vamos a enviar a la placa y los mensajes que vamos a recibir:

Mensajes <u>enviados</u> a la placa:

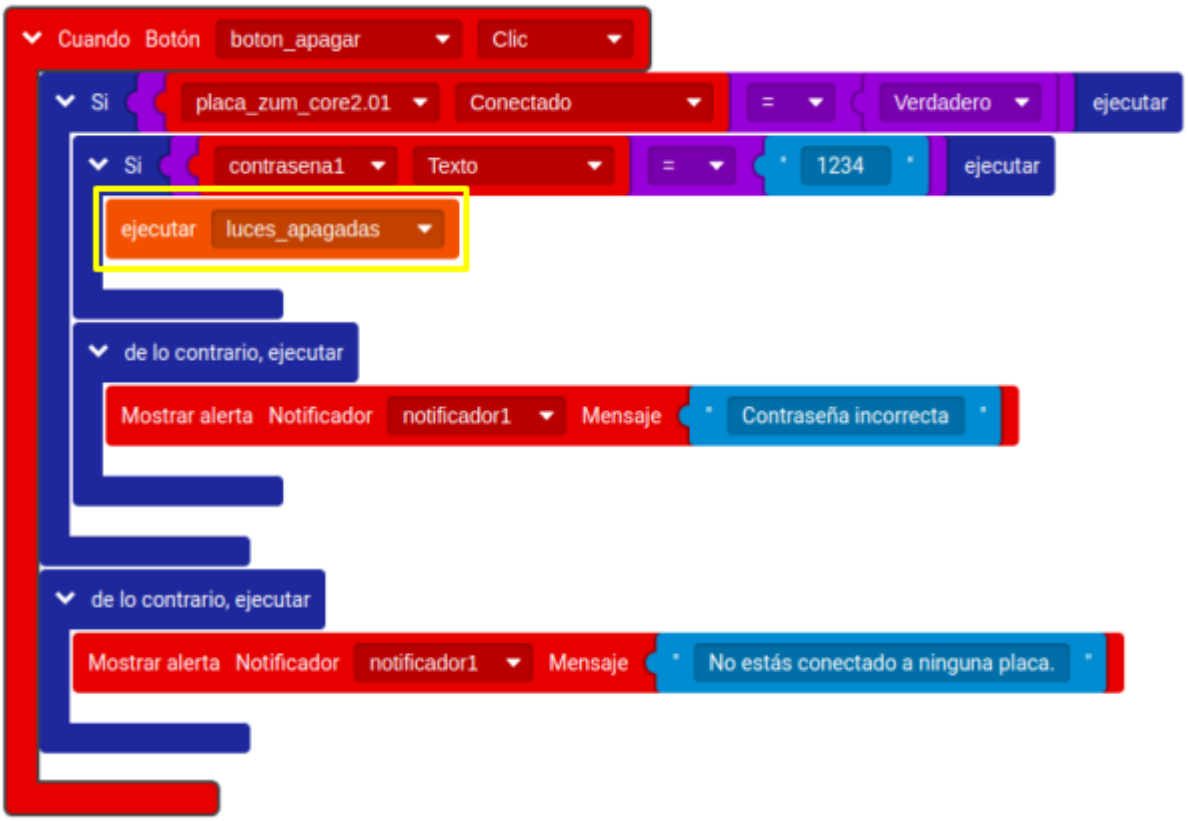
- "encender": para encender el LED de la farola.
- "apagar": para apagar el LED de la farola.
- "abrir": para abrir la puerta del garaje.
- "PreguntaLuz": para comprobar cuánta luz mide el LDR.
- "PreguntaCoche": para comprobar si hay un coche sobre el sensor infrarrojos.

Mensajes <u>recibidos</u> desde la placa:

- "abrir": para esconder el botón *Abrir*, indicando así que la puerta está abierta.
- "cerrar": para mostrar el botón *Abrir,* indicando así que la puerta vuelve a estar cerrada.
- "muchaluz": para indicarle a nuestra app que el LDR detecta mucha luz.
- "pocaluz": para indicarle a nuestra app que el LDR detecta poca luz.
- "nohaycoche": para indicarle a nuestra app que no hay un coche sobre el sensor infrarrojos.

Dicho esto, primero procedemos a programar los botones *boton_encender* y

boton_apagar. En esta nueva app no encenderemos la luz al pulsar el botón, sino que **le preguntaremos a nuestra placa "cuánta luz hay" y dependiendo de su respuesta encenderemos las luces o no**. Por tanto, la programación quedaría así:

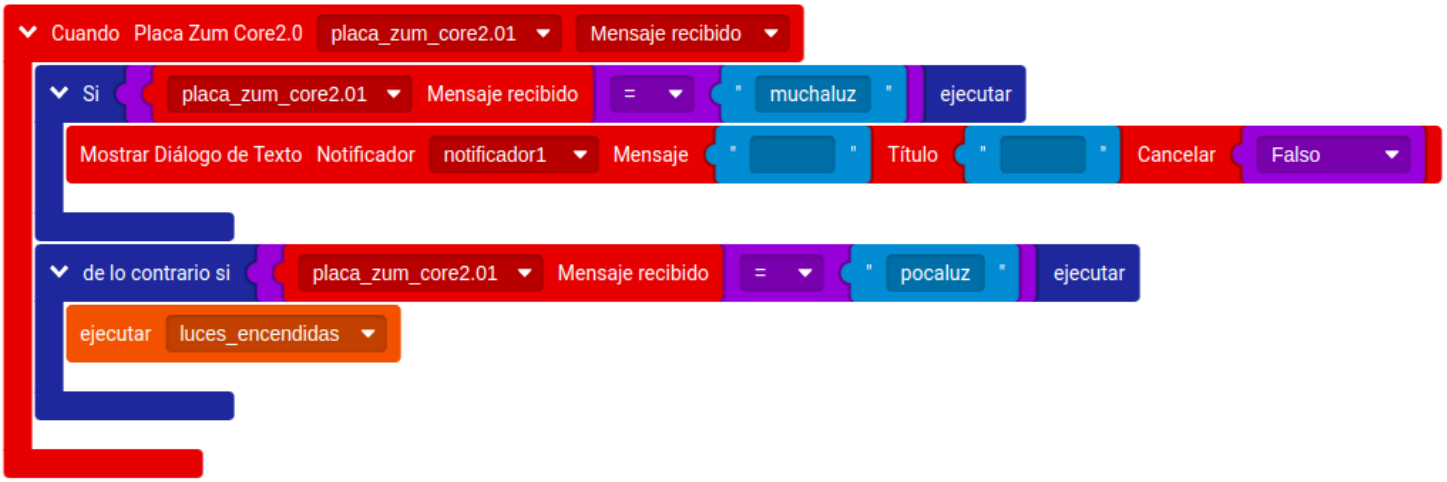


Para saber cuánta luz detecta el LDR tengo que utilizar el bloque de evento *Cuando Placa Zum Core2.0 mensaje recibido* y programar mediante condicionales la comprobación del texto recibido. Además, en el caso de recibir el mensaje “pocaluz”, ejecutamos la función de “luces_encendidas”.

En esta ocasión vamos a ayudarnos de la función del notificador *texto introducido* para poder decidir si encender las luces o no, aunque haya mucha luz. Por tanto, vamos a programar el bloque *Mostrar Diálogo de Texto notificador* para que cuando se reciba el mensaje “muchaluz” éste nos diga lo siguiente:

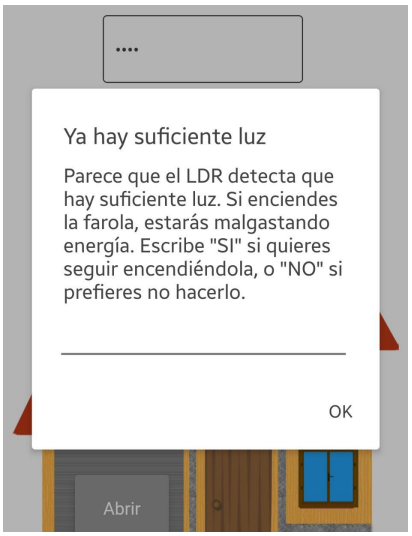
Título: Ya hay suficiente luz

*Mensaje: Parece que el LDR detecta que hay suficiente luz. Si enciendes la farola, estarás malgastando energía. Escribe "**SI**" si quieres seguir encendiéndola, o "**NO**" si prefieres no hacerlo.*



A continuación, necesitamos programar el notificador para que, dependiendo del texto que escribamos, realice una acción u otra. Para ello, necesitaremos el bloque de evento *Cuando Notificador Después de entrada de texto,* quedando la programación de la siguiente manera:

Esta notificación se verá así en nuestro dispositivo móvil:



Programación en Bitbloq Robotics

En Bitbloq Robotics tendremos que programar tanto el LDR para que envíe a nuestra app los mensajes “muchaluz” o “pocaluz” dependiendo de la cantidad de luz que detecte (en este ejemplo, consideramos que por encima de 100 es mucha luz):

2. Sensor infrarrojos.

Programación en Bitbloq Apps

En el caso de la puerta, le indicaremos al programa que cuando pulsemos el botón abrir, 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 desde la placa el mensaje “HayCoche” y, en el caso contrario, el mensaje recibido será “nohaycoche”. Lo programaremos más adelante en Bitbloq Robotics).

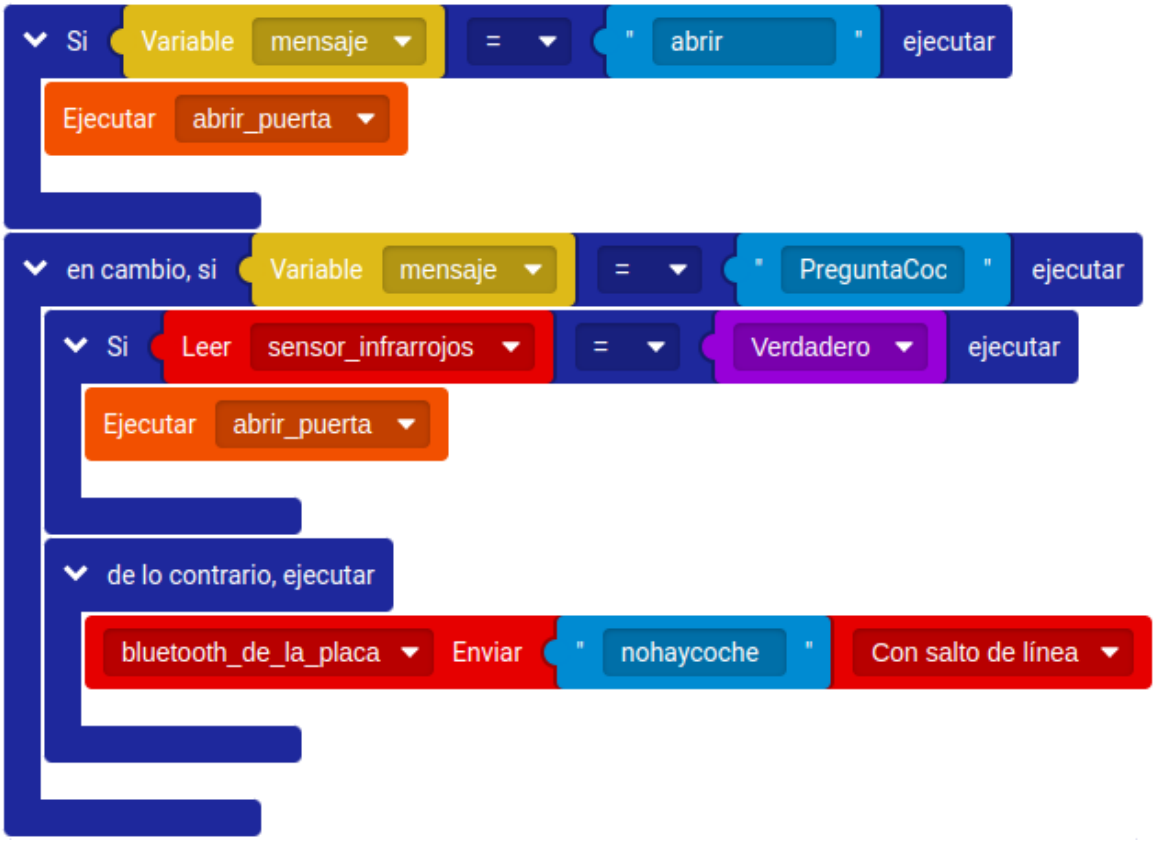
Más tarde programaremos en Bitbloq Robotics que, si el sensor infrarrojo detecta al coche se abra la puerta, pero si no detecta coche, avise a nuestra app con el mensaje “nohaycoche”.

Ahora tendremos que incorporar al bloque de evento *Cuando Placa Zum Core 2.0 mensaje recibido* la recepción de los mensajes “abrir”, “cerrar” y “nohaycoche”, teniendo en cuenta que con este último mensaje vamos a abrir el notificador (con entrada de texto) para que nos avise de que no hay coche y nos pregunte si aún así queremos abrir la puerta. Si escribimos “abrir” la abrirá, y si no, podemos pulsar en *Cancelar* para que no realice ninguna acción. Para que esté disponible esa opción debemos habilitar *Cancelar* como *Verdadero*.

Finalmente, tenemos que añadir a nuestro bloque de evento *Cuando Notificador Después de entrada de texto* que, al recibir la palabra “abrir” envíe el mensaje “abrir” a la placa.

Para finalizar, programaremos la placa controladora. Como hicimos en Bitbloq Apps, para simplificar la programación, vamos a declarar una función de la secuencia de apertura de la puerta llamada *abrir_puerta* en el apartado de *Variables globales y funciones*.

Tras esto, ejecutaremos la función *abrir_puerta* al recibir el mensaje *abrir* y, además, si el sensor infrarrojos detecta el coche, al recibir el mensaje “PreguntaCoche”. Si no detecta el coche, enviaremos el mensaje “nohaycoche” a nuestra placa



Se puede consultar la programación de todas estas mejoras en los archivos: *MEJORAS_Reto_2_Casa_Domotica_Apps.bitbloq* y *MEJORAS_Reto_2_Casa_Domotica_Robotics.bitbloq*.

Temporalización sugerida:

1ª Sesión: Iniciación a Bitbloq Apps. 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