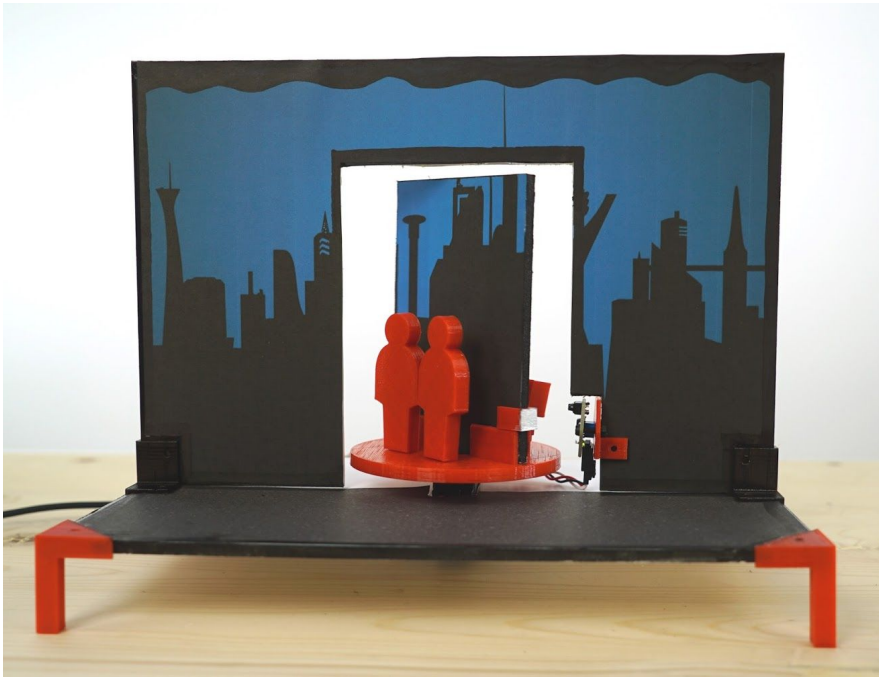
Proyecto:

TEATRO ROBÓTICO



Descripción del proyecto

En este proyecto el alumno construirá un teatro robótico que tendrá dos escenas para cada una de las cuales tendrás que desarrollar un diálogo. El proyecto integrará robótica, programación de aplicaciones móviles y diseño e impresión 3D.

Dificultad

Media

Tiempo de dedicación estimado

8 horas

¿Qué necesitamos?

Para realizar este proyecto se necesitarán los siguientes materiales y programas:

Componentes del kit de robótica

- Placa Arduino Zum
- 1 x Sensor de infrarrojos
- 1 x Servo de rotación continua
- 1 x Pulsador

- Cabezal circular pequeño del servo de rotación continua
- Tornillo negro pequeño para el servo de rotación continua
- Cable de conexión USB - mini USB

Materiales

- Dos placas de cartón pluma de 5 milímetros de espesor y tamaño similar a un din A4 cada una de ellas
- Algunas imágenes para el fondo y el suelo del escenario, aunque también se pueden dibujar a mano
- Pegamento de barra
- Destornillador de estrella pequeño
- Cúter
- Tijeras
- Rotuladores o pinturas para el escenario
- Lápiz, goma y regla
- 2 tornillos M3 para sujetar el sensor infrarrojos a su soporte
- 2 tuercas M3

Programas

- Google Chrome
- AppInventor
- Bitbloq
- Openscad
- Cura

Otros

- Dispositivo móvil Android
- Impresora 3D
- Filamento para impresora 3D

Desarrollo de sesiones

La duración del proyecto será de 8 sesiones, distribuidas de la siguiente forma:

1. Preparación de la historia que se contará en el teatro y diseño de los personajes
2. Preparación de las piezas para ser impresas
3. Montaje del escenario
4. Programación de la placa de Arduino para primeras pruebas

5. Diseño de la interfaz de la aplicación móvil
6. Programación de la aplicación móvil
7. Modificación de la programación para arduino para usar la aplicación móvil
8. Extras

Preparación de la historia

En este proyecto el alumno tiene que construir un teatro que tendrá un escenario fijo con una parte central que girará para mostrar dos situaciones o escenas diferentes. Cada una de las escenas tendrá dos personajes. El objetivo será crear una historia breve en la que se produzcan diálogos entre las parejas de personajes.

Se ha creado una historia que puede ser utilizada sin modificarla, servir de base para ser ampliada o modificada, o bien puede ser un mero ejemplo, creando tus propios personajes e historia.

Dado que va a condicionar todo el proceso, lo primero que se debe hacer es decidir si se va a recrear la misma historia, modificar la que ponemos de ejemplo o ampliarla.

La historia que proponemos como ejemplo es la siguiente:

Escenario 1

Aparecen dos seres humanos que inician el siguiente diálogo:

Humano 1: *"Pues a mí me dan miedo los robots…"*

Humano 2: *"No te preocupes, la primera ley de la robótica dice que un robot nunca puede hacer daño a un ser humano o, por inacción, permitir que un ser humano sufra daño"*

Escenario 2

Aparecen dos robots, que inician el siguiente diálogo:

Robot 1: *"Mi lógica me dice que los humanos deberían seguir la primera ley de la robótica y dejar de hacerse daño entre ellos"*

Robot 2: *"Mi lógica me dice que tu expresión es verdadera"*

Esta historia puede ser modificada o continuada, el escenario central gira para mostrar unos personajes u otros, de forma que se pueden realizar varias sesiones

de diálogo entre las diferentes parejas. El diálogo será mostrado por la aplicación móvil y también, más adelante, habrá que grabarlo en audio para reproducirlo.

En este primer paso se debe crear una historia sólida, para ello el alumno tiene que decidir si utilizará los personajes dados o elegirá sus propios personajes y buscará la forma de incluirlos en el proyecto.

Aunque el diseño de personajes se trabajará en una sesión posterior , se debe tener en cuenta que existe la posibilidad de diseñarlos con OpenSCAD (creando unos personajes algo simples a base de cubos, cilindros, esferas...), buscar personajes en algún repositorio de diseños en 3D como Thingiverse o bien, en vez de imprimirlos, construirlos, representandolos con cartón y un dibujo, usando muñecos, etc. Las posibilidades son muchas, el alumno debe pensar qué personajes e historia quiere reproducir.

Para recordar cómo diseñar con OpenSCAD se puede acudir al siguiente enlace:

http://diwo.bq.com/course/curso-de-iniciacion-al-diseno-3d-con-openscad-por-obijuan/

Para buscar diseños en Thingiverse se puede acudir al siguiente enlace:

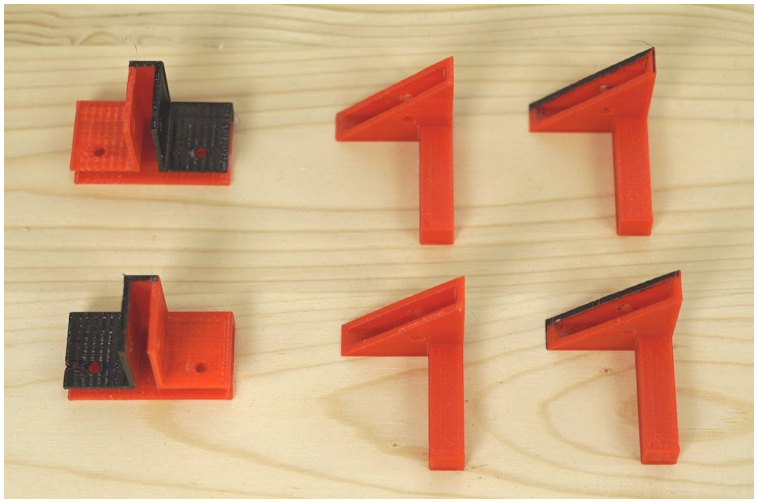
https://www.thingiverse.com/

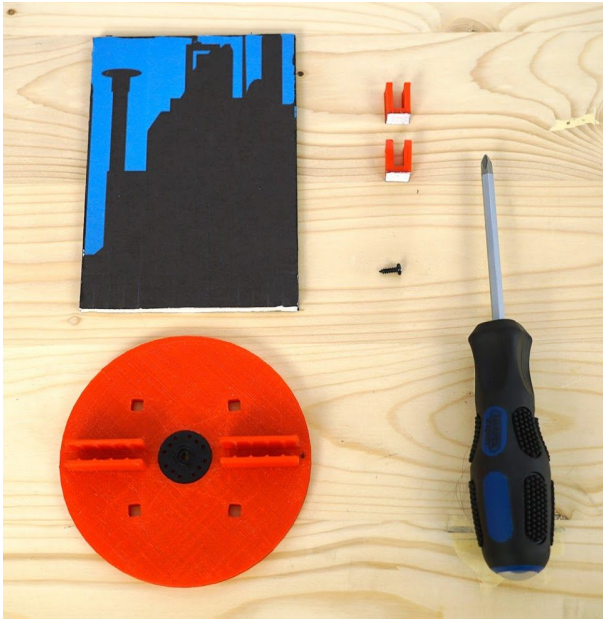
Preparación de las piezas para ser impresas

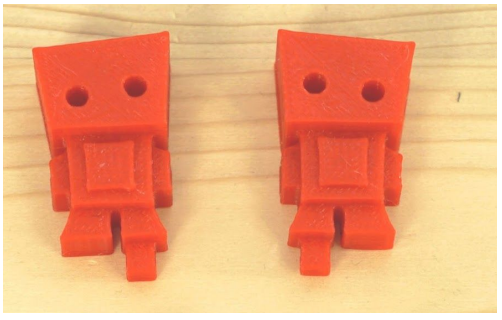
Una vez se ha decidido una historia y se han desarrollado los diálogos, toca preparar todas las piezas necesarias para montar el escenario.

Las piezas necesarias son las siguientes:

- Patas laterales que eleven el escenario
- Dos piezas que unan el suelo del escenario con el fondo del escenario
- Una pieza para situar en la cabeza del servo de rotación contínua y en la cual se puedan poner los personajes y el trozo de escenario giratorio
- Un soporte para sujetar el sensor de infrarrojos al escenario
- Dos piezas para situar en el lateral de la parte móvil del escenario y poder ser pintadas de blanco, de forma que las detecte el sensor de infrarrojos y pueda detener el giro







Antes de proseguir conviene explicar por qué se usa un sensor de infrarrojos para detener el giro del servo de rotación continua. Es habitual que los alumnos piensen en la opción de poner el servo a girar un tiempo determinado y pararlo tras ese tiempo, pero la realidad es que el giro del servo no es lo suficientemente exacto como para poner un tiempo y confiar en que siempre va a girar exactamente los mismos grados. Por ello, es mucho mejor detener o poner en movimiento los servos por condiciones externas como puede ser que un sensor detecte cierto cambio. En nuestro caso el cambio será que pase de detectar vacío (negro) a detectar blanco en el borde del escenario.

Para facilitar el proyecto, se dan todas las piezas ya diseñadas. La pieza que será la cabeza del servo de rotación continua se entrega en STL para ser impresa, así como los personajes, pero el resto de piezas se entregan diseñadas en OpenSCAD, además de en STL, para que se pueda identificar de qué pieza se trata y ver cómo está diseñada.

Se ha seleccionado el cartón pluma como material del suelo y el escenario por no ser muy rígido, de esta forma se pueden diseñar piezas que se introduzcan en el cartón pluma a modo de grapa, sin tener que utilizar adhesivo ni ningún tipo de elemento de unión. Aun así se puede pegar el material a las piezas impresas y las mismas tienen orificios para poder utilizar un pasador o algún elemento de unión extra, aunque por lo general no es necesario.

A continuación se muestran los códigos para ser utilizados, basta con copiarlos y pegarlos en OpenSCAD.

Sería muy positivo que el alumno tratase de identificar las partes del diseño. Podría comentar todo el código para indicar qué hace cada bloque del mismo.

Patas:

```
translate([-11,19,-11])
    cube([8,30,8],center=true);
difference(){
    cube([30,8,30],center=true);
    translate([17.5,0,18])
        rotate([0,45,0])
            cube([50,10,50],center=true);
    cube([27,5,27],center=true);
    translate([-4,0,-4])
        rotate([90,0,0])
            cylinder(r=1.6,h=10,$fn=20,center=true);
    }
```

Soporte para el sensor infrarrojos:

```
translate([5,3.86,0])
    rotate([0,0,90])
        difference(){
            cube([10,10,15]);
            translate([5,-1,9])
                rotate([-90,0,0])
                    cylinder(r=1.6,h=10,$fn=20);
            translate([-1,2.5,2.5])
                cube([12,5,15]);
            }
difference(){
    cylinder(r=16, h = 2, $fn=6);
    translate([10,0,-0.1])
        cylinder(r=1.7,h=2.2,$fn=20);
    translate([-10,0,-0.1])
        cylinder(r=1.7,h=2.2,$fn=20);
    }
```

Soportes laterales del escenario:

```
difference(){
    union(){
        cube([20,40,8],center=true);
        translate([0,0,14])
            cube([20,8,20],center=true);
        }
    translate([1.5,0,0])
        union(){
            cube([20,42,5],center=true);
            translate([0,0,14])
                cube([20,5,24],center=true);
            }
    translate([5,-13,0])
        cylinder(r=1.6,h=20,$fn=20,center=true);
    translate([5,13,0])
        cylinder(r=1.6,h=20,$fn=20,center=true);
    translate([5,0,18])
        rotate([90,0,0])
            cylinder(r=1.6,h=20,$fn=20,center=true);
    }
```

Pieza para que el infrarrojos detecte el borde del escenario giratorio:

```
difference(){
    cube([10,10,15]);
    translate([5,-1,9])
        rotate([-90,0,0])
            cylinder(r=1.6,h=10,$fn=20);
    translate([-1,2.5,2.5])
        cube([12,5,15]);
    }
```

Para descargar los archivos y poder imprimirlos se puede buscar en Thingiverse el proyecto “Little Theater” o bien usar el siguiente enlace al proyecto donde están subidas todas las piezas en STL listas para ser laminadas e impresas:

https://www.thingiverse.com/thing:2538492

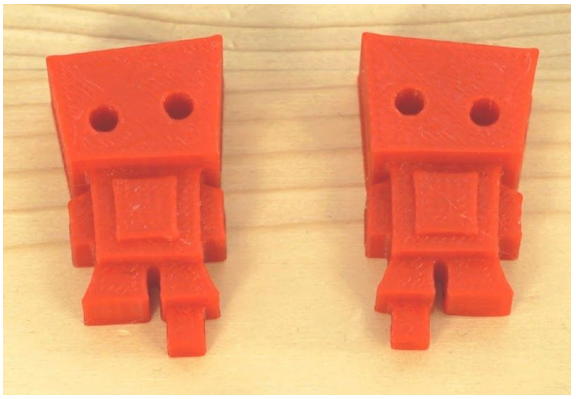
Personajes

Por defecto el proyecto está construido usando 2 parejas de personajes. En el primer escenario aparecen 2 figuras humanas y en el segundo aparecen 2 robots.

Si se opta por utilizar cartón o similar para crear unos personajes se puede diseñar una pieza que encaje en los huecos dejados en la parte del escenario que gira, de forma que sujete al personaje de cartón.

En todo caso, para cualquier diseño que se haga se debe tener en cuenta que el agujero en el suelo de la parte del escenario que gira en el cual se pondrá el personaje es de **forma cuadrada y de 5 milímetros de lado**. Para que cualquier pieza impresa encaje, se debe dejar cierta holgura pues la impresión 3D no es algo tan exacto como para poder imprimir del tamaño del hueco y que quepa en el hueco. Por ello la patilla para ser introducida en el agujero del suelo, debe ser de **sección cuadrada y de unos 4,6 milímetros de lado**. Posiblemente, por ser la primera capa impresa más ancha, haya que limar un poco los bordes de la patilla para introducirla.

Lo idóneo sería, en caso de querer imprimir los personajes, seleccionar o diseñar piezas que puedan ser impresas tumbadas, con la patilla también apoyada en el suelo ya que, sobre el escenario, la parte trasera de las piezas no se verá.



El **tamaño** de los personajes también es un aspecto a tener en cuenta, ya que los agujeros están próximos y el ancho de los mismos no puede ser elevado. Así mismo, la pieza giratoria tiene un diámetro concreto y hay que tener cuidado de no poner personajes demasiado voluminosos que choquen con la parte fija del escenario al producirse el giro.

Montaje del escenario

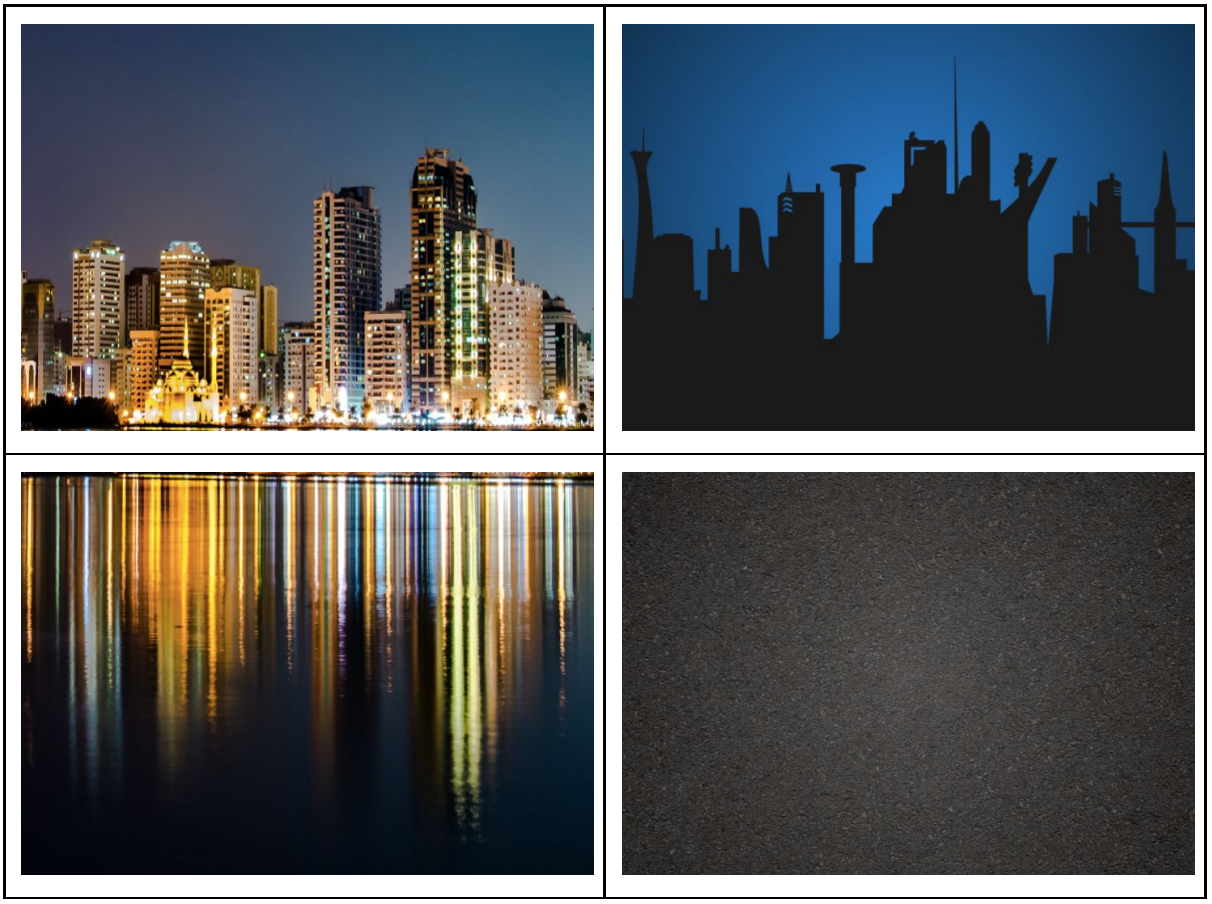
Una vez se tengan todas las piezas impresas necesarias y se haya conseguido el cartón pluma, se puede proceder a preparar y montar el escenario.

Fondo

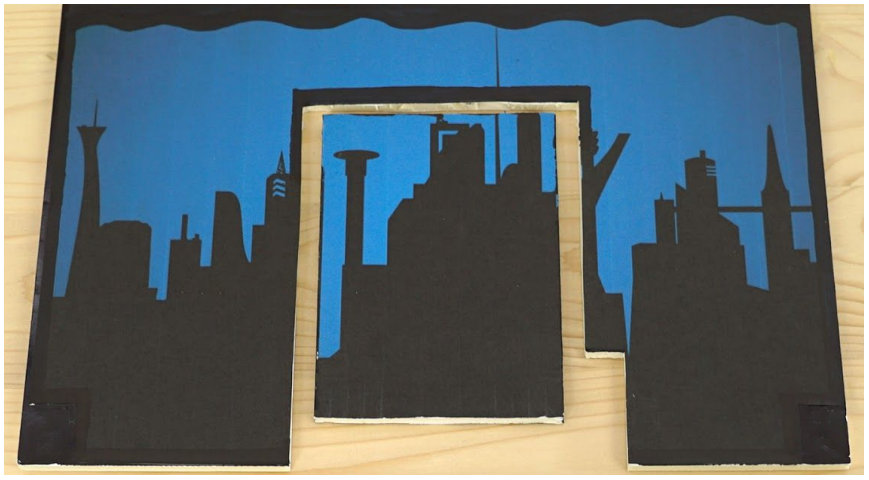
En primer lugar, el escenario debe tener un decorado de fondo y un decorado de suelo. Existe la posibilidad de pintar sobre el cartón pluma, o bien imprimir algún fondo y pegarlo **con pegamento de barra** en la pared y el suelo del escenario.

En caso de decidirse por decorar el escenario a mano, dibujando, se puede decorar antes de cortarlo. En caso de elegir imprimir un fondo y pegarlo, se recomienda cortar primero el cartón pluma y luego pegar el fondo al suelo y a la pared.

A modo de ejemplo se proponen dos ideas para fondo y suelo:



Si se va a imprimir un fondo se necesitarán dos copias, una de las mismas irá sobre el fondo y una de las caras de la parte giratoria del escenario, la otra servirá para poder poner en la otra cara de la parte giratoria también la parte del fondo que le corresponde.



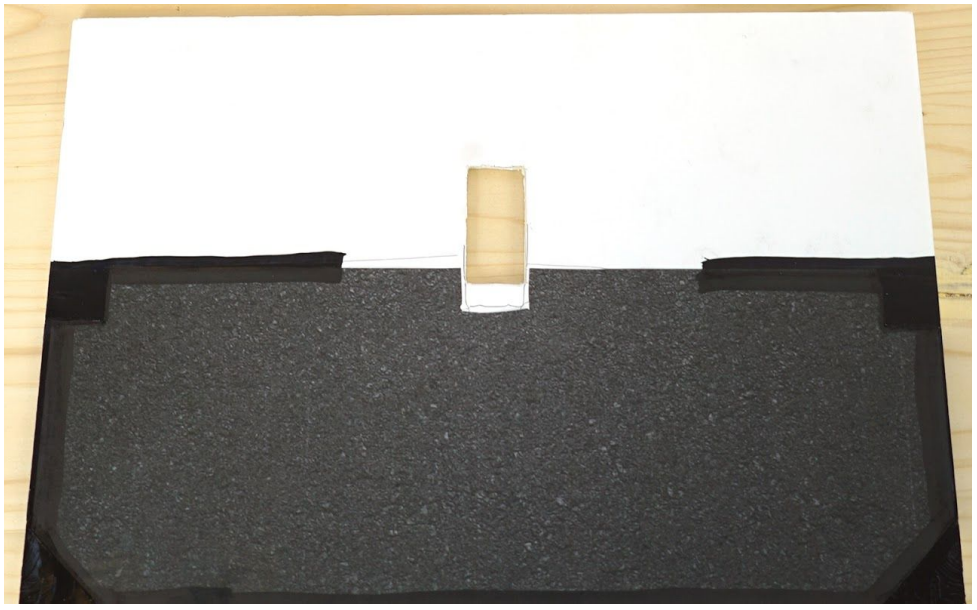
En caso de optar por pintar o dibujar el fondo manualmente, se deben elegir pinturas que funcionen bien sobre la superficie del cartón pluma. Los rotuladores funcionan

muy bien, pero los lápices no tanto. Es mejor no utilizar pinturas diluidas (témperas, por ejemplo) ya que la humedad hace que se combe el cartón.

Cortando el cartón pluma

El cartón pluma se corta muy bien con un cúter, teniendo mucha precaución para no cortarse y poniendo el cartón pluma sobre una superficie protegida frente al filo del cúter. Para proteger una mesa lo idóneo es utilizar cartón gordo o plástico rígido (una tabla de cortar de cocina, por ejemplo).

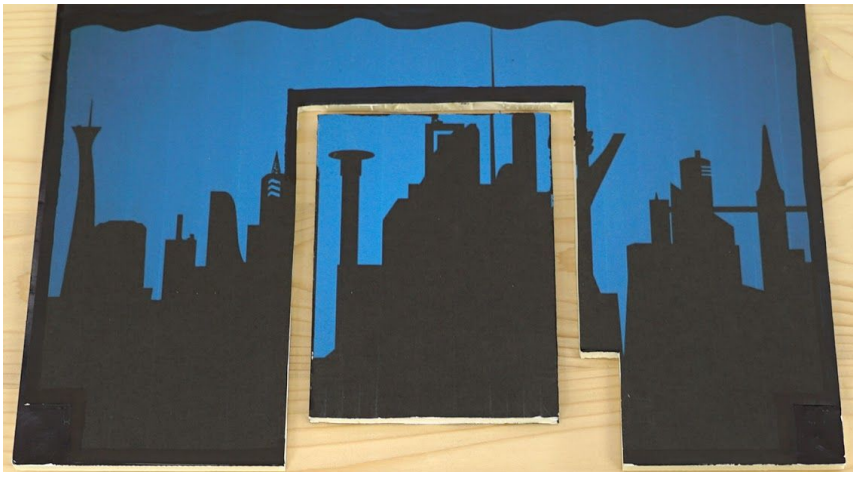
Se debe marcar y cortar, sobre el cartón pluma que hará la función de suelo del escenario, un **rectángulo para introducir el servo de rotación continua**. El tamaño del rectángulo debe ser el mismo que tenga el servo (en el caso de los servos que vienen en el kit de robótica dicho rectángulo será de 40 x 20 mm). El servo se introducirá a presión para que quede fijo sin tener que pegarlo ni atornillarlo.



Lo ideal es que el servo esté un poco más atrás de la mitad del escenario. Para introducirlo en el orificio rectangular habrá que utilizar algo de fuerza, con cuidado de no romperlo, introduciendo primero el cable y luego forzando un poco el servo de cara a que se introduzca en el agujero.

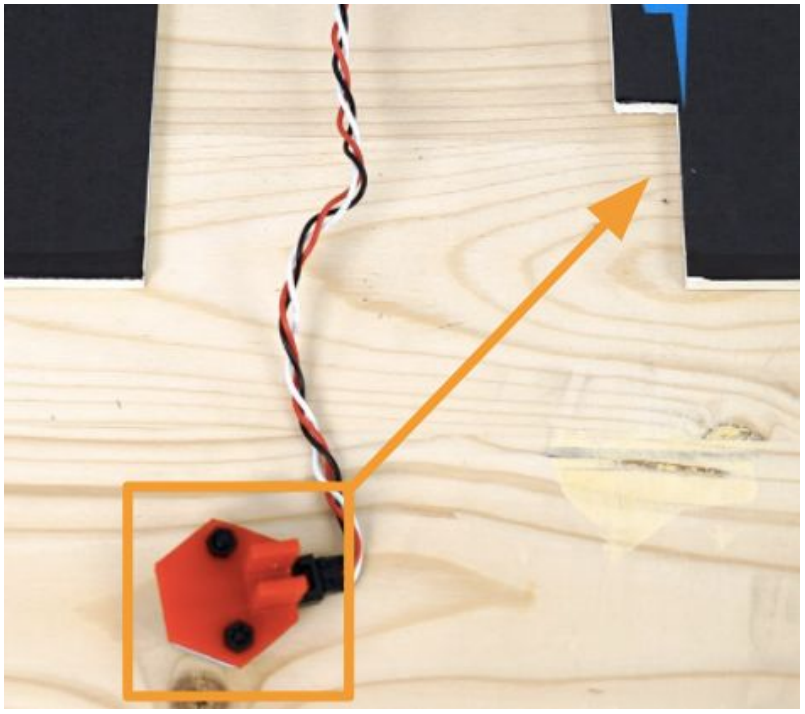
Una vez se haya colocado el servo, se puede unir al mismo la pieza impresa en la que irán los personajes, a la cual hay que colocarle el cabezal circular del servo y atornillarla con uno de los tornillos negros que contiene la bolsita con componentes del servo.

Cuando esté colocado el servo con la pieza atornillada se puede marcar sobre el fondo el agujero que se debe hacer para poder colocarlo y que tenga una parte giratoria. Para ello se debe poner el fondo sobre el suelo del escenario, sin sujetarlo, pero con la exactitud necesaria para ver exáctamente dónde está la pieza impresa respecto del suelo del escenario.



Hay que tener en cuenta que, al girar, la cabeza del servo puede estar ligeramente inclinada a alguno de los lados y, además, el cartón pluma tiene un cierto espesor, por lo que es mejor dejar holgura, como se muestra en la imágen superior, de forma que al girar el escenario móvil no choque con el escenario fijo.

La mejor opción es marcar con una regla el rectángulo que se va a cortar y luego cortarlo con el cuter teniendo mucho cuidado. Además habrá que cortar, en la parte inferior de uno de los laterales del hueco, un rectángulo más para poder situar ahí el sensor de infrarrojos.



La pieza de cartón pluma retirada va a ser la parte móvil, aunque habrá que recortarla un poco más para dejar la ya mencionada holgura. Además, será bastante menos alta que el hueco sobre el escenario ya que el servo y la pieza impresa sobresalen del suelo. Para ello lo mejor es medir el hueco que queda una vez montada la parte fija del fondo y, conociendo esa medida, recortar la parte móvil.

Una vez recortado el cartón pluma conviene pintar las dos piezas impresas que servirán para que el sensor infrarrojos detecte el borde del escenario. El sensor detecta el vacío como negro, por lo que debes pintar el borde de las dos piezas de blanco y así, al estar colocadas en el canto del escenario móvil, el sensor infrarrojos las detectará y podremos detener el giro. Para pintarlas lo más recomendable es usar tipex o algún permanente blanco.

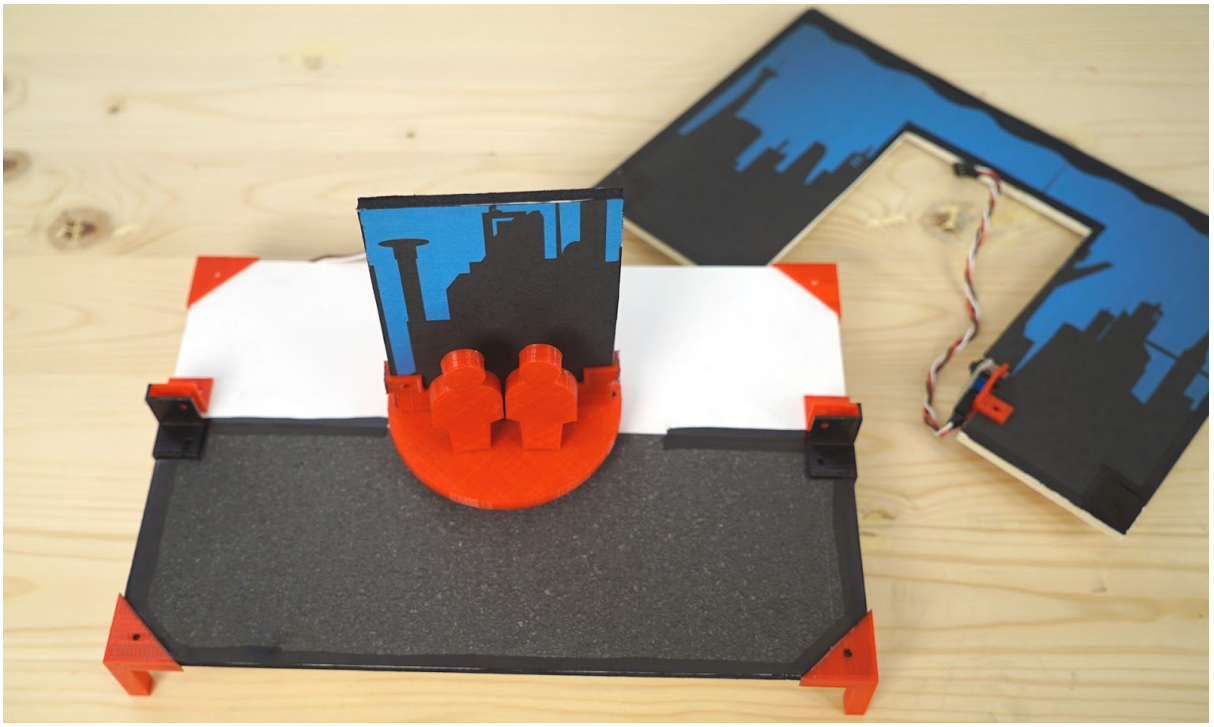
Otra posibilidad es no utilizar estas piezas y pintar directamente de blanco el borde del escenario giratorio, pero estas piezas permitirán poner el sensor de infrarrojos a cualquier altura.

Una vez pintadas ya se puede montar el escenario.

El orden idóneo es:

1. Dejar bien sujeto el servo al suelo del escenario
2. Poner las patas en las esquinas del suelo del escenario
3. Unir la pieza impresa al servo
4. Colocar los personajes en los agujeros de la pieza que va unida al servo
5. Colocar el trozo de cartón pluma en la pieza impresa unida al servo
6. Poner en el trozo de cartón pluma las piezas para que el sensor infrarrojos detecte el blanco

7. Colocar en el fondo fijo el sensor infrarrojos, sujetándolo a la pieza de sujección impresa con dos tornillos y sus dos tuercas
8. Colocar el fondo fijo sobre el suelo del escenario, utilizando las piezas impresas para tal fin



Durante el proceso de montaje se debe pegar, en caso de haber optado por imprimirlas, las imágenes que decoran el escenario, tanto la del fondo, como la del suelo, como las dos caras de la parte móvil del escenario.

Programación de la placa de Arduino para primeras pruebas

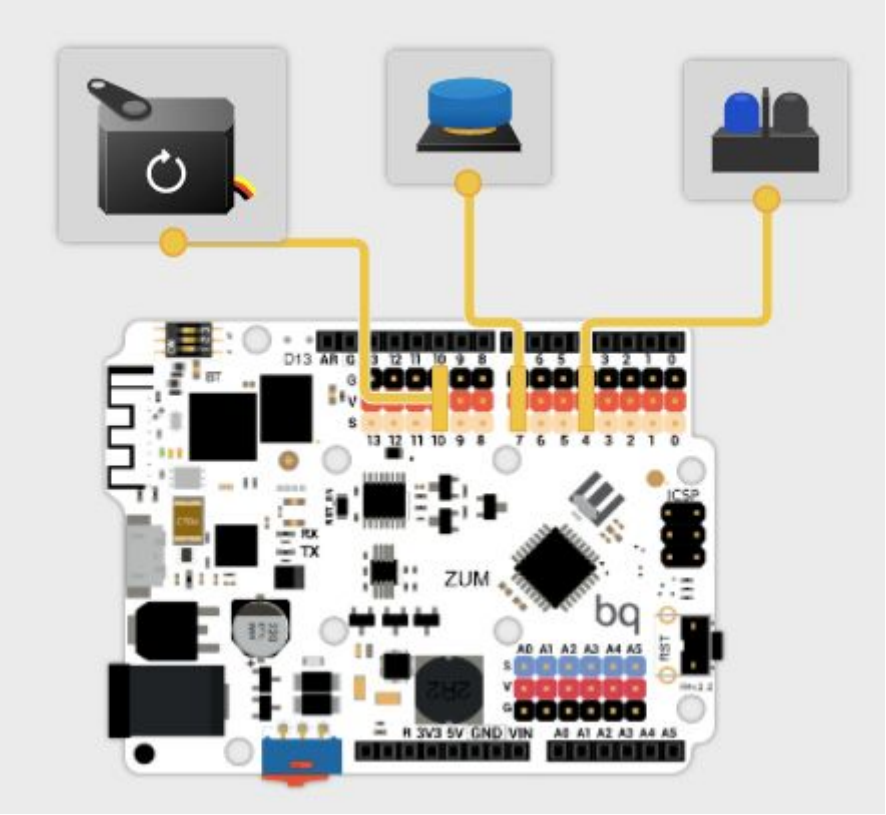
Una vez esté el escenario montado, es el turno de la programación de la placa de Arduino, a la cual se **conectará el servo de rotación contínua y el sensor de infrarrojos**.

La programación va a ser sencilla, el servo girará hasta que el sensor infrarrojos detecte blanco, momento en el que se detendrá. Más adelante el comienzo del giro lo determinará la aplicación móvil que se construirá con AppInventor, pero ahora mismo vamos a iniciar el giro **añadiendo a la placa un pulsador**.

Si hay dudas de cómo usar algún componente o programarlo se puede recordar leyendo el curso de programación de placas Arduino con Bitbloq:

http://diwo.bq.com/course/aprende-robotica-y-programacion-con-bitbloq-2/

En la aplicación Bitbloq se conectan los componentes en los pines que se hayan utilizado en la placa:



Cuando esté conectado todo, toca programar que, al pulsar el botón, el servo empiece a girar **hasta que detecte blanco** el sensor de infrarrojos. Hay que tener cuidado, si el escenario móvil se detiene al detectar el sensor de infrarrojos blanco, cuando se ponga en marcha de nuevo, el sensor ya estará detectando blanco, pero necesitamos que gire hasta que vuelva a detectar el blanco del otro lateral de la parte móvil del escenario.

Por ello, vamos a programar una secuencia similar a:

Si el botón está pulsado → gira el servo de rotación continua en sentido horario → mientras detectes blanco sigue girando → mientras detectes negro sigue girando → cuando detectes blanco de nuevo detén el servo de rotación continua.

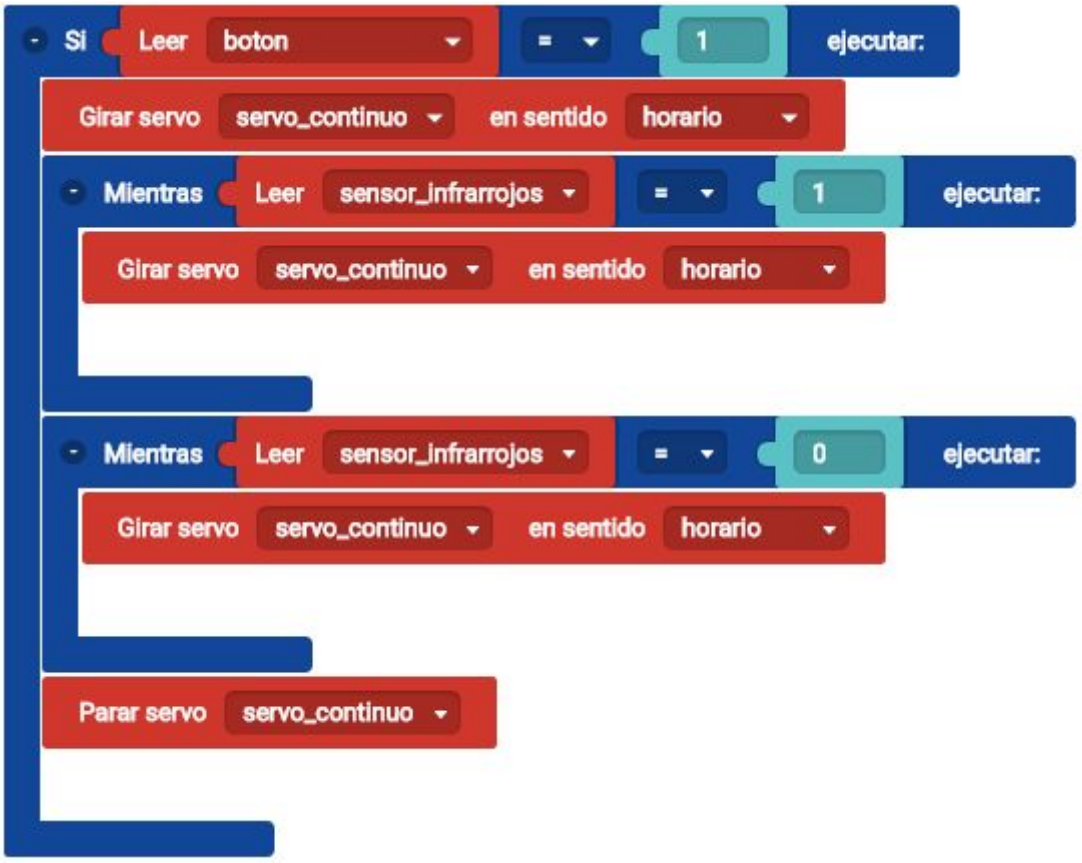
Se van a usar bucles *mientras* (*while*) porque nos permiten que la placa continúe realizando una acción ***mientras* se produzca una condición**, atrapando al programa en ese bucle hasta que se deje de cumplir la condición. Se programará lo siguiente:

- Si pulso botón
    - Poner servo a girar
    - Mientras sensor infrarrojos detecte blanco
        - Sigue girando el servo
        - Deja de detectar blanco, el programa sale del bucle mientras
    - Mientras sensor infrarrojos detecte negro

        - Sigue girando el servo
        - Deja de detectar negro, el programa sale del bucle mientras
    - Parar el servo

Sería idóneo que el alumno intentase hacer esa programación solo. Si tiene problemas puede acudir a los tutoriales de Diwo para ver cómo se programaba cada componente.

Si tras intentarlo con ganas no lo consigue, se le puede enseñar cómo se programaría la secuencia descrita.



Es una programación bastante sencilla, incluso se puede simplificar más aún. Una vez que decimos al servo que comience a girar, éste no va a detenerse hasta que no le demos la orden de detenerse o apaguemos la placa. Por lo tanto, sólo hay que indicar los cambios de estado y no es necesario volver a dar la orden de poner a girar el servo si ya está girando.

La programación, teniendo en cuenta lo anterior, podría ser así de sencilla:

El objetivo de esos bucles *Mientras* vacíos es simplemente **atrapar al programa** para que no pueda continuar hasta que no se deje de cumplir la condición que contiene el *mientras*. El programa pone a girar el servo, luego pasa a quedarse en bucle hasta que deja de detectar blanco, luego pasa a quedarse en bucle hasta que deja de detectar negro y luego para el servo.

Es posible que el sensor infrarrojos detecte blanco en zonas que no sean la pieza pensada para tal fin. Por ejemplo, el plástico de colores claros (como el plástico rojo de las imágenes) puede ser confundido con blanco por el sensor, o el propio cartón pluma si no se ha pintado o decorado de un color oscuro.

Por ello, es muy importante calibrar bien el sensor de infrarrojos para que detecte únicamente como blanco la pieza lateral. En esta entrada del curso de Bitbloq puedes recordar cómo se calibra un sensor:

http://diwo.bq.com/o-blanco-o-negro-el-sensor-infrarrojo/

Diseño de la interfaz de la aplicación móvil

Una vez esté en funcionamiento la placa y los componentes, se va a empezar a diseñar y programar la aplicación móvil. En esta ocasión, de nuevo, no vamos a hacer algo muy complejo, aunque sí es cierto que programar una aplicación móvil siempre lleva algo más de tarea que programar una placa Arduino.

Para empezar, hay que trabajar en dos áreas diferentes. Una es el ***frontend***, es decir, lo que el usuario va a ver de la aplicación y otra es el ***backend***, es decir, cómo se van a comportar esos botones y partes gráficas al ser utilizados por el usuario.

En AppInventor tenemos dos zonas para diferenciar lo anteriormente mencionado.

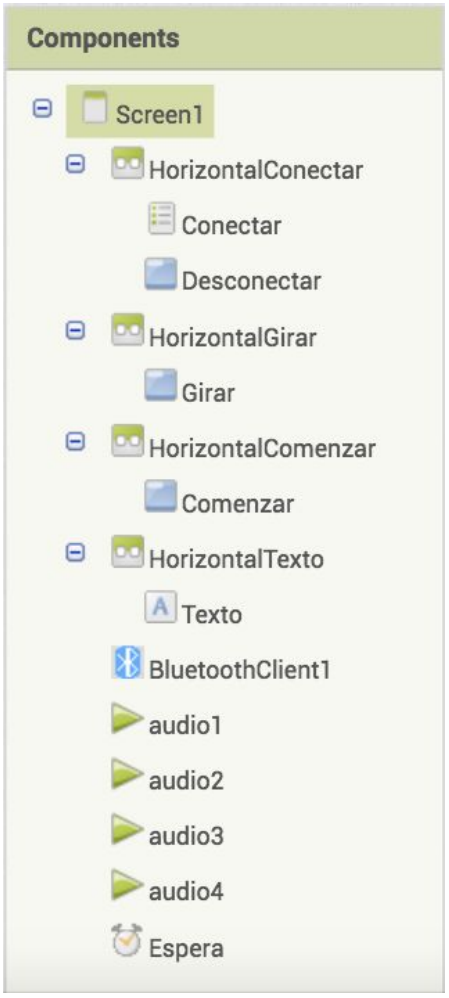
En la parte *Designer* crearemos el *frontend*, es decir, la parte visible para el usuario y en la parte *Blocks* crearemos el *backend*, es decir, programaremos cómo queremos que se comporte la aplicación.

Empezaremos, en esta primera etapa, con el *backend*, en un nuevo proyecto en AppInventor (http://ai2.appinventor.mit.edu/) y, en el mismo, en la parte de *Designer*.

Para programar la aplicación se necesitarán una serie de elementos en la interfaz para poder programarla:

- Un ***Listpicker*** (lista para seleccionar) para conectar el móvil al Bluetooth de la placa (que se elegirá de la lista de Bluetooth).
- Tres ***Button*** (botones). Uno para desconectar el móvil de la placa, otro para iniciar la secuencia del teatro y un último botón para girar media vuelta el servo por si no está inicialmente colocado en la posición correcta.
- Una ***Label*** (etiqueta) para escribir el texto de los diálogos que debe mostrar la aplicación.
- Un ***BluetoothClient*** (cliente Bluetooth) para poder conectar la aplicación a la placa Arduino.
- Un ***Clock*** (reloj) para producir esperas en caso de necesitarlas.
- Tantos ***Player*** (reproductores) como diálogos tenga tu proyecto. En el caso del ejemplo hay cuatro diálogos, por lo que habrá cuatro *Player*.

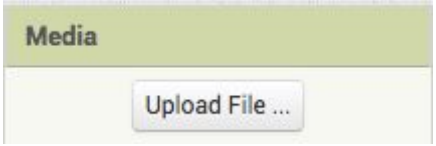
Los elementos anteriores se dispondrán según el siguiente esquema, añadiendo las disposiciones horizontales necesarias:



Es importante renombrar todos los elementos para poder luego programar sin problemas.

El dispositivo *Player* necesita que indiquemos que audio tiene que reproducir, por lo que hay que subir un archivo de sonido con cada uno de los diálogos grabados en audio. El alumno debe grabar por separado cada diálogo con algún compañero o compañera. Una vez estén grabados se seguirá el siguiente proceso:

Seleccionar *Upload File…* en el menú *Media*:



Seleccionar el archivo de audio a subir:

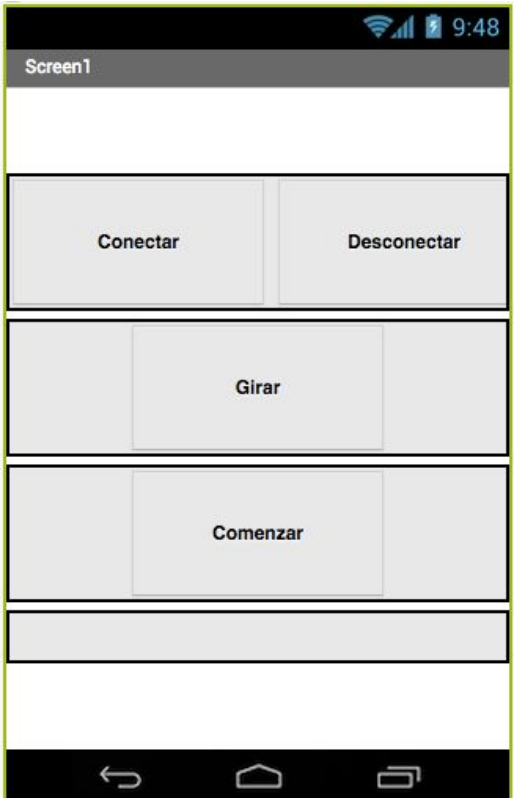
Una vez se haya subido el archivo, añadirlo en el apartado *Source* del menú del *Player* que corresponda:

Se repetirá el proceso para cada uno de los demás diálogos.

La pantalla principal de AppInventor, llamada por defecto *Screen1*, es una disposición vertical. Para que los botones estén centrados verticalmente se debe activar la opción de alineación vertical centrada dentro de *Properties* de *Screen1*.

Una vez estén incluidos todos los elementos toca darles el mejor aspecto posible. El alumno debe cambiar las opciones que considere necesarias para dejar bien el tamaño, forma, textos, etc. El texto de la etiqueta inicialmente debe estar vacío, pues ahí se escribirán los diálogos del proyecto.

El aspecto debe ser similar a este (o más atractivo):



Se utilizará AiCompanion para observar cómo se visualiza el proyecto en el dispositivo móvil o tablet. El siguiente enlace explica el proceso para conseguirlo:

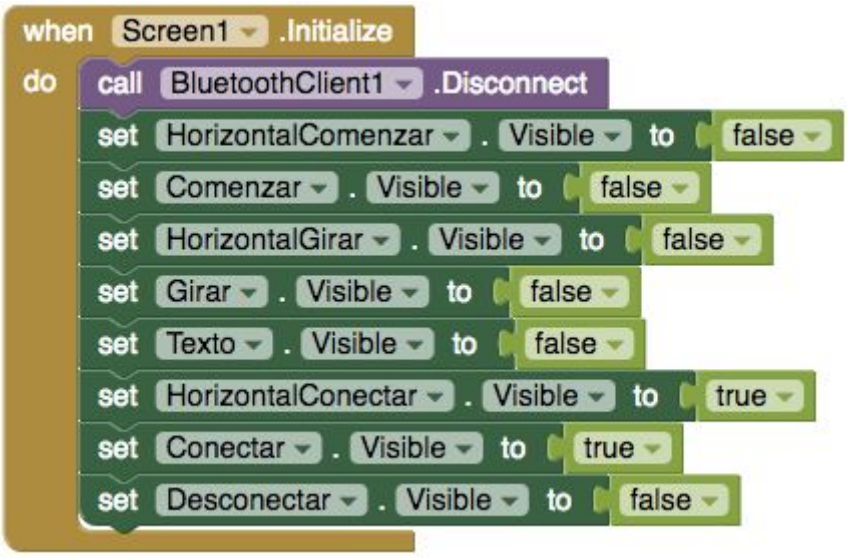
http://appinventor.mit.edu/explore/ai2/setup-device-wifi.html

Una vez tenga un aspecto lo más depurado posible, toca programar la aplicación.

Programación de la aplicación móvil

En la opción *Blocks* se programará el *backend* de la aplicación.

Lo primero que hay que indicar es qué partes de la aplicación están visibles y qué partes están ocultas al inicializarse:

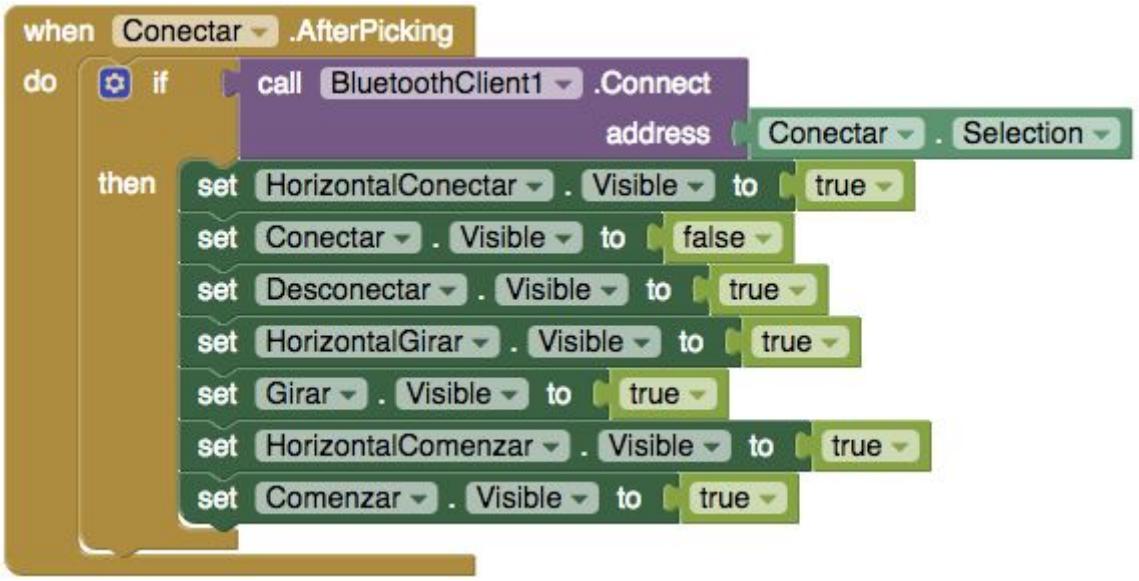


Al iniciarse la aplicación se visualiza, por defecto, la pantalla *Screen1*. Dejaremos sólo visibles el *Listpicker Conectar* y su disposición horizontal, quedando el resto como no visible (o, lo que es lo mismo, *Visible* como *false*). También, para asegurar que esté desconectado el Bluetooth, desconectamos el mismo.

Lo segundo será cargar, como elementos del *Listpicker*, todos los dispositivos ya vinculados por Bluetooth al móvil:

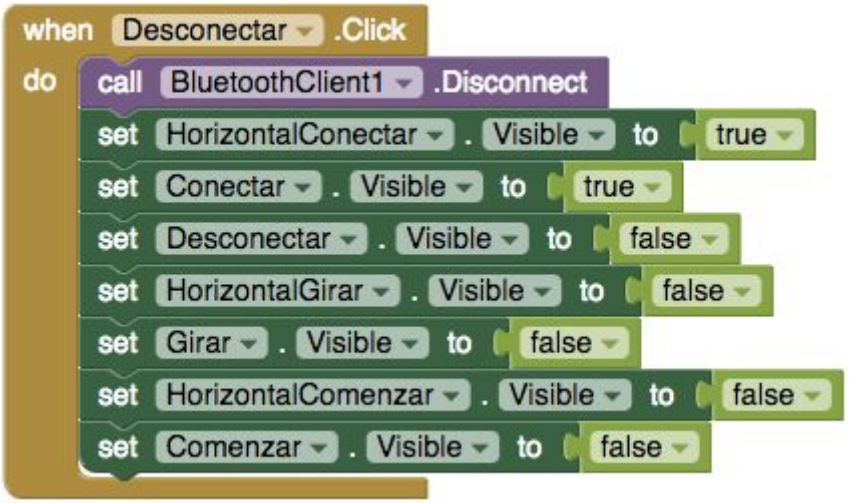
Para que los dispositivos aparezcan en la sección *AddressesAndNames* del Bluetooth **el Bluetooth del dispositivo debe estar activado** y, además, **debe haber sido vinculado al mismo con anterioridad,** desde las opciones de Bluetooth del dispositivo.

Cuando se pulse el *Listpicker*:

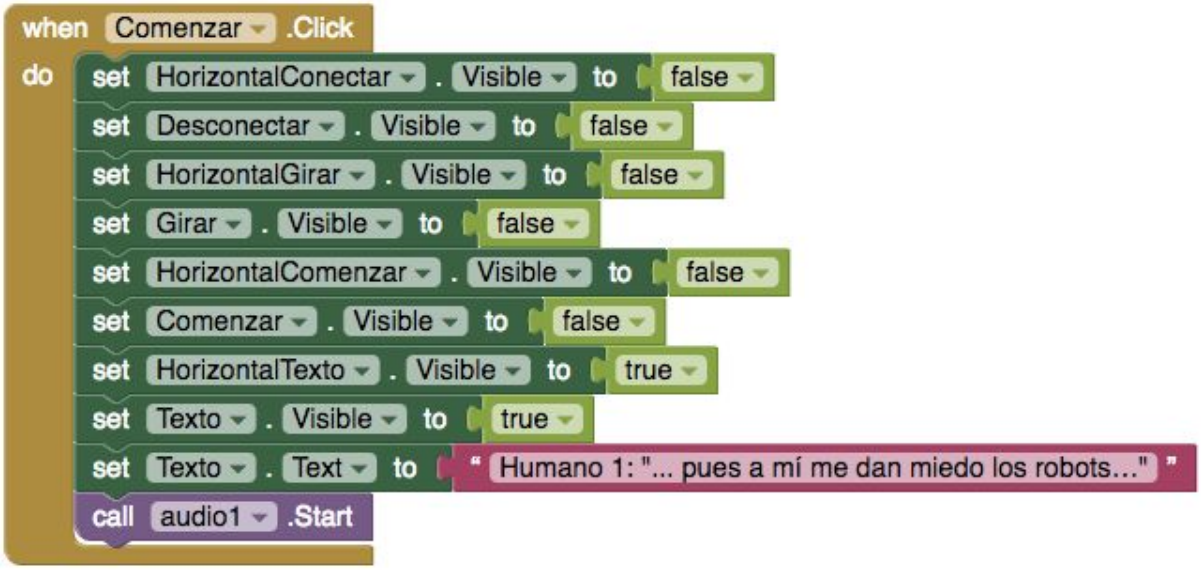


Al pulsar el *Listpicker* se desplegará nuestra lista de Bluetooth disponibles y se conectará el dispositivo al seleccionado, siempre y cuando sea posible. Una vez conectado se ocultará el *Listpicker Conectar* y se mostrarán los botones de *Comenzar, Girar* y *Desconectar*.

El botón *Desconectar* es importante para, en caso de tener problemas con la conexión Bluetooth, desconectar y poder volver a conectar. Por eso, si se pulsa en desconectar, se debe desconectar el dispositivo y volver a mostrarse el *Listpicker Conectar* para poder volver a realizar la conexión:

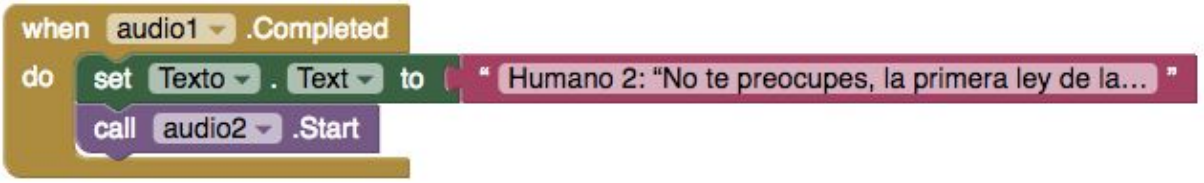


Toca ahora programar el botón *Comenzar*:



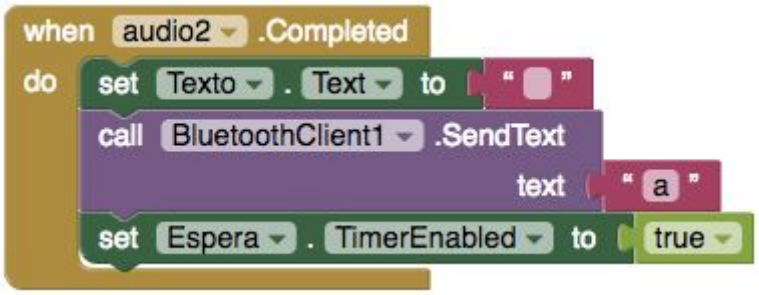
Se mostrará únicamente la etiqueta *Texto* con el texto del primer diálogo, quedando el resto de elementos no visibles y se ejecutará el primer diálogo en audio.

El siguiente evento se lanza cuando concluya el *audio1*:



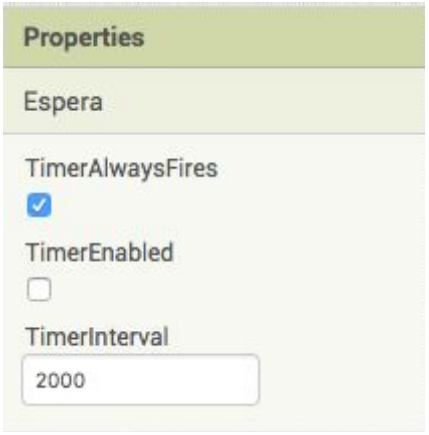
Al terminar el *audio1* se cambiará el texto mostrado en la etiqueta al segundo diálogo y se ejecutará el *audio2.*

Al terminar el *audio2* se lanzará el siguiente evento, el cual no mostrará ningún texto (ya que se han terminado los dos primeros diálogos y hay que girar el escenario) y enviará el primer mensaje de Bluetooth a la placa Arduino. El mensaje enviado será la letra “a”, la cual usaremos en la programación de la placa. Dicho mensaje enviado activará el giro del escenario y no podemos iniciar un diálogo hasta que no se haya terminado el giro, por lo que activaremos el *Clock* (o reloj), llamado *Espera,* para que empiece a marcar ciclos:

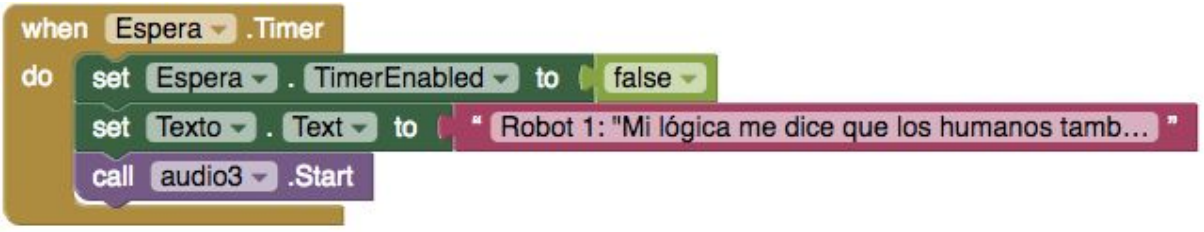


El reloj de AppInventor marca un ritmo, activando un evento cada vez que se cumple el tiempo que se ha indicado como tiempo de ciclo o intervalo en la sección *Designer*. Si ponemos 10 segundos de ciclo o intervalo, cada 10 segundos se producirá un evento llamado *Timer*.

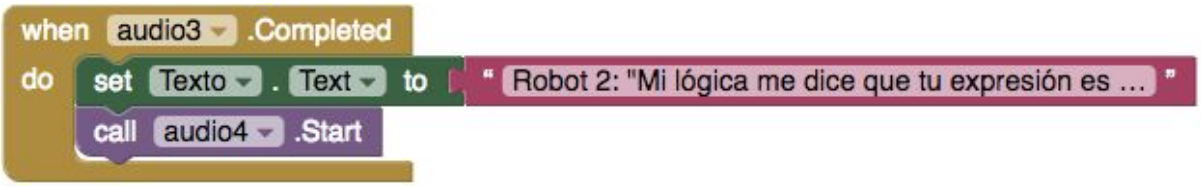
Se debe cambiar el *TimerInterval* en la parte de *Properties*. El *TimerInterval* es una cantidad en milisegundos. Seguramente con 2 segundos (2000 milisegundos) sea suficiente para activar el evento:



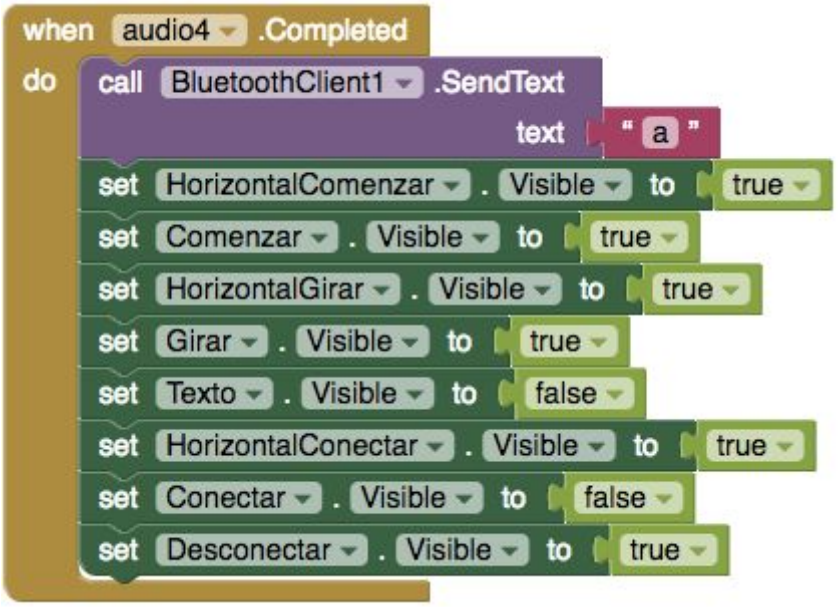
Al activar el reloj, se inicia una cuenta de 2 segundos de duración y cuando se termina, se activa el evento llamado *Timer*:



Cuando se cumpla el primer intervalo o ciclo de reloj, lo desactivaremos para que no vuelva a producirse el evento. Cambiamos el texto de la etiqueta al tercer diálogo (el primero tras girar el escenario) y se activará el *audio3*, el cual activará el nuevo evento al finalizar:



El evento mostrará en la etiqueta el último diálogo (en nuestro ejemplo, puede que en el de tus alumnos aún haya más diálogos) y se activará el *audio4* que, al finalizar, activará el evento final:



El último evento enviará a la placa el mensaje “a” para que vuelva a girar el escenario y ponerse como al comienzo y ocultará la etiqueta *Texto*, mostrando los botones de *Girar, Desconectar* y *Comenzar*.

Lo último que queda es programar qué evento activa el botón *Girar*. Lo único que se necesita es enviar el mensaje “a” a la placa para activar el giro del escenario. Este botón es necesario para poder colocar el escenario de la forma correcta para comenzar en caso de que no esté bien colocado:

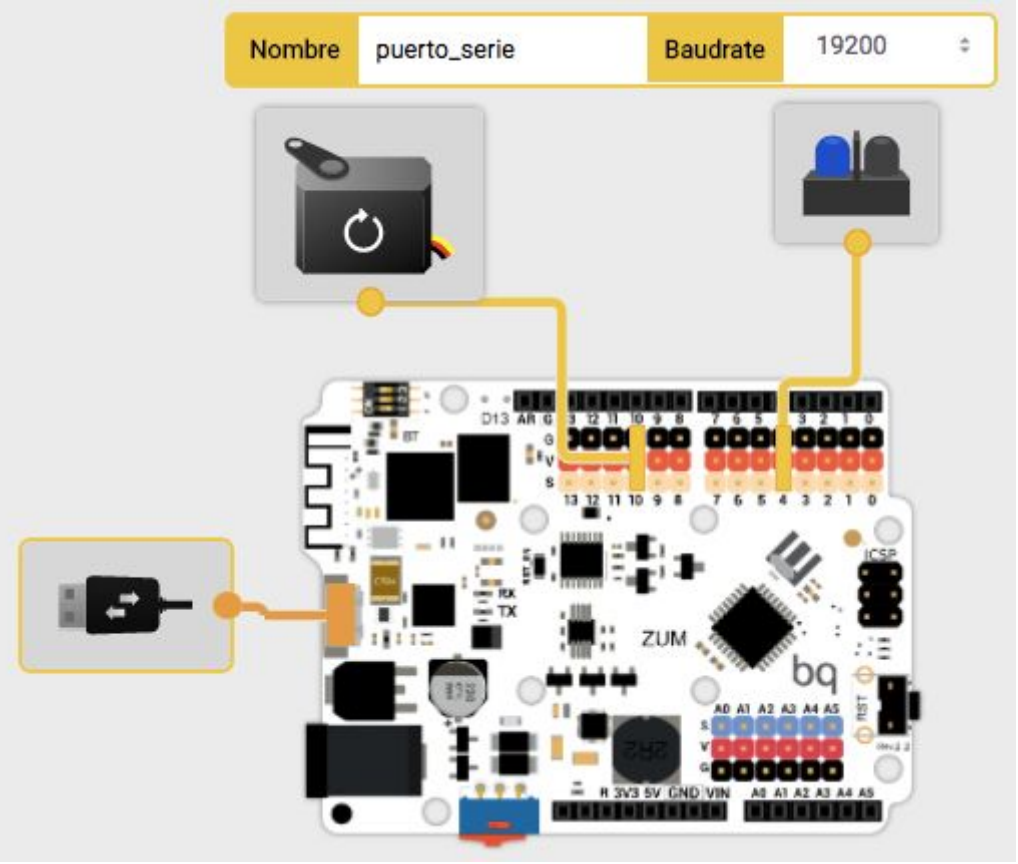
Cuando esté todo programado se compilará la aplicación e instalará en el dispositivo móvil. Para compilar se debe ir al menú superior *Build* y en el mismo seleccionar *App (provide QR code for .apk)*. Ésta opción generará un código QR que

se captura desde la aplicación AppInventor en el dispositivo y se accede a una página web para instalar la aplicación creada.

Modificación de la programación Arduino para usar la aplicación móvil

Una vez se tiene la aplicación móvil lista, toca cambiar la programación en Bitbloq para que puedan conectarse el dispositivo móvil y la placa Arduino.

Hay que cambiar algunas cosas en Bitbloq:

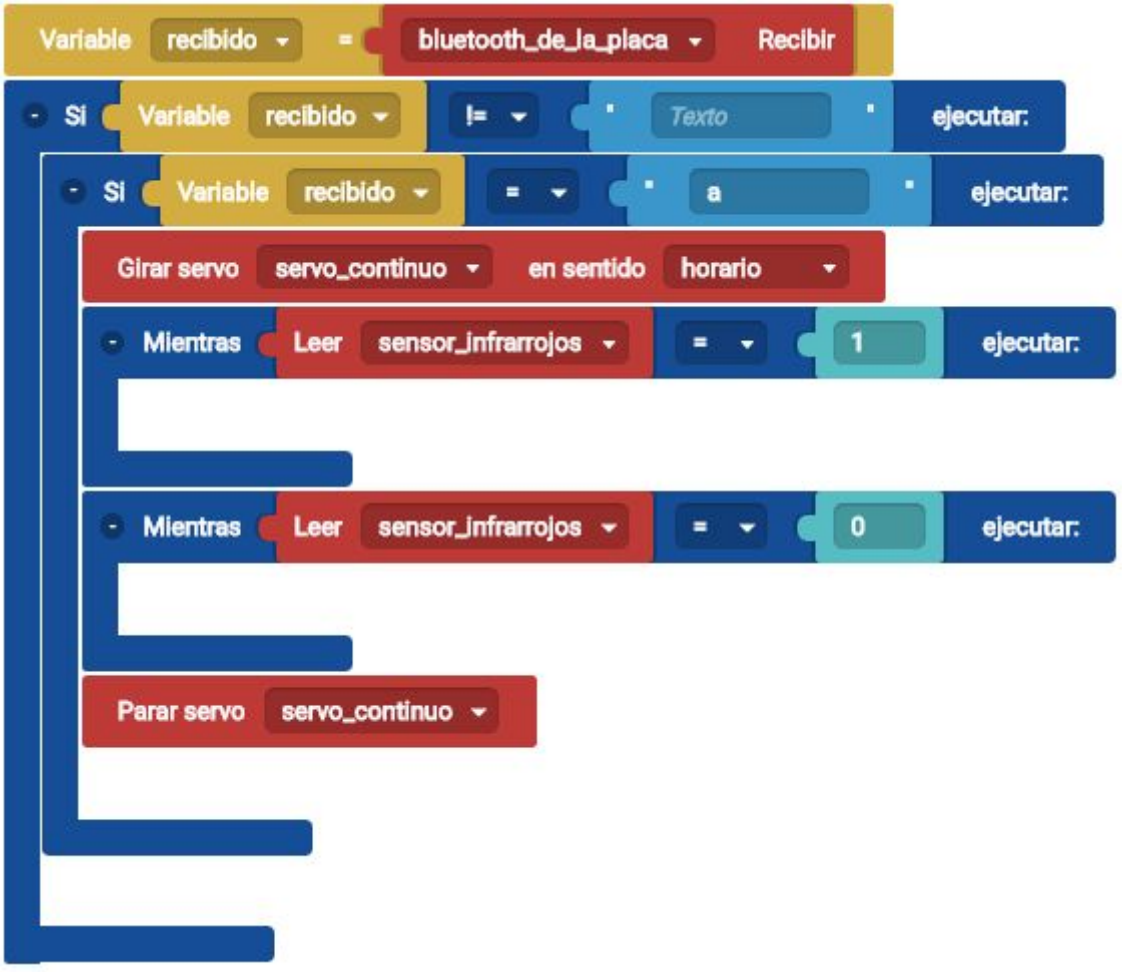


Se eliminará el botón y añadirá un *puerto_serie* para que la placa se pueda comunicar con el Bluetooth del móvil. Se cambiará el *Baudrate* a 19200, ya que es la frecuencia de comunicación de los dispositivos móviles y si no se comunican al mismo ritmo será como si el dispositivo móvil hablase alemán y la placa Arduino hablase turco.

Una vez cambiado toca realizar la nueva programación. Se crea una variable para almacenar lo que llegue desde el Bluetooth del dispositivo móvil:

Como contenido inicial de la variable vamos a poner un texto vacío, de forma que la variable será de tipo *String* (cadena), que permite almacenar texto.

En el *Loop*, vamos a empezar almacenando lo que se recibe del Bluetooth y, en caso de haber recibido algún carácter, se analizará para girar el escenario si lo recibido es una "a". En cada ciclo del *Loop* la variable *recibido* va a almacenar lo que reciba del Bluetooth de la placa. Si no recibe nada, la variable tendrá como contenido un texto vacío (es decir, " "). Es muy recomendable sólo comprobar qué hay que hacer con lo recibido en caso de recibir algo diferente de nada (usamos la expresión *Si variable recibido != " "* que significa *Si la variable recibido contiene algo*):



Cargaremos el programa en la placa, conectaremos la aplicación móvil y la emparejaremos con la placa de Arduino. Una vez hecho la aplicación móvil y la programación de la placa deberían funcionar correctamente.

Detalles finales

Una vez esté todo el proyecto funcionando, toca terminar todo lo que sea necesario para que la maqueta tenga el mejor aspecto posible. Un buen proyecto no es aquel que tiene la mejor programación, sino aquel que presenta un equilibrio entre solución adoptada, buena programación y buen aspecto y terminación.

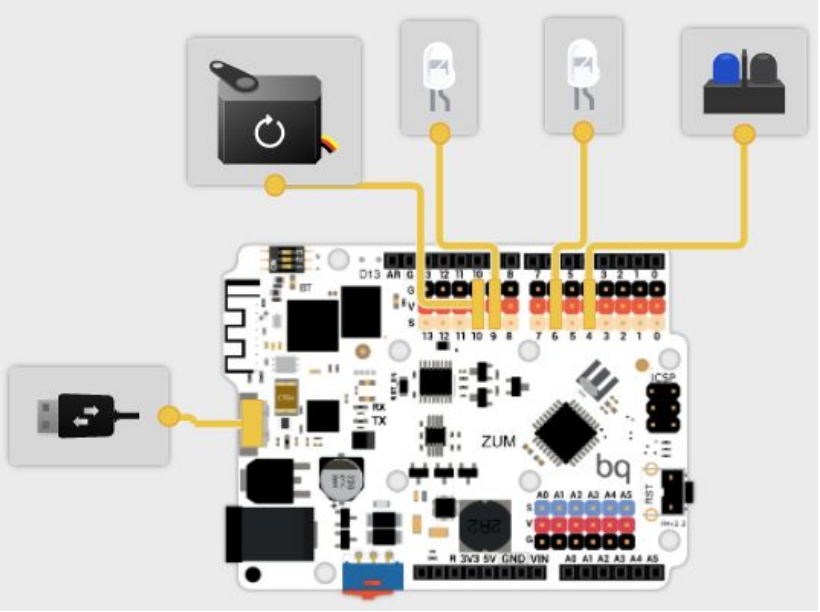
Se pueden pintar los bordes del escenario del color que tenga el fondo y el suelo, también pintar algunas de las piezas impresas si el alumno cree que quedará mejor, pegar todo lo que falte del escenario, etc.

Si tras dejar todo terminado aún hay tiempo de mejorar el proyecto, vamos a proponer algunos retos.

Iluminar a los personajes

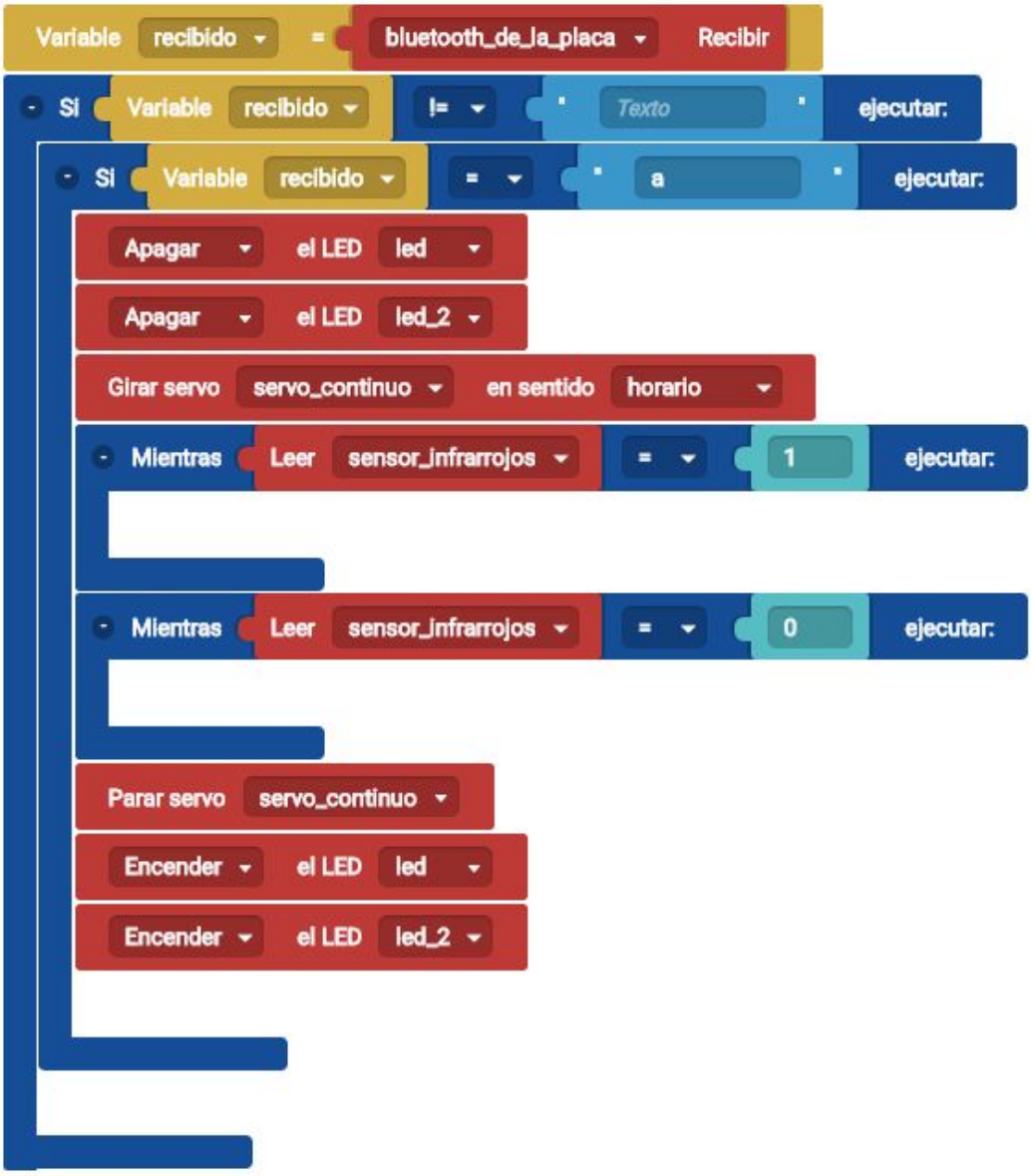
La primera mejora que se propone es incluir dos LED que iluminen a los personajes. Se incluirá, en la programación, dos LED que se enciendan tras terminar el giro del escenario.

Para ello, conectaremos un LED en el pin digital 6 y otro en el pin digital 9:



Se añade el encendido de los LED en la parte de programación para que estén encendidos iluminando los personajes al encenderse la placa:

Los LED están puestos en el *Setup*, con esta programación los LED se encenderán una vez iniciado el programa. El problema es que no se van a apagar nunca. Podríamos apagar los LED al menos durante el giro del escenario, para ello habrá que hacerlo justo antes del giro y volverlos a encender tras el mismo:



Hay que colocar los LED de alguna forma en el escenario. Esta parte el alumno tendrá que idearla totalmente, buscando la forma de colocar los LED para que iluminen a los personajes. Se puede hacer algún agujero en el suelo para pasar el cable y colocarlo con alguna pieza diseñada e impresa o bien construyendo un soporte en cartón. También puede estar en el fondo fijo del escenario a los lados de los personajes, o sobre ellos…

Iluminar únicamente al personaje que habla

Vamos a complicarlo un poco más. Al ser los dos personajes fijos, no queda claro en ningún momento cuál de los dos está hablando. Podemos hacer que sólo se encienda el LED del personaje que está hablando.

Los diálogos se inician y finalizan desde la aplicación móvil, por lo que para producir los encendidos y apagados de los LED vamos a utilizar la aplicación, que es la que nos permite utilizar el comienzo y final de un diálogo.

Lo primero de todo es ver qué nuevos estados del sistema robótico tenemos con esta propuesta. Dado que hay un LED para cada personaje y suponemos que no van a hablar a la vez ambos personajes, tenemos tres nuevas posibilidades, a las que vamos a nombrar con nuevas letras (recuerda que la letra “a” era la que producía el giro):

- LED 1 encendido y LED 2 apagado → “b”
- LED 1 apagado y LED 2 encendido → “c”

Si pensamos un poco en cómo funciona nuestro programa, los LED se van a apagar tras escuchar los diálogos, pero también es el momento en el que se va a producir el giro del escenario, por lo que podemos hacer coincidir que los LED se apaguen con el giro de la placa y no será necesario incluir una nueva letra:

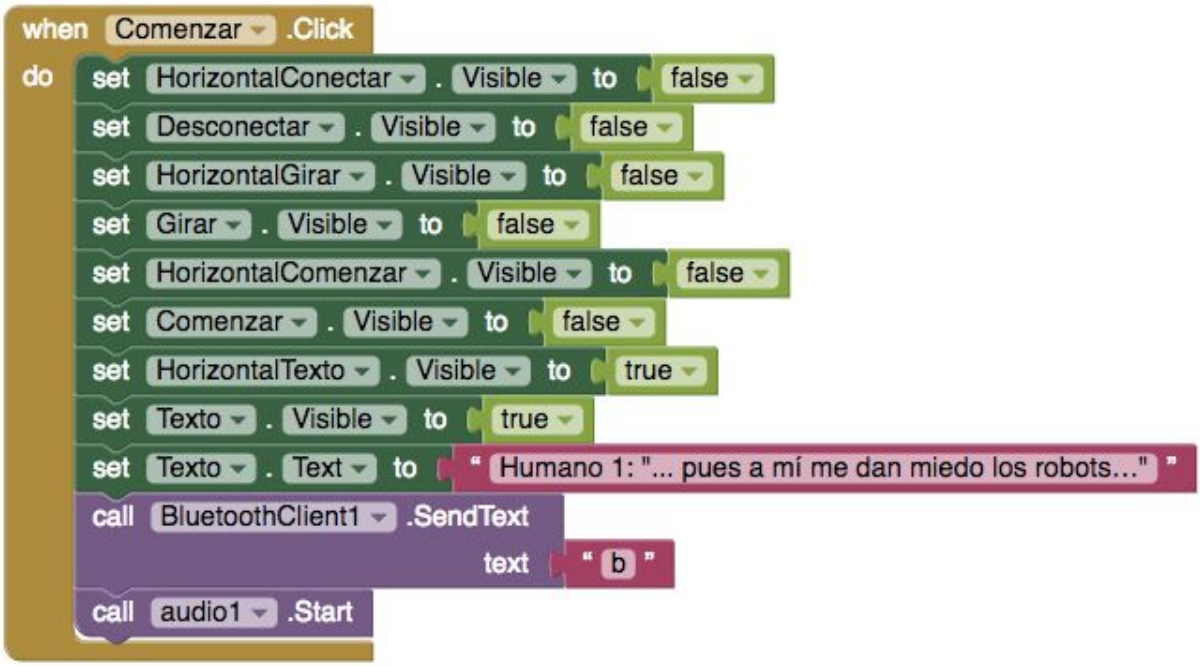
- LED 1 y LED 2 apagados → “a”

Es muy importante intentar, siempre, **optimizar los programas**, utilizando la menor cantidad de información, envíos y estados posible. En Arduino no es habitual tener problemas de espacio en memoria o velocidad de ejecución en programas, pero en un programa informático este tipo de detalles hacen que sea más o menos rápido y, por lo tanto, más o menos útil para el usuario.

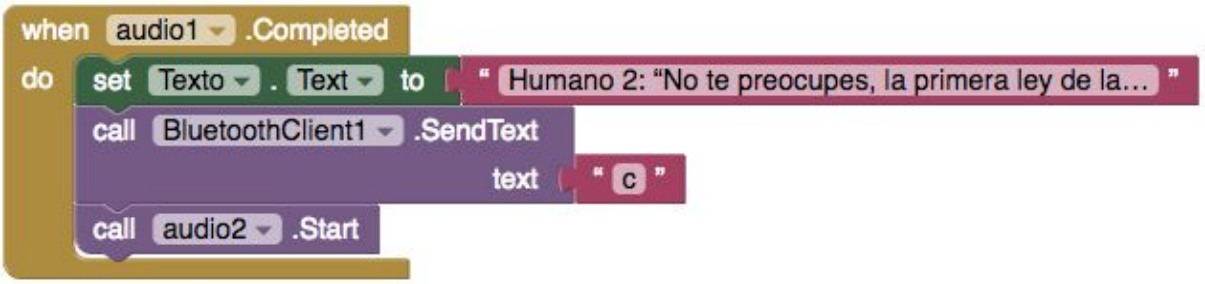
Vamos, pues, a incorporar las letras a la programación de AppInventor. Para nuestro proyecto, el LED 1 iluminará al humano 1 y al robot 1 y el LED 2 iluminará al humano 2 y al robot 2. Si se han cambiado los personajes y/o diálogos se debe tener claro qué personaje habla en cada momento para iluminarlo.

De inicio van a estar todos los LED apagados y encenderemos el que corresponda únicamente mientras habla el personaje al que ilumina. Es un proceso laborioso pero sencillo, sólo se debe ir enviando la letra correspondiente con AppInventor justo antes de empezar un diálogo y justo al finalizarlo.

Antes de empezar el *audio1* enviaremos la letra “b”. Sólo hay que añadir, en el bloque que muestra la imagen, un *call BluetoothClient1* con el texto “b”:

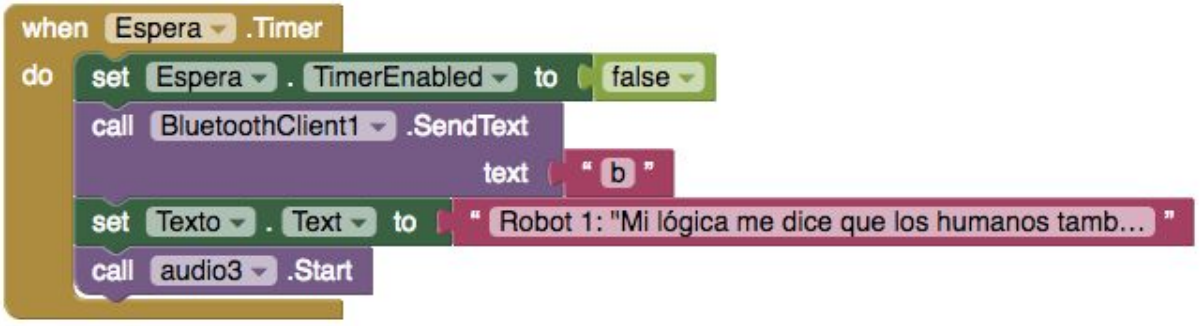


Al finalizar el *audio1* enviaremos la letra “c”:

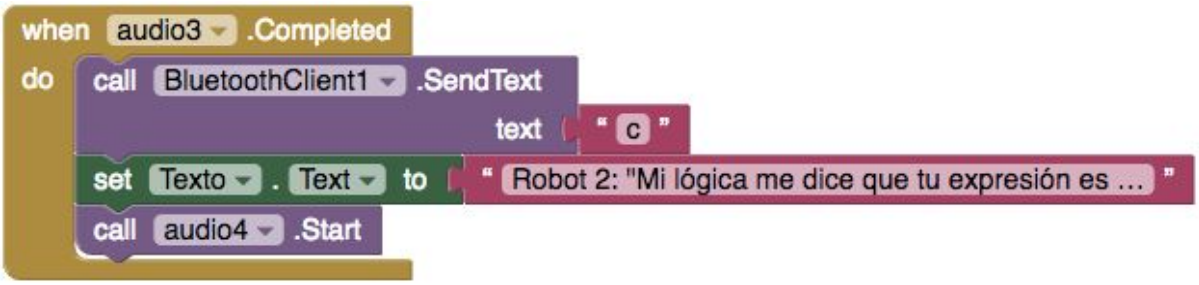


Cuando termine el *audio2* llega el momento de girar el personaje, así que tendremos que apagar ambos LED. Como hemos dicho anteriormente no hay que enviar ningún nuevo mensaje, simplemente apagaremos en Bitbloq ambos LED cuando se vaya a producir el giro.

El siguiente cambio llega justo antes de iniciarse el tercer diálogo, que se iniciaba con el *Timer* del reloj. Enviamos en ese momento la letra “b” de nuevo:



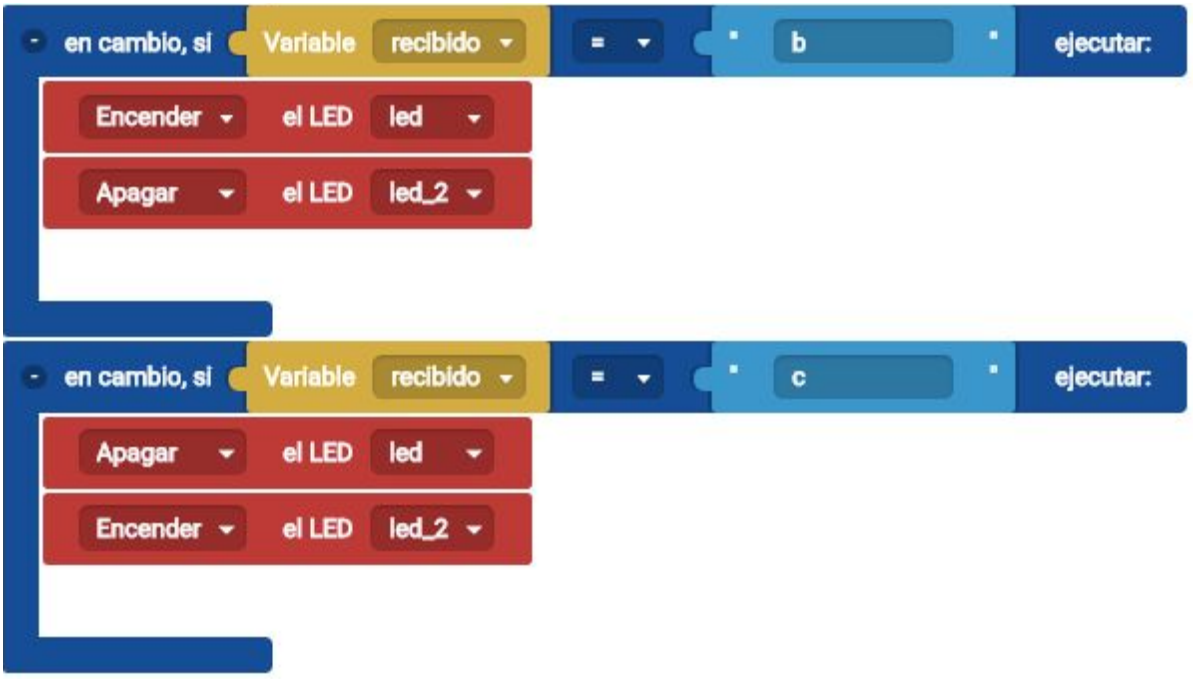
Al terminar el *audio3* se iniciará el cuarto diálogo y tendremos que enviar la letra “c”:



Por último, al terminar el *audio4* se va a producir de nuevo el giro, por lo que no hace falta enviar una nueva letra para apagar los LED. Se instalará la aplicación con los cambiosen el dispositivo móvil.

Vamos ahora a cambiar la programación en Bitbloq para actualizarla a las nuevas posibilidades que hemos incluido en AppInventor.

En el programa de Bitbloq que tenía el control de la placa, se eliminará el encendido de los LED del *Setup* y, tras ello, lo único que hay que hacer es incluir nuevos condicionales para la letra “b” y la letra “c”, justo debajo del de la letra “a”:



Una vez se hayan introducido todos los cambios, se cargará el programa en la placa y probará con la aplicación móvil que el teatro funciona correctamente.

Controlar el brillo de los LED

La última mejora que te vamos a proponer es conseguir que el LED brille a la intensidad que quiera el usuario, manejando el brillo con un **potenciómetro.**

Los LED del kit de robótica apenas varían su intensidad de brillo (no se puede decir que no la varíen, pero sí que no es apreciable para el ojo humano). Pero, podemos realizar un pequeño truco para que el LED, aparentemente, brille más o menos. El truco es producir **parpadeos** del LED a una velocidad tan grande que el ojo humano lo aprecie como un cambio de intensidad de la luz.

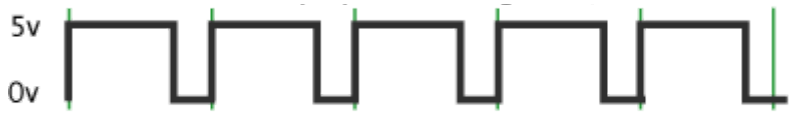
Si el LED, en su parpadeo, está más tiempo encendido que apagado, parecerá que brilla más. Por el contrario, si está más tiempo apagado que encendido, parecerá que brilla menos. El tiempo que está encendido y apagado es muy muy pequeño.

En Arduino, hay una serie de **pines digitales** que permiten que conectemos la corriente de 5 voltios (es decir, encendamos el LED) y desconectemos la corriente (es decir, apaguemos el LED) muy rápido y de forma continua. Dichos pines se denominan **pines PWM**, de *pulse-width modulation* o, en castellano, modulación por ancho de pulsos. No todos los pines de una placa de Arduino aceptan la salida PWM, sólo los pines 3, 5, 6, 9, 10 y 11, por eso anteriormente conectamos los LED en los pines 6 y 9.

Veamos una imagen que ilustra cómo funciona una salida digital PWM:

En la gráfica anterior, la señal eléctrica está mucho menos tiempo en 5 voltios (encendida) que en 0 voltios (apagada), por lo que el LED brillará poco. Cada línea verde vertical separa la señal, que es cíclica (se repite). Un conjunto de HIGH (5 voltios) y LOW (0 voltios) se denomina ciclo.

En la gráfica anterior, la señal eléctrica está el mismo tiempo en 5 voltios (encendida) que en 0 voltios (apagada), por lo que el LED brillará en una intensidad media.



En la gráfica anterior, la señal eléctrica está mucho más tiempo en 5 voltios (encendida) que en 0 voltios (apagada), por lo que el LED brillará bastante.

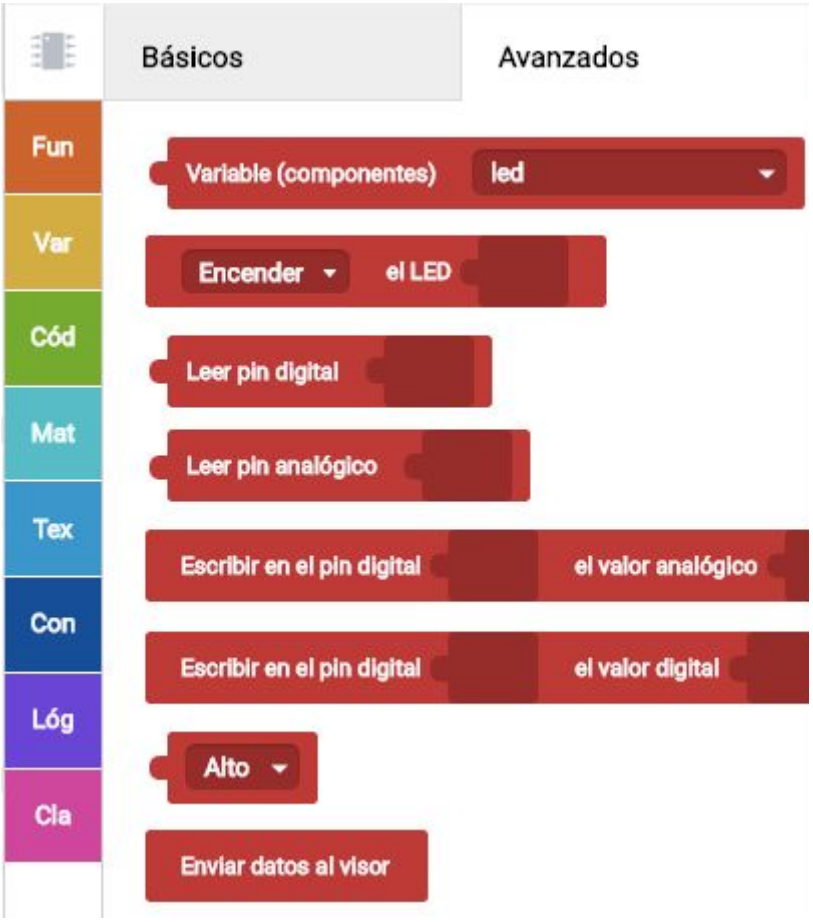
El ciclo de encenderse y apagarse se produce muchas veces por segundo y de esta forma podemos engañar a nuestro ojo y hacer que parezca que el LED brilla más o menos.

Ahora bien, el número de posibilidades PWM que podemos asignar a una salida digital van desde 0 hasta 255, siendo el número elegido la cantidad del tiempo de cada ciclo que está la señal en 5 voltios. Si escribimos un PWM de valor 0, estaremos diciendo que la señal esté un 0% del tiempo de ciclo en 5 voltios y un 100% del tiempo en 0 voltios, por lo que el LED siempre estará apagado.

En cambio, si elegimos el valor 255 estará un 100% del tiempo del ciclo en 5 voltios y un 0% del tiempo en 0 voltios, por lo que el LED estará siempre encendido.

Si escribimos, por ejemplo, 128, estará prácticamente el 50% del tiempo en 0 voltios y el otro 50% del tiempo en 5 voltios, por lo que el LED brillará a una intensidad media.

Por lo que, para que el LED brille a diferentes intensidades hay que escribir en un pin digital PWM un valor entre **0 y 255**. El bloque para escribir en un pin está en **componentes avanzados:**



Como es un valor de 0 a 255, se trata de un valor analógico. Un valor digital es aquel que sólo toma 2 estados: 0 o 1.

Para escribir en un pin digital un valor analógico se debe construir un bloque como éste:

Ahora bien, en la propuesta inicial se quería **regular la intensidad de brillo con un potenciómetro.** No nos sirve escribir un valor, tendríamos que poder escribir el valor analógico que marcase el potenciómetro.

Lo primero de todo será conectar un potenciómetro a un pin analógico de la placa y conectarlo también en la parte de **hardware** de Bitbloq en el mismo pin analógico.

El único problema es que el potenciómetro adopta valores entre 0 y 1023 y el pin PWM sólo acepta valores entre 0 y 255. Tenemos que distribuir los 1024 posibles valores del potenciómetro entre los 256 diferentes valores que acepta el pin PWM... por suerte existe una función de Arduino que hace exactamente lo que queremos. Dicha función es **mapear.** Mapear sirve para distribuir una cantidad de valores

agrupándolos equilibradamente en grupos, de forma que varios de los valores iniciales se correspondan con un único valor.

Dicho esto, lo que necesitamos es básicamente construir el siguiente bloque:

Ahora sólo tienes que sustituir los *Encender LED* por ese bloque. *Apagar LED* se puede dejar como estaba, no necesitamos apagarlo por PWM. Se cargará el programa en la placa y probará, activando desde la aplicación móvil el teatro, que se puede regular con el potenciómetro el brillo de los LED.