



# Programación de Videojuegos

Currículo de Aprendizaje



Desarrollado por IEEE y la Universidad del Norte

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exercitation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iust.

Dio dignissim qui blandit praesent.

# Tabla de Contenidos

1	mBlock .....	1
	Primeros Pasos .....	2
	Taller #1 - Números Correctos, Parte 1 .....	8
	Taller #2 - Números Correctos, Parte 2 .....	18
2	C# .....	25
	C#: Primeros Pasos .....	26
3	Arduino .....	31
	Configuración y Monitor Serie .....	32
	Guía Básica .....	34
	Taller #1 .....	41
	Taller #2 .....	45
	Taller #3 .....	49
4	Unity .....	53
	Introducción a Unity .....	54
	Taller Juego Quiz Múltiples Respuestas en Bolt-Unity .....	60



mBlock



# mBlock: Primeros Pasos



**Nota:** Omitir el primer y segundo paso si no quieres instalar el complemento mLink para ver el código python.

Para poder ver el código python con mBlock, primero se deben instalar un complemento y después crear un usuario en la plataforma virtual.

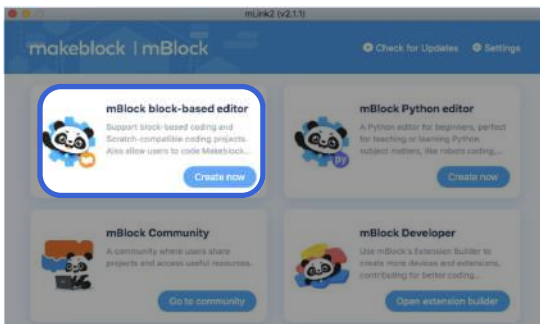
- 1 Ingresa a la página <https://ide.mblock.cc/#/> y te saldrá este mensaje para instalar mLink:



El proceso de instalación es simple, sin embargo si tienes dudas puedes consultar las instrucciones en esta página:

<https://mblock.makeblock.com/en-us/download/mlink/>

- 2 Una vez instalado el complemento, te aparecerá una ventana como esta:

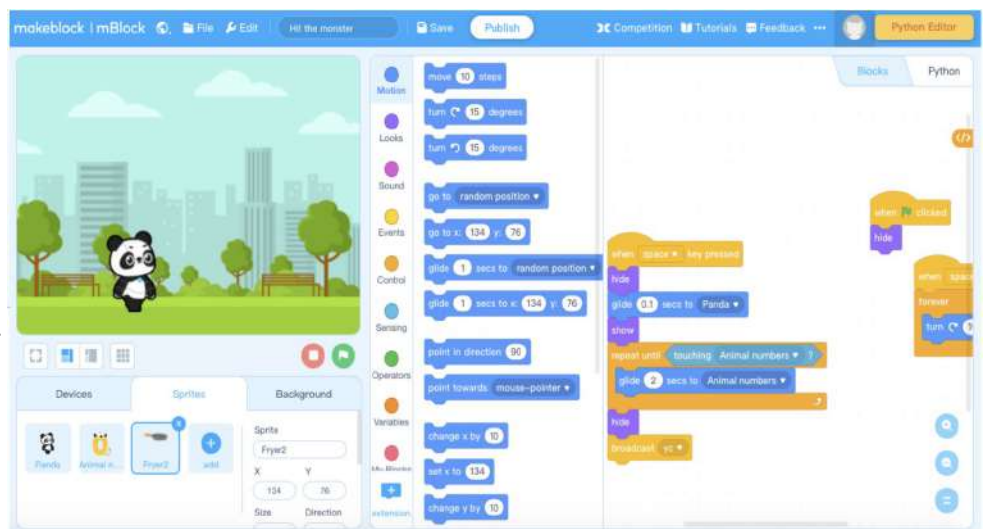


Selecciona la primera opción de **Create Now** y te llevará a la interfaz online de mBlock, que es el mismo link del comienzo:

<https://ide.mblock.cc/#/>

Ahora podrás ver la **interfaz de mBlock** y descubrir las herramientas que te ayudarán a desarrollar tu propio juego:

Interfaz de la plataforma



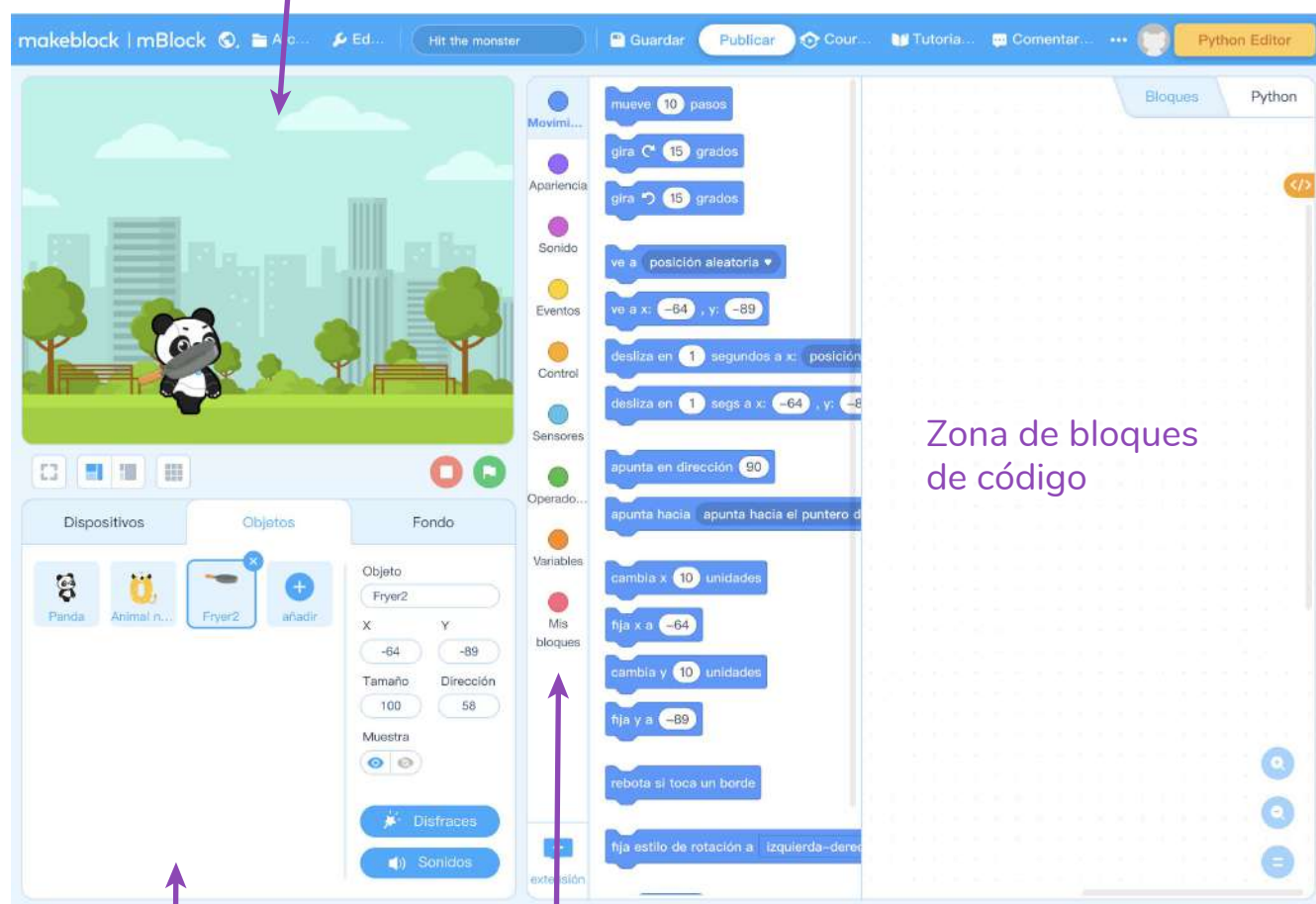
- 3 Ahora deberás crear un usuario para trabajar y guardar **tus proyectos**. En la parte superior de la interfaz te aparecerá el ícono de un panda gris en un círculo (ver imagen de la derecha). Al seleccionarlo se abrirá una ventana emergente.



Aquí seleccionar en **Signup** (Registrarse) y te piden email y rango de edad. Ingresas tu correo electrónico y una contraseña. Te enviarán un código de verificación al correo. Tienes que ingresar el código en los siguientes 60 segundos, sino caducará y deberás pedir uno nuevo.

- 4 Para el desarrollo y para trabajar de una mejor manera conozcamos ahora las partes que componen la interfaz principal de mBlock:

Vista de nuestro juego



Objetos

Secciones de Bloques



## 4.1 Las secciones de bloques



**Movimiento:** Estos bloques se utilizan para darle movimiento a los objetos sobre los cuales se utiliza

**Apariencia:** Bloques que permiten modificar la apariencia de los objetos, también permiten cambiar el estado de los objetos de mostrarse o no hacerlo.

**Sonido:** Bloques que permiten modificar el

**Eventos:** Contiene bloques que permiten detectar acciones como, clicks, presionar el teclado, detectar el ratón del PC.

**Controles:** Estos bloques permiten controlar los tiempos de ejecución de las acciones; es decir, te permite ponerle condiciones.

**Operadores:** Contiene bloques que permiten realizar operaciones matemáticas, principalmente para realizar comparaciones entre números y demás operaciones similares.

**Sensores:** Te permiten controlar las acciones de los personajes según en donde se ejecuten el espacio que se les determine.

**Variables:** En esta sección se pueden crear variables para los juegos y contienen bloques que permite modificar estas variables.

**Mis bloques:** Permite crear nuevos bloques, suele ser para usuarios más experimentados.

## 4.2 Objetos

En esta parte encontramos todos los **Objetos** que utilizamos en el juego, desde aquí se pueden añadir nuevos objetos y modificar las características principales de estos. también, en esta parte podemos acceder a los fondos, para modificarlos.



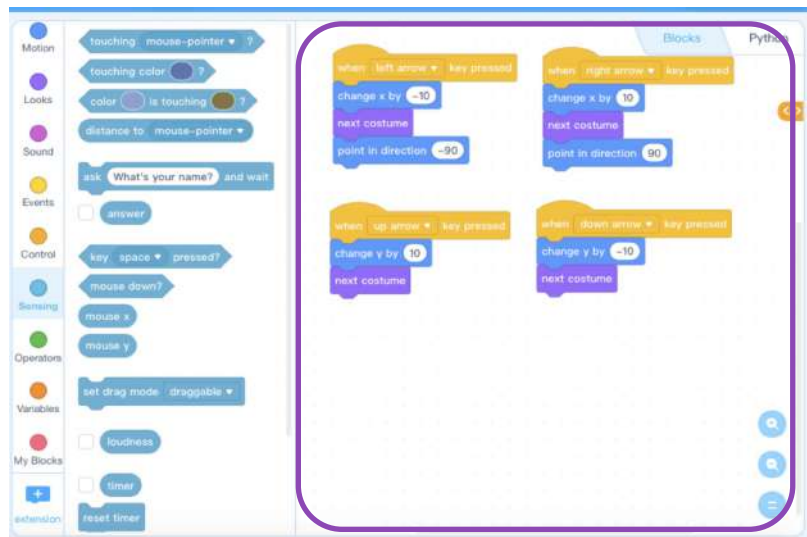
## 4.3 Vista del Juego

Aquí se muestra una previsualización de nuestro juego. Muestra las acciones, fondos, personajes y sus disfraces que se les ha atribuido al juego. Hay dos botones: el verde sirve para ejecutar el juego (acciones y eventos) y el rojo es para detener dicha ejecución.



#### 4.4 Zona de Bloques

En este espacio se colocan los bloques de código. Tan solo debes escoger el que necesites y arrastrarlo hacia la zona.

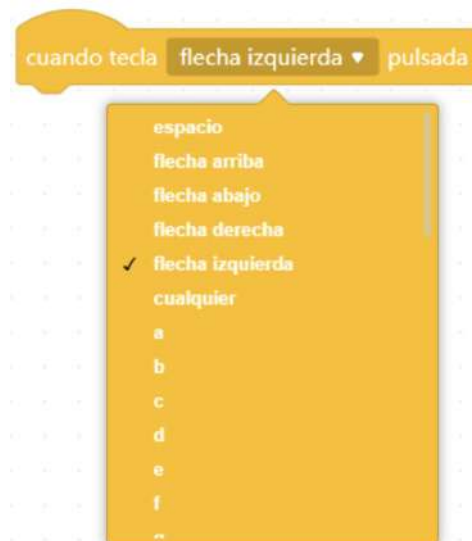


#### 5 Algunos bloques importantes

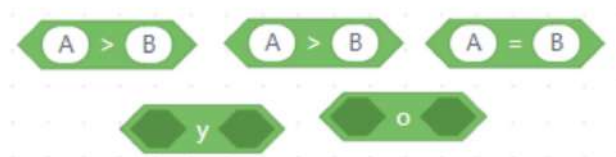
**5.1 Bloque cuando clic en banderita verde:** Este bloque nos marca el inicio de ejecución de todos los bloques.



**5.2 Bloque de evento cuando tecla <> pulsada:** Este bloque permite ejecutar una conjunto de bloques cuando se pulsa la tetcla que le indiquemos.

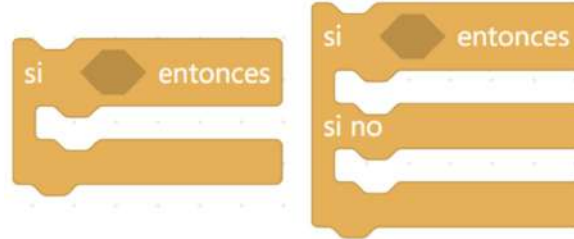


**5.3 Bloques de condiciones:** estos bloques representan operaciones lógicas, que dicen si algo es cierto o no.





**5.4** Bloques **si <> y si <> sino**: Estos bloques solo ejecutan los bloques que consienten solamente si la condición que se está evaluando es verdadera.



**5.5** Bloques de los ciclos: **repite hasta**, y **repite siempre**: todos los bloques que se usen dentro de estos bloques se ejecutan tantas veces como el bloque lo indique, en repite se define por medio de un número, en repite hasta que evalúa una condición y el **para siempre**, se repite infinitamente.



**5.6** Bloque **espera**: Este bloque detiene la ejecución durante el número de segundos que se indique, y luego continúa.



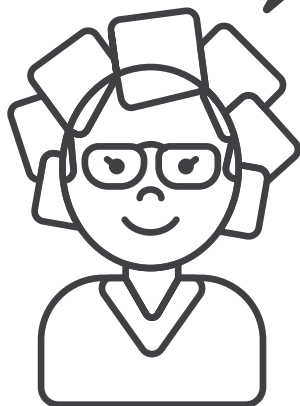
**5.7** Bloques para cambiar posición y dirección: Son los bloques más comunes para cambiar de posición a los objetos, usando los ejes X y Y. Para esto es importante tener en cuenta, en que la vista del juego el punto (0,0) se encuentra en el centro de la pantalla, y de ahí se aumenta o disminuye la posición de los elementos.



**5.8 Bloque de sensores:** Estos bloques sirven para modificar el comportamiento del objeto seleccionado cuando llegue hasta cierto punto en el tablero, cuando toque el puntero del ratón o cuando toque otros objetos.



**¡Una vez haya leído este documento estás listo para comenzar con tu primer taller!**



# mBlock: Taller #1

## Juego de los Números Correctos - Parte 1

**Objetivo:** Realizar un videojuego en el que un personaje tenga que atrapar números que van cayendo del cielo, aplicando conceptos de programa, como condicionales y loops.

Plataforma de trabajo: [mblock https://ide.mblock.cc/#/](https://ide.mblock.cc/#/)

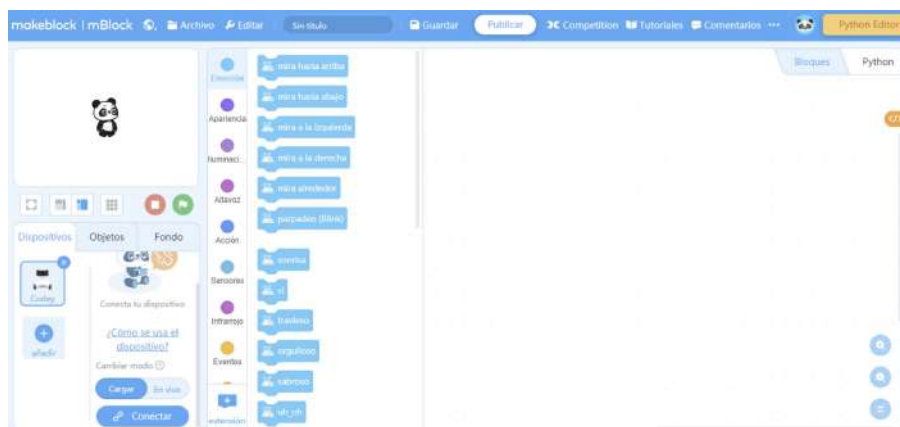
Demo del juego final esperado: [Demo: Primer Juego](#)

### Desarrollo

- 1 **Crear un proyecto nuevo:** Luego de crear nuestra cuenta en mBlock, en el menú superior vamos a seleccionar la opción **Archivo** y luego en nuevo, como se observa en la siguiente imagen:



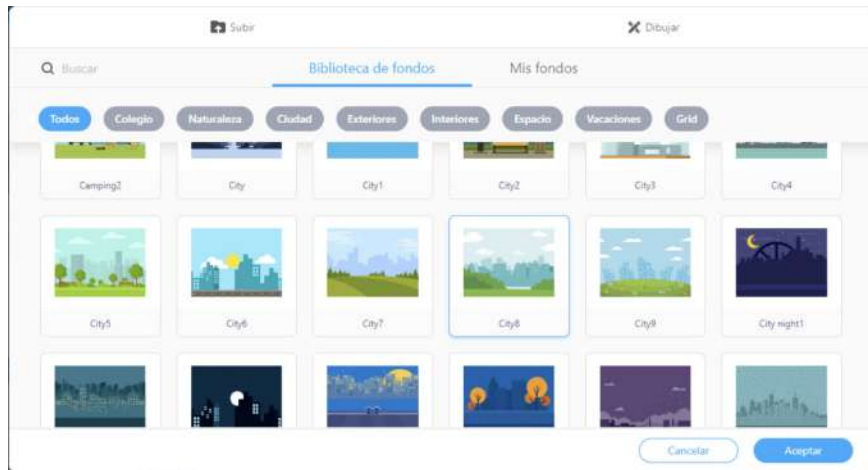
Nos aparecerá un proyecto con un tablero en blanco y por defecto trae un pandita:



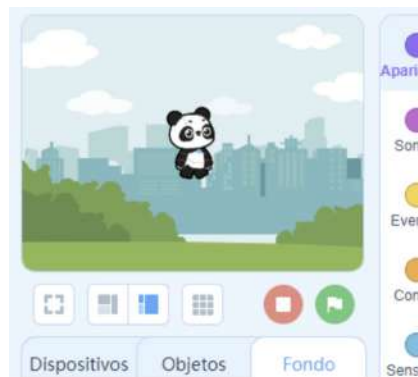
- 2 **Elegir un fondo:** En este paso, vamos a ponerle un fondo a nuestro juego. Para esto, en la parte inferior izquierda del proyecto buscamos la pestaña **Fondo** y seleccionamos el signo más, tal como se ve en la imagen:



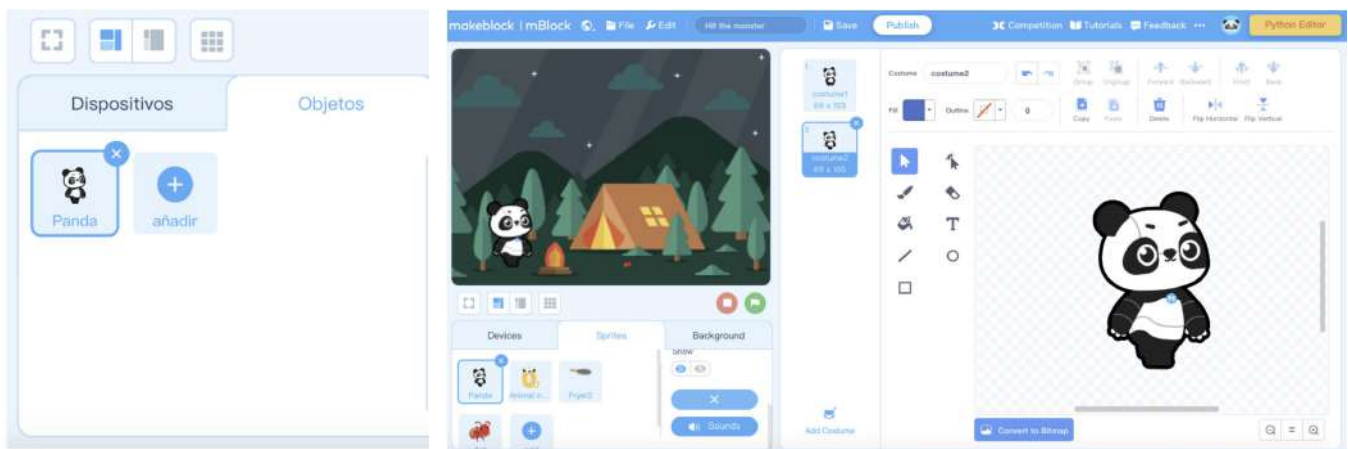
Esto nos llevará a una galería de fondos donde podremos seleccionar el de nuestra preferencia. Finalmente le damos en aceptar:



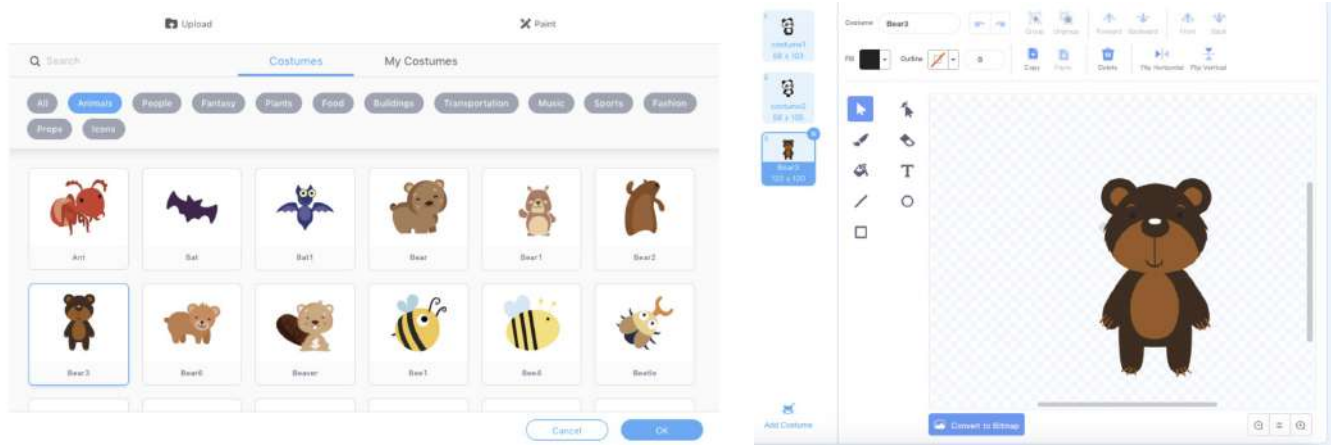
Con esto tendremos nuestro fondo listo para continuar nuestro juego.



**3 Disfraces de los personajes:** Se puede modificar la apariencia de los objetos mediante la opción de disfraces que se encuentra ubicada en la parte de abajo de las opciones del objeto. Al seleccionarla te saldrá el siguiente menú en donde podrás hacer modificaciones a la apariencia de los objetos. Puedes darle a un solo objeto varios disfraces que pueden ser usados para representar acciones, cambiar los colores y ajustar las partes de este para hacerlo cambiar de posición.



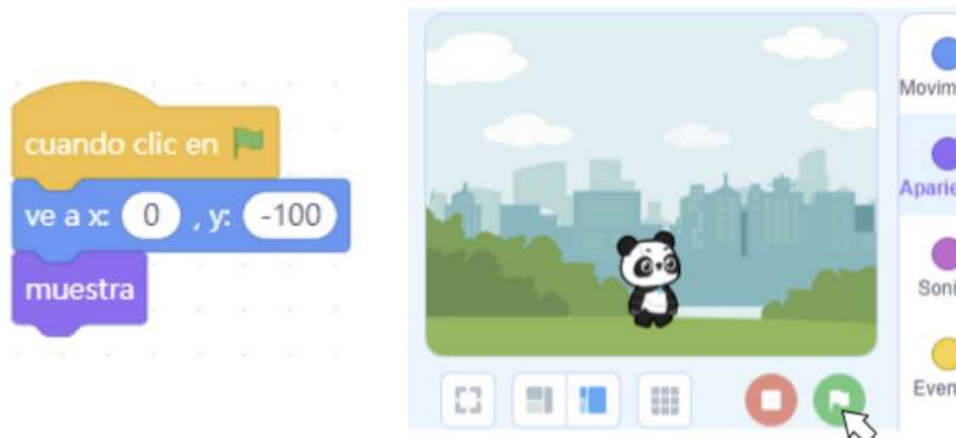
Si quieres cambiar o agregar otro disfraz puedes seleccionar el botón en la parte de abajo de “Agregar Disfraz” en donde te saldrá una galería. Ahí seleccionas el disfraz que necesites o puedes subir los tuyos. Una vez haya elegido, escoges Aceptar para efectuar los cambios.



**4 Darle movimiento a nuestro personaje:** Para darle movimiento a nuestro personaje, primero vamos a ubicarlo en la parte inferior de la pantalla. Nos vamos a **Eventos**, en esta sección tenemos diferentes bloques, el que usaremos inicialmente será cuando click en banderita verde. Lo arrastramos hasta la parte donde organizaremos nuestros bloques.



Abajo de este bloque ponemos el bloque que está en la sección de movimiento llamado **ve a x: 0, y: 0** e inmediatamente debajo de este ponemos el bloque de apariencia llamado **muestra**. Todos los bloques juntos quedarían de la siguiente forma (los valores de x y y puedes cambiarlos a tu gusto, lo ideal es que sean posiciones que tengan sentido para el juego) y le damos en la banderita verde para que nuestro personaje se posicione:



5 **Movimiento en diferentes direcciones:** Primero, debemos tener presente que cada objeto tendrá un código diferente, es decir que a cada uno podremos ponerle distintos bloques sin que afecten al otro. Para darle movimiento al personaje debemos asegurarnos que estemos en el objeto del personaje, en este caso el panda. Luego, vamos a utilizar los siguientes bloque: en la sección evento escogemos el bloque llamado cuando tecla x pulsada y en la sección de movimiento elegimos el bloque llamado cambiar x 10 unidades.



Para el movimiento a la derecha, luego de arrastrar y soltar el bloque en la zona de bloques, primero cambiamos en el bloque de cuando tecla espacio pulsada a espacio por flecha derecha y abajo de este le agregamos el bloque cambiar x 10 unidades. Nuestra zona de bloques se vería así:



Para el movimiento a la izquierda, luego de arrastrar y soltar otro bloque llamado cuando tecla espacio pulsada en la zona de bloques, cambiamos en el bloque de cuando tecla espacio pulsada a espacio por flecha izquierda y abajo de este le agregamos el bloque cambiar x 10 unidades, y en este caso se cambia el 10 por un -10 (este bloque debería quedar cambiar x -10 unidades). Ahora, toda nuestra zona de bloques se ve así:





Con esto si volvemos a darle click a la banderita verde debajo de la vista de nuestro juego y luego presionamos las flechas izquierda o derecha de nuestro teclado, el personaje debería moverse, si esto no sucede, revise nuevamente los pasos anteriores.

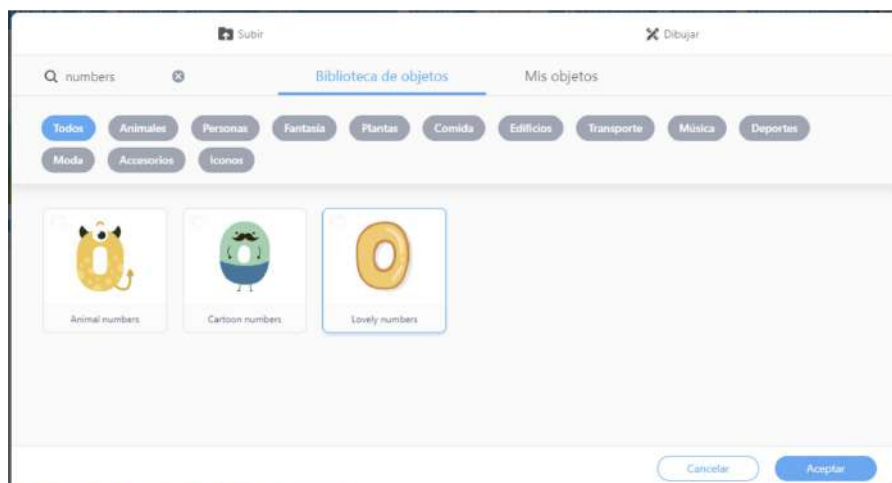
Para que nuestro movimiento se vea más natural agregamos los bloques del siguiente disfraz de la sección de apariencia y apuntar en dirección (-90 si es para la izquierda 90 si es para la derecha) de la sección movimiento. Quedando de la siguiente manera.



- 6 **Añadir otros objetos al proyecto:** Si queremos añadir otros objetos a nuestro juego debemos ir a la parte inferior izquierda, nos vamos a la pestaña objetos y clickeamos en el signo añadir, tal como se ve en la imagen.



Esto nos desplegará una galería de objetos, como nuestro juego será sobre el número correcto, buscamos numbers en la galería. los resultados que parecen son los siguientes:



Podemos escoger el de nuestra preferencia, finalmente damos en aceptar y tendremos ya nuestro nuevo objeto agregado. Luego de esto podemos cambiarle sus características, como tamaño, posición en el [eje x](#) [posición en el eje y](#), el nombre del objeto y la dirección es hacia donde apunta.

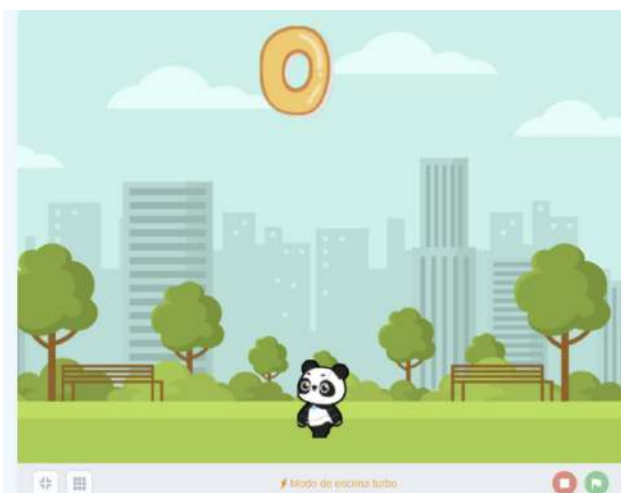
**Nota:** Para el demo se usó un tamaño de 60 en ambos objetos.



**7** [Hacer que nuestros números caigan:](#) Para hacer esto vamos a introducir un nuevo bloque de la sección de control llamado para siempre este nos va a permitir hacer que nuestro número está cayendo una y otra vez, como una danza infinita. Del mismo modo, usaremos el bloque [si <> entonces](#) esta nos permitirá evaluar si una afirmación es verdadera o falsa, también se encuentra en la sección control.

Primero debemos situarnos en nuestro objeto de los números, para hacerlo nos vamos a la parte donde están nuestros objetos y le damos click al de los números que añadimos en pasos anteriores.

En la zona de bloques agregamos el bloque cuando click en bandera verde, luego para fijarlo en una posición aleatoriamente, (similar a como hicimos con nuestro personaje) ahora usaremos los bloques [ve a posición aleatoria](#) y [fija y a <y posición>](#) para este caso el valor de y posición será de **130**, así nuestro número comenzará desde arriba. Los bloques y la vista luego de darle a la banderita verde deben quedarnos similar a la siguiente forma:



A lo anterior le agregamos un bloque para siempre. Ese bloque representa un loop o ciclo que se repite infinitamente. Dentro de este ciclo agregamos [cambia y <10> unidades](#); el valor de este bloque lo podemos cambiar a nuestro gusto, el número que se le coloque se le sumará al valor actual de y (en este caso sería 130) si es positivo, y si es negativo se le restará.

Cómo queremos que el objeto caiga debemos restar la posición y (es decir, el número usado debe ser negativo), por esta razón sugerimos los valores de **-10** para el bloque cambia y. Luego, le agregamos un segundo bloque al loop de la sección de control, llamado espera, y le agregamos un valor de **0.006** segundos para el bloque. La función de este es detener el desplazamiento por unidades en los segundos indicados. Deben verse así (imagen de la derecha):



Para observar cómo funciona le damos click a la banderita verde.

Verificado que el objeto cae de forma correcta, ahora haremos que cada vez que llega abajo, vuelva a aparecer en una posición aleatoria arriba. Para esto vamos a usar un nuevo bloque de la sección de control llamado condicional o bloque **si <condición>**; la condición que usaremos en este caso está en la sección sensores y se llama **¿tocando <puntero del ratón>?** este nos regresará una respuesta, si o no, esa respuesta es la que vamos a evaluar en nuestro condicional:



Entre las opciones la que nos sirve es **borde**, así sabremos que llegó al borde de abajo. Luego, ¿Qué vamos a hacer si llego? la respuesta ya la tenemos, debemos hacer que el objeto aparezca en una nueva posición aleatoria en la parte de arriba, para esto usamos los mismos bloques de antes **ve a <posición aleatoria>** y **fija y a <130>**. Justo debajo de este, añadimos el bloque siguiente disfraz de la sección de apariencia. El resultado debe verse así:



Todo este bloque lo ponemos debajo del bloque espera, quedando todo de la siguiente forma:



Ejecutamos y vemos nuestro juego y ahora deberíamos ver nuestro objeto cayendo y volviendo a aparecer arriba una y otra vez. Observemos que podemos seguir moviendo nuestro panda ya que es un objeto distinto y que ya hemos programado.

- 8 **Atrapar los números con nuestro personaje:** Avanzado con nuestro juego, vemos que ya tenemos dos objetos, el personaje y el objeto de los números. Para hacer que el personaje parezca atrapar los números debemos usar otro bloque condicional, llamado si con la condición `¿tocando <panda>?` en caso de que el objeto del personaje tenga otro nombre debemos usar ese, debe aparecer entre las opciones. Lo que va dentro de esta condición debe ser exactamente lo mismo que va dentro del bloque `si ¿tocando borde?` anterior. Todo debe verse ahora:



Probamos si funciona con el boton verde de la banderita y movemos nuestro personaje intentando atrapar los números que caen.

Hasta acá tenemos la primera parte de nuestro juego hecho, en el siguiente taller aprenderemos cómo hacer para que nuestro juego nos diga qué número debemos atrapar y podamos saber si el número que atrapamos es el número correcto.

### 9 Poniendo un toque de diversión:

Para este paso regresamos al objeto del Panda. Por si no lo recuerdas, este es el código que le habíamos puesto:



Para darle un toque más divertido al juego, vamos a añadirle diálogos al personaje. Para ello, dirígete a la sección de apariencia y escoge el bloque [di ¡Hola! durante 2 segundos](#).



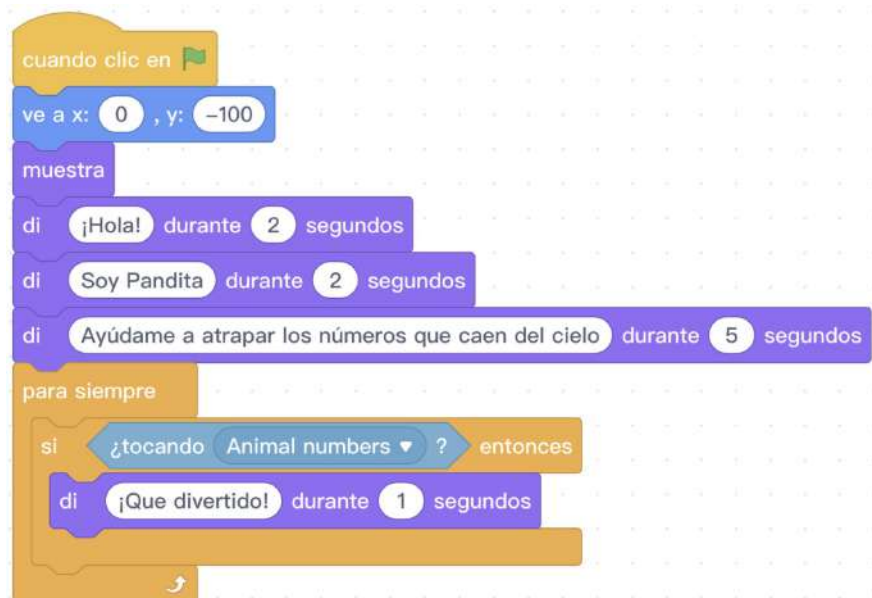
Colócalo debajo del bloque de muestra. Luego agrega otro bloque igual y cambia el mensaje de [¡Hola! a Soy Pandita](#) y ponle unos 2 segundos para que el jugador pueda leer el contenido. Por último, agrega un tercer bloque y escribe las instrucciones del juego. Esta vez puedes colocarle 5 segundos ya que es un mensaje más largo. El código se vería así:



Ahora para hacer que Pandita hable cada vez que toque un número, ve a la sección de control y escoge el bloque **si \_\_ entonces**, dentro del cual colocarás el bloque de la sección de sensor **¿tocando \_\_ ?** y reemplazas el contenido puntero de ratón a el objeto de los números de animales. Después, dentro del bloque **si \_\_ entonces** añade un bloque de apariencia **di \_\_ durante \_\_ segundos** y coloca el mensaje que quieres que Pandita diga. Todo este bloque lo deberás colocar dentro del bloque para siempre para que se repita cada vez que toque los números. El código se verá así:



Por último todos estos bloques se colocan debajo del bloque original de los movimientos del panda:



**¡Excelente trabajo, has completado  
el primer taller!  
¡Nos vemos en el próximo!**





# mBlock: Taller #2

## Juego de los Números Correctos - Parte 2

**Objetivo:** Implementar conceptos de programación como uso de variables, condicionales anidados y ciclos repetitivos, a través de complementar el desarrollo del juego base hecho en el primer taller.

Plataforma de trabajo: [mblock https://ide.mblock.cc/#/](https://ide.mblock.cc/#/)

Demo del juego final esperado: [Demo: Primer Juego](#)

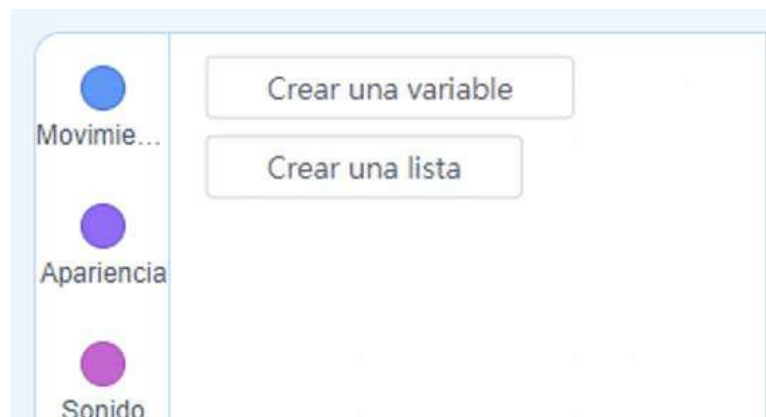
### Desarrollo

- 1 **Retomando nuestro proyecto:** En este apartado vamos a continuar trabajando con el primer taller. Para abrirlo nos vamos a mBlock y le damos Archivo -> Abrir escogemos nuestro proyecto y aceptar.



- 2 **Creando variables:** Las variables en programación son un espacio donde podemos guardar información. Por ejemplo, para poder guardar un puntaje o si queremos guardar el nombre de un jugador necesitamos una variable.

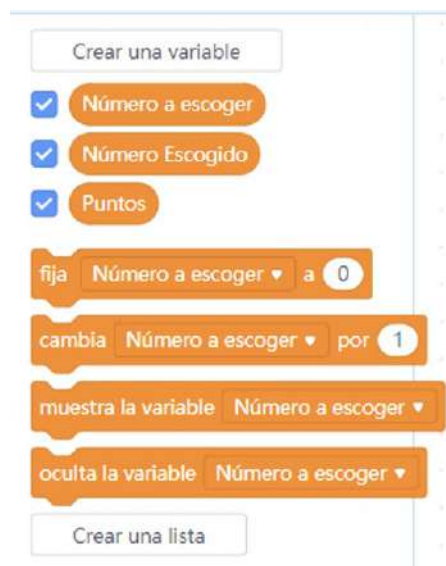
Para crear variables nos vamos a la sección variables y escogemos Crear una variable:



Nos pedirá que nombre queremos ponerle a nuestra variable (todas las variables tienen un nombre que las identifica). Vamos a colocarle Puntos para que nos ayude a guardar la puntuación obtenida durante la partida y seguido damos en aceptar. En la próxima nuestra primera variable creada:



Siguiendo estos mismos pasos, vamos a crear otras dos variables. La primera se llamará Número a escoger y la segunda Número Escogido. Al final nuestra sección de variables debe verse así:

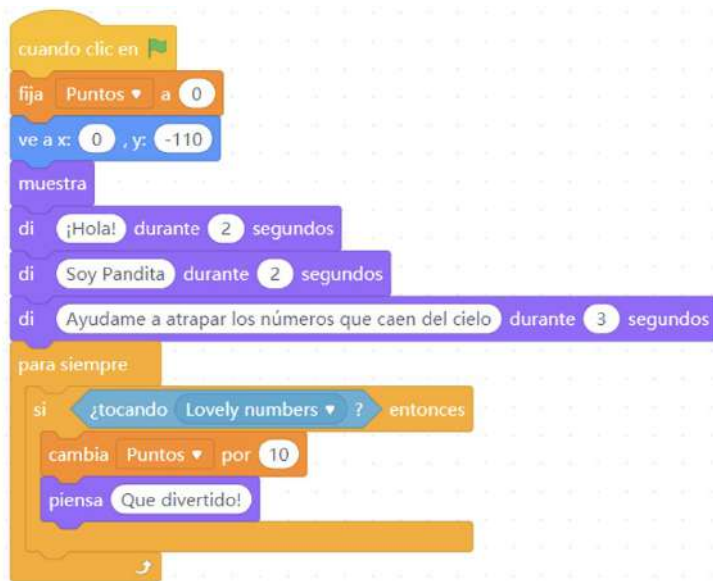


**3 Aumentado el puntaje:** Como en cualquier juego, cada vez que el personaje atrape un número el puntaje debe aumentar. Para hacer esto vamos a utilizar el bloque que está en la sección de variables llamado **fija <variable>a <0>**. Dejamos el valor como cero y lo agregamos al inicio de todos los bloques, es decir, debajo del bloque cuando clic en banderita verde:



Lo anterior nos ayuda a darle un valor inicial a nuestra variable. Este concepto se conoce como “[inicialización de variable](#)”.

Ahora, usaremos otro bloque, de la misma sección, llamado [cambia <puntos> por <1>](#). Escogemos la variable puntos y el valor, que por defecto es 1, lo cambiamos a 10. Este valor es el número que vamos a sumarle a nuestro puntaje cada vez que nuestro personaje atrape un número. Por esta razón debemos ponerlo dentro de la condición que definimos en el taller anterior llamada [si <¿tocando <objeto número>?> entonces](#). Hasta ahora, nuestro código de bloques del objeto personaje debe verse similar a la siguiente forma:



**Nota:** Es importante que dentro del bloque condicional el bloque cambia puntos por esté de primero.

Una vez terminado el proceso anterior, ejecutemos y observamos nuestro juego, utilizando puntos para que sea más divertido.

- 4 [Generando el número a escoger](#): En este punto haremos que nuestro juego nos dé un número que es el que vamos a tener que elegir con nuestro personaje, para esto debemos hacer que nuestros bloques de código lo generen y nos diga qué número es el que debemos elegir. Para esto vamos a hacer uso de nuestra variable llamada Número a escoger, la cual creamos en pasos anteriores. El siguiente trabajo se hará en la zona de bloques de nuestro objeto de números.

Primero al iniciar el juego debemos decirle que Número a escoger tenga el valor de cero. Para esto usamos el bloque de la sesión de variables llamado [fija <Número a escoger> a <0>](#). Recordemos que si este dice el nombre de otra variable de esta debe cambiar por Número a escoger.



Luego, se debe poner este bloque después del evento darle a la banderita verde de nuestro objeto de números. Por lo cual el inicio de esa sección de bloques debe verse así:

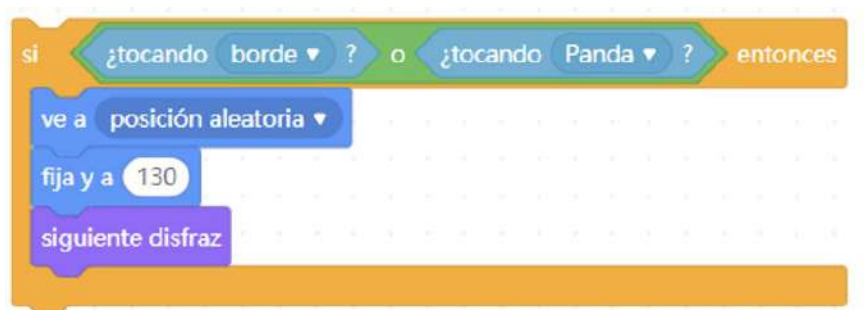


Ahora, se debe modificar el número a escoger usando un bloque llamado número al azar en este se pone un rango que vaya desde el número 0 al número 9. Luego, ese bloque que se puede llamar sentencia, se incluye dentro del bloque fija Número a escoger quedando de la siguiente forma:

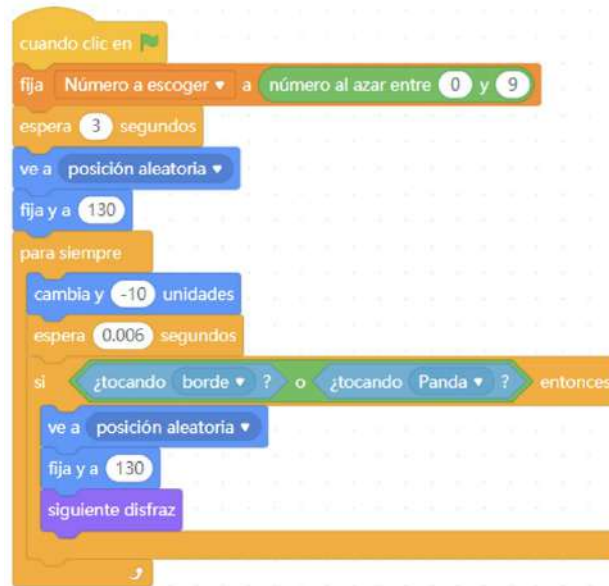


**5 Cambiando los números que caen de manera aleatoria:** Hasta el momento de nuestro desarrollo tenemos que los números que van cayendo lo hacen de forma ascendente, es decir, que primero cae el cero luego el uno, sigue el dos así sucesivamente, cuando llega al nueve vuelve y cae nuevamente el cero. Esto lo determina el bloque siguiente disfraz. Lo que queremos ahora es que mediante la variable número a escoger se determine cuál es disfraz que tomará nuestro objeto número. Esto para que nuestro juego tenga más sentido y lo entenderemos mejor en el siguiente paso. Por ahora, haremos lo siguiente.

Primero usaremos una sentencia que me permita unir los bloques que verifican si el número está tocando al panda y el condicional que determine si está tocando. Para esto nos vamos a la sección operadores y tomamos el bloque de sentencia o. Luego, esa sentencia la ponemos como condición del bloque si y dentro de este los bloques que comparten ambos condicionales. Todo junto debe verse así:

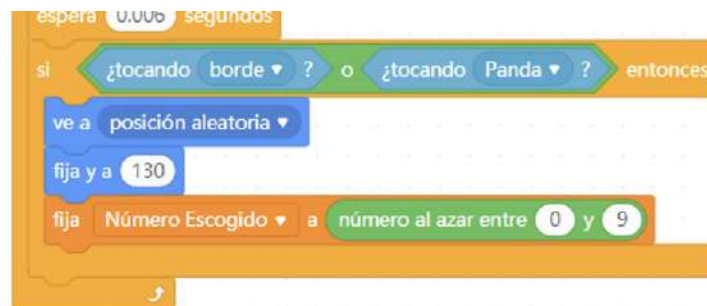


El bloque anterior lo agregamos a nuestro conjunto de bloque principal del objeto número el cual debe quedar de la siguiente manera:



Mucho mejor, ¿cierto? Lo anterior nos permitirá evitar repetir bloque de código y evaluar las dos situaciones al mismo tiempo. Es a lo que se le conoce como una operación lógica.

Seguidamente, quitaremos el bloque siguiente disfraz, pues es el que vamos a reemplazar. Luego, ya que nuestra variable número escogido determina el disfraz de nuestro objeto número, utilizaremos al igual que con la variable número escogido el bloque **fija <número escogido a <número al azar entre 0 y 9>**. Donde estaba antes el bloque siguiente disfraz ponemos ese bloque mencionado.

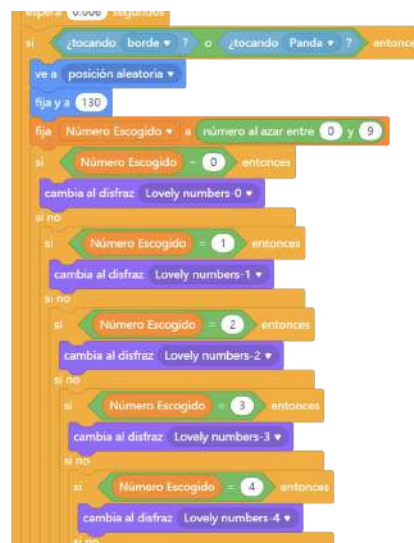


En este punto haremos algo interesante, usaremos un concepto llamado condicionales anidados. ¿Cómo lo haremos?, usando el bloque llamado **si <condición> sino**. Se llama anidado porque usaremos unos dentro de otros. Esto nos ayudará a decirle al objeto número que, si la variable número escogido es igual a cero el va a ponerse el disfraz uno, sino es así, se verificará después si la variable número escogido es igual a uno si es así se pondrá el disfraz de uno en caso de que no sea así, sigue a comprobar si la el numero es cogido es dos y así sucesivamente comparar la variable número escogido con todos los números y al encontrar el indicado el sabrá cual disfraz ponerse.

De esta forma sabremos qué número es el que compararemos más adelante. El código se debe ver de la siguiente manera:



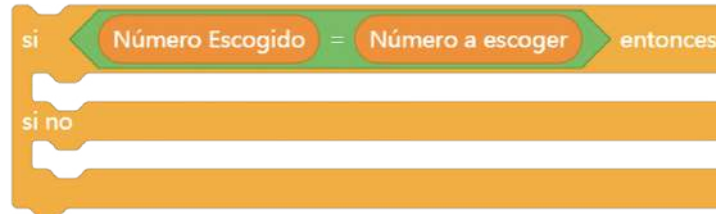
Todo ese conjunto de bloques unidos, o condicionales anidados como hemos dicho. Debe ir justo después del bloque fija <número escogido a <número al azar entre 0 y 9> dentro de nuestro condicional que verifica si el número tocó al panda o al borde inferior quedando justo después del bloque fija que pusimos en el paso anterior.





6 Comparar si el número escogido es el correcto: En esta parte nos vamos a dirigir a nuestro objeto personaje, y trabajaremos en la zona de bloques de este.

Primeramente, para determinar si el número escogido es el correcto debemos ir al contenido del bloque que dice ¿tocando a mi objeto número? y debemos poner allí, justo al inicio un bloque si <condición> sino, que contenga la sentencia que diga: ¿Número a escoger es igual a número escogido? Este bloque debe verse de la siguiente forma:



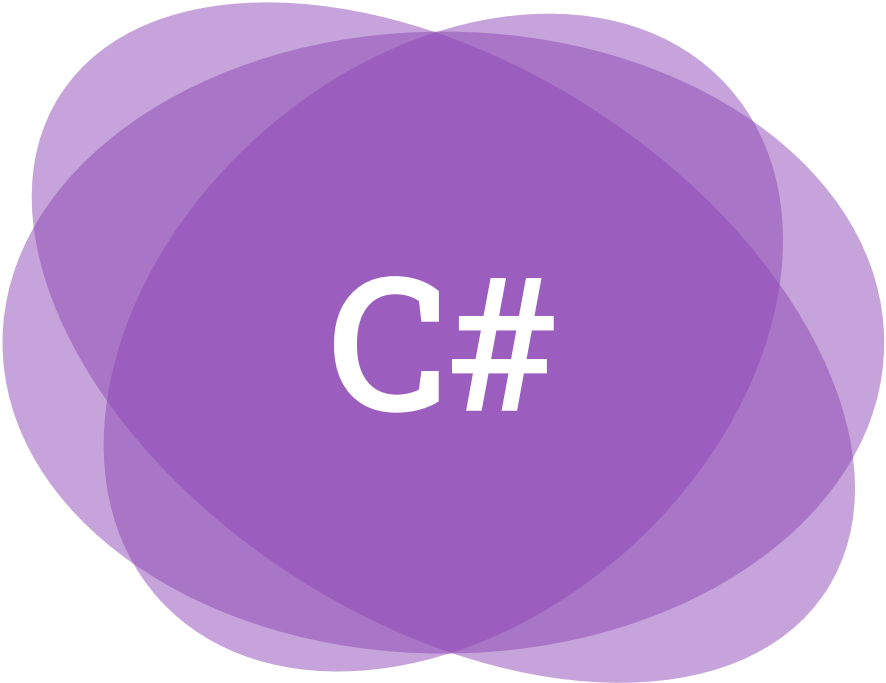
Ahora, la ponemos dentro del bloque si ¿tocando a mi objeto número? y a su vez dentro de esta. Al cumplirse la condición plateada confirmamos que el número que atrapó el panda es el correcto en este caso aumentamos la variable puntaje y enviamos el pobloque piensa que divertido!. En caso contrario, en la casilla del sino, se debe enviar un mensaje indicando que es el número incorrecto. Dentro del sino debemos penalizar el error cometido por lo cual debemos disminuir el puntaje para esto una consideración a tener en cuenta es que si el puntaje es cero, no debemos restar nada. Esto quiere decir que, si el puntaje es mayor que 0 cuando ocurra un error debemos descontarle puntos, ¿cómo? usando el bloque cambia puntos y ponemos el número en negativo de los puntos que le queremos, por ejemplo vamos a usar menos cinco (-5).

Lo anteriormente explicado debe verse de la siguiente manera:



Con esto tenemos nuestro primer juego con bloques completado, ¡Bienvenidos al mundo de la programación y creación de videojuegos!





C#

The image features a central graphic of three overlapping circles in various shades of purple. The text 'C#' is centered within the darkest, most prominent circle. The background is a light gray gradient, and the composition is framed by abstract geometric elements in purple and gray at the corners and edges, including lines, squares, and a large 'X' shape in the top right.

# C#: Primeros Pasos

**Nota:** Omitir el primer y segundo paso si no quieres instalar el complemento mLink para ver el código python.

Objetivo: Aprender los fundamentos necesarios para escribir y leer código de programación con el lenguaje C.

Plataforma de trabajo: [Repl.it](https://repl.it)

## De bloques a código:

### 1 Inicio de nuestro programa:

Para iniciar nuestros programas en mblock usábamos el bloque cuando clic en banderita verde.

cuando clic en 

En lenguaje C este bloque se representa de la siguiente manera:

```
1  #include <stdio.h>
2
3  int main(void) {
4      |
5      return 0;
6  }
```

### 2 Mostrar mensajes por pantalla:

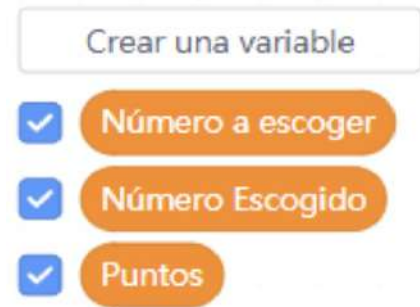
Con bloques cuando queríamos mostrar un mensaje por pantalla usamos, los bloques: di <mensaje> durante <x> segundos y di <mensaje> durante <x> segundos. Esto en código C lo usamos por medio de la línea de código:

```
printf("Mensaje");
```

### 3 Variables:

Las variables como el puntaje, en bloques las creábamos y usábamos de la siguiente manera:

Crear una variable en C es también fácil, la diferencia es que debemos especificar que guardará un número o texto. Para el caso de las variables de número se usa la palabra `int` seguido pones el nombre de la variable, que queramos. Las variables que creamos en mBlock se crearía en C así:



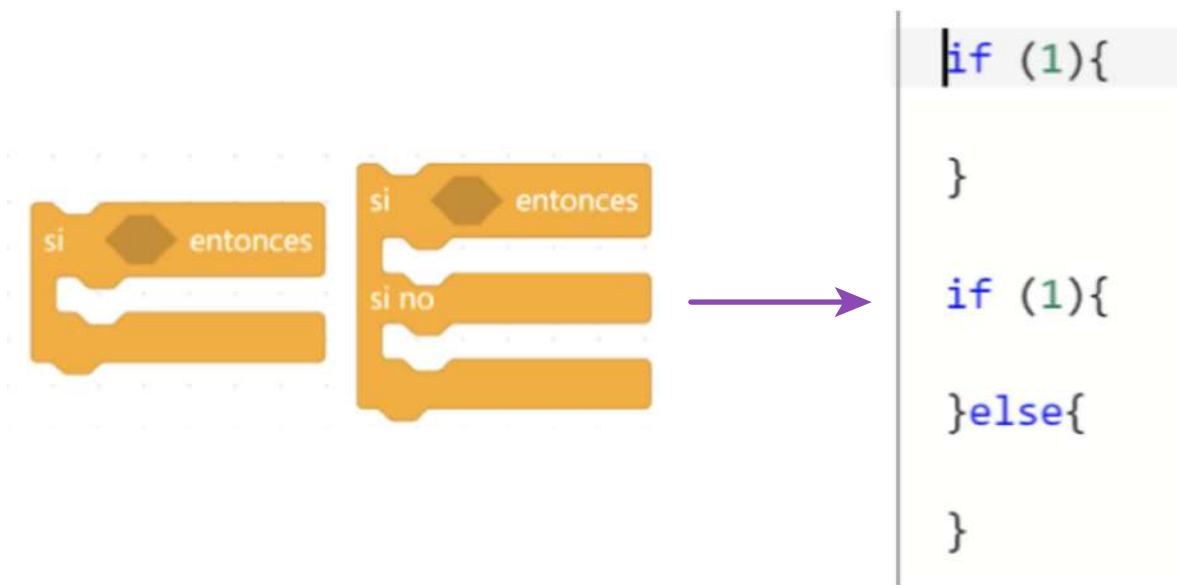
```
int puntos;  
int Numero_a_escoger;  
int Numero_escogido;
```

Para darle valores a estas variables que creamos usamos un signo igual, en bloques recordemos que usamos el bloque fija <variable> a , por ejemplo si queremos que los puntos sean cero, el número a escoger sea 3 y el número escogido sea 3, lo ponemos de la siguiente manera:

```
puntos = 0;  
Numero_a_escoger = 3;  
Numero_escogido = 3;
```

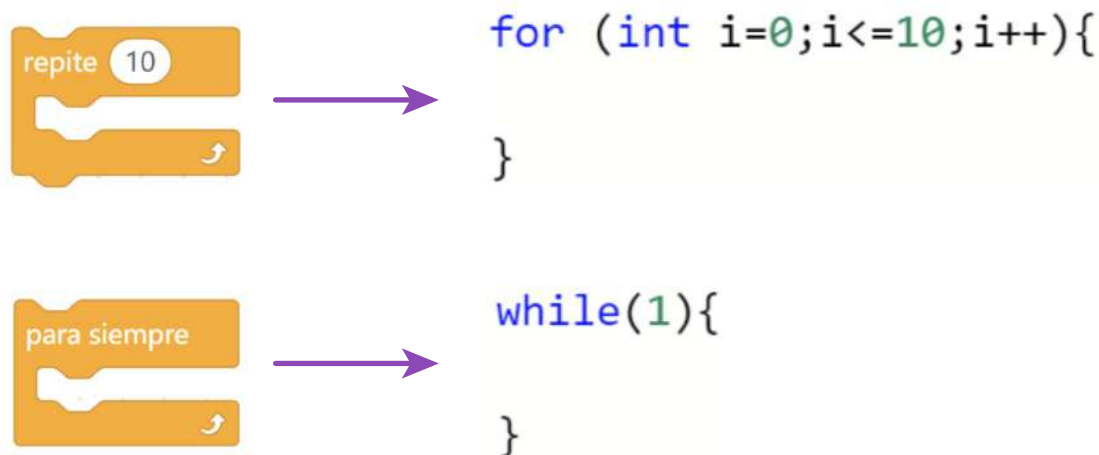
### 4 Condicional si-sino:

Estas sentencias vamos a verlo directamente como se ve en bloques y como luce en código C:



## 5 Ciclos:

A continuación vemos como se ve la transición de los ciclos de bloques a código C:



Para el ciclo repite hasta en c se implementa de forma distinta, usamos un while con una condición. Mientras que se cumpla esta condición se va seguir repitiendo el contenido que se pone dentro hasta que se deje de cumplir. En mBlock se ejecutaba hasta que la condición se cumpliera, allí radica la diferencia que puede haber entre el bloque y el código. Pero, el principio será siempre el mismo.



## Preparando el entorno de trabajo con Repl.it:

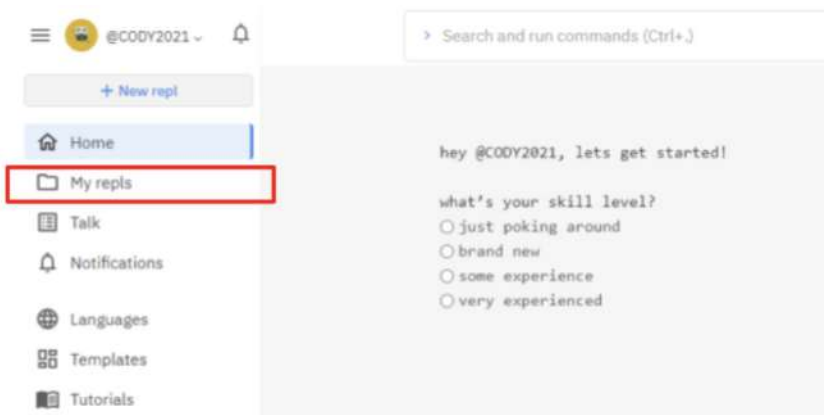
- 1 Accedemos a la plataforma donde trabajaremos, la página inicial es la siguiente. En ella le damos clic donde dice start coding.



Aparecerá la siguiente ventana para registrarnos. Deberás ponerle un nombre de usuario, un correo al que tengamos acceso y una contraseña.

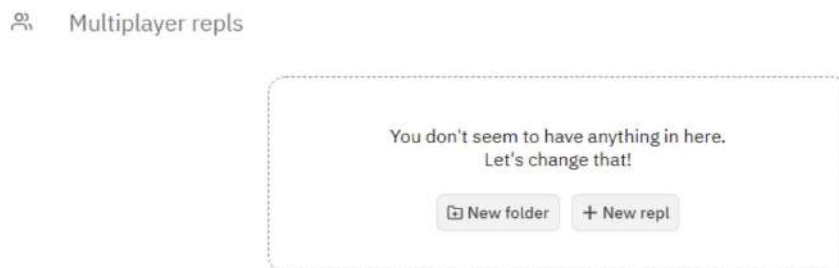
Le damos clic en Sign up y luego vamos a nuestro correo electrónico y verificamos la cuenta. Ahora en la página principal de la plataforma le damos en Log In y ponemos nuestro usuario y contraseña.

En la ventana inicial abrimos el menú lateral izquierdo y nos vamos a My repls, allí haremos nuestros códigos de programación.

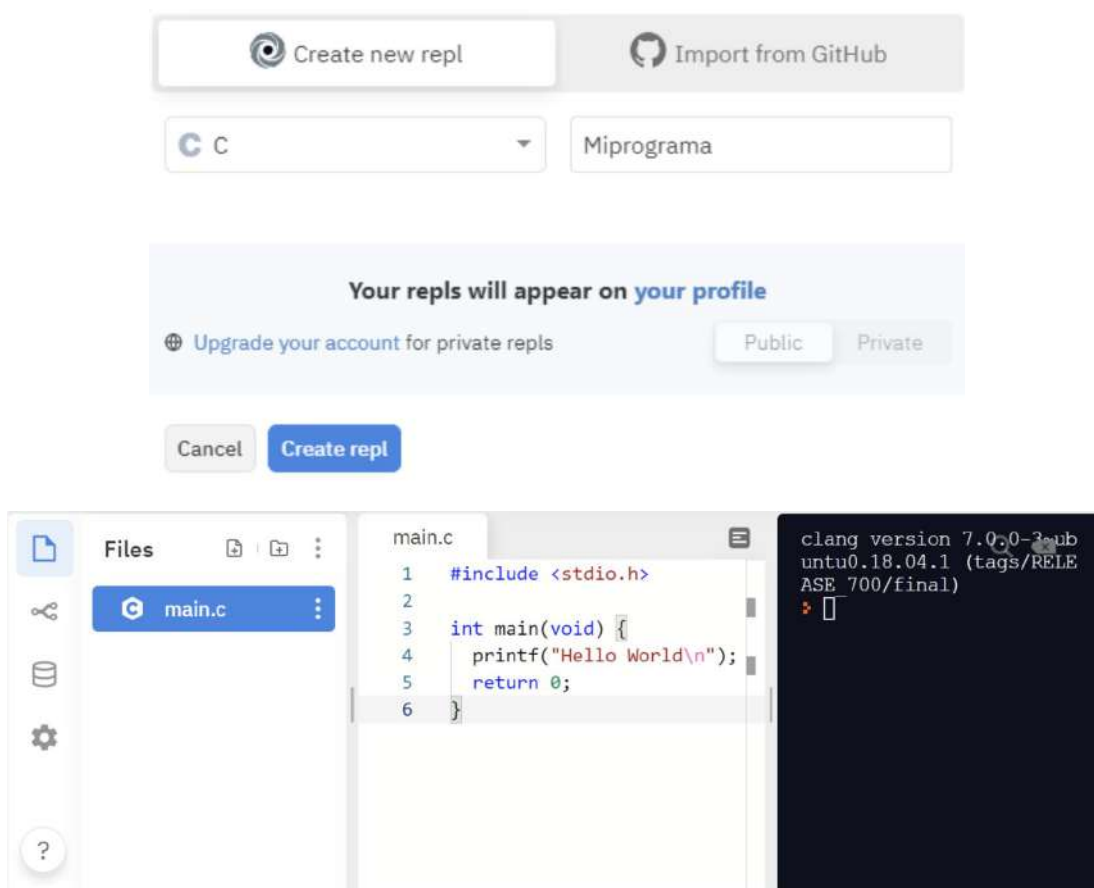




Nos mostrará esta ventana, en donde le damos clic a new Repl:



Luego, escogemos el lenguaje de programación C de la lista que nos aparece y ponemos el nombre de nuestro programa, en este caso se llamará Mi programa. Finalmente, damos clic en create repl y nos llevará a nuestro entorno de trabajo.



**¡Una vez haya leído este documento  
estás listo para comenzar con tu  
primer taller de C#!**



The image features a minimalist, abstract design. In the center, the word "Arduino" is written in a white, sans-serif font. It is positioned within a series of three overlapping, rounded rectangular shapes in shades of blue. The background is a light grey. Decorative elements include several blue lines: a vertical line on the left, a horizontal line at the bottom, and a vertical line on the right. These lines are accompanied by various grey geometric shapes, including a plus sign in the top left, a small square below it, a circle to the right of the plus sign, a small square on the right side, a larger square at the bottom right, and a small circle at the bottom left. The overall aesthetic is clean and modern, typical of a technical or educational presentation.

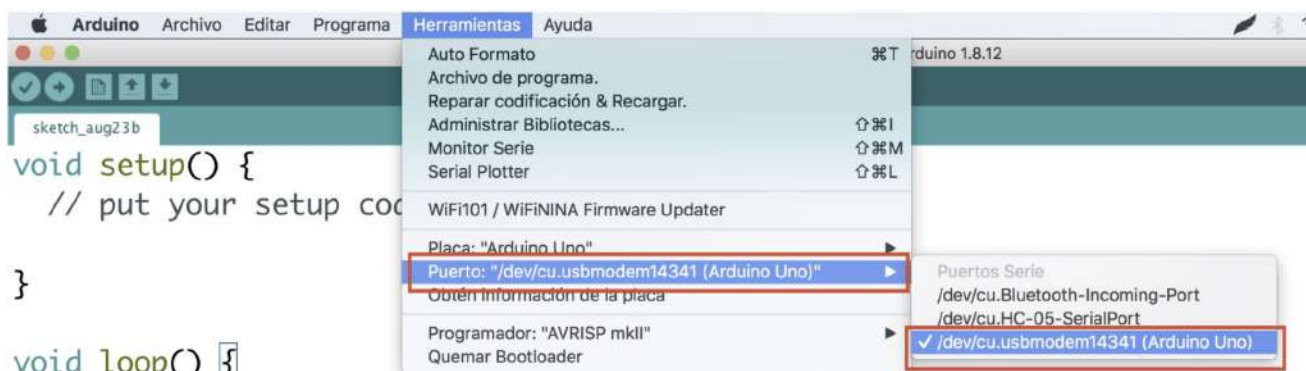
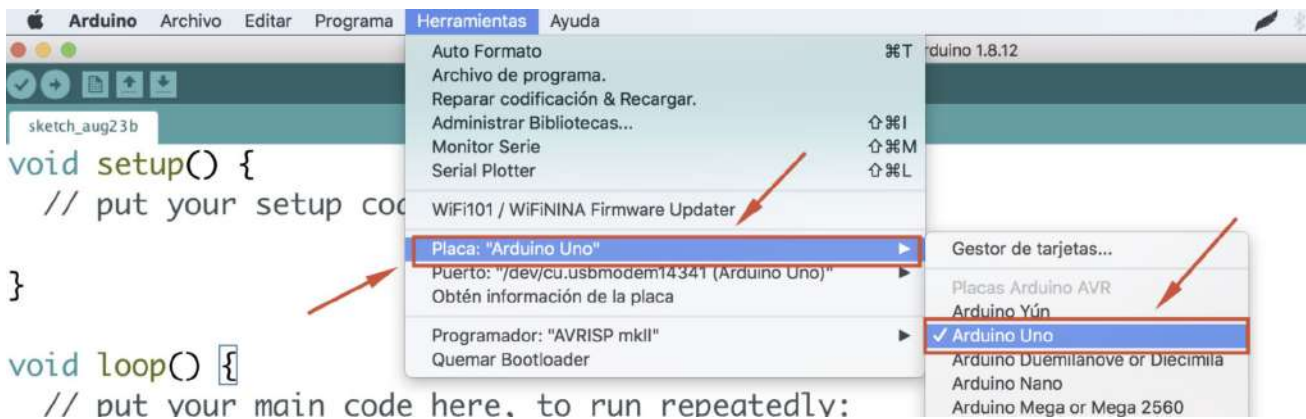
# Arduino

# Configuración y Monitor Serie

## Paso 1

Una vez ensamblado el prototipo, conecte la placa a su computadora y abra el programa de arduino. Asegúrese que tenga la siguiente configuración:

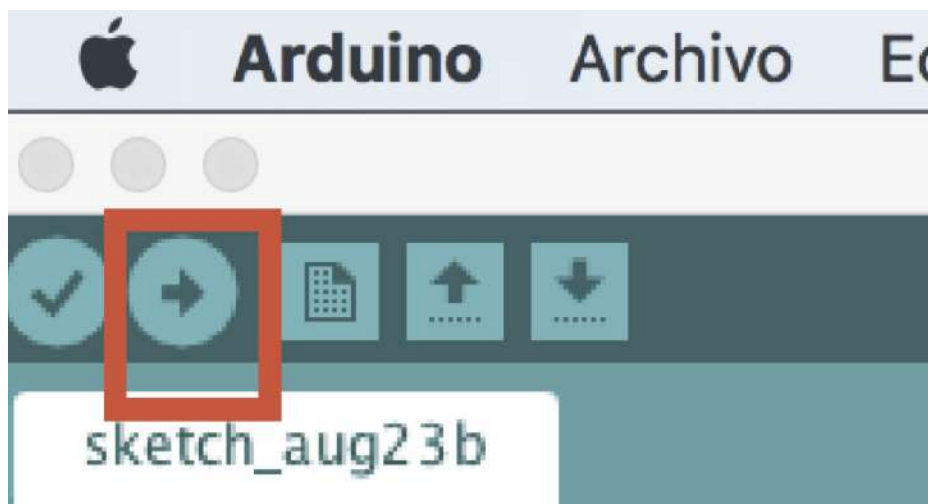
❖❖ NOTA: El puerto depende del computador.



⚠ En sistemas con sistema operativo Windows, el nombre de los puertos comienza con **COMX**, siendo X un número, deberá probar con todos los puertos para asegurarse cual de estos es el que está conectado a el Arduino.

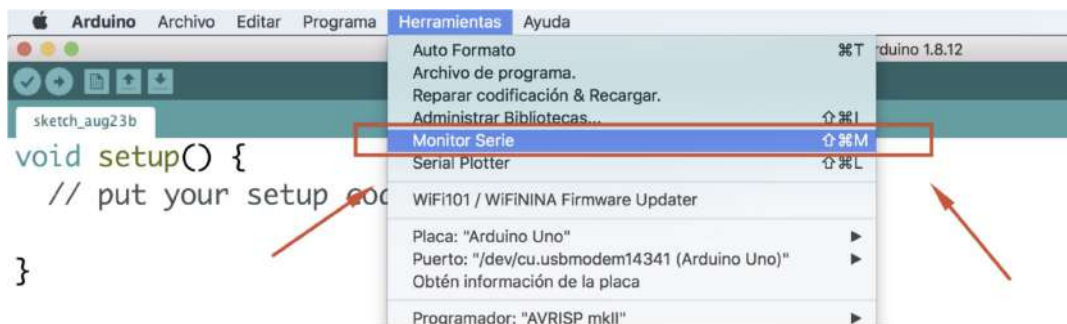
## Paso 2

Suba el código a la placa, oprimiendo el siguiente botón:



## Paso 3

Abra el monitor serie para observar las señales que le envía el arduino.

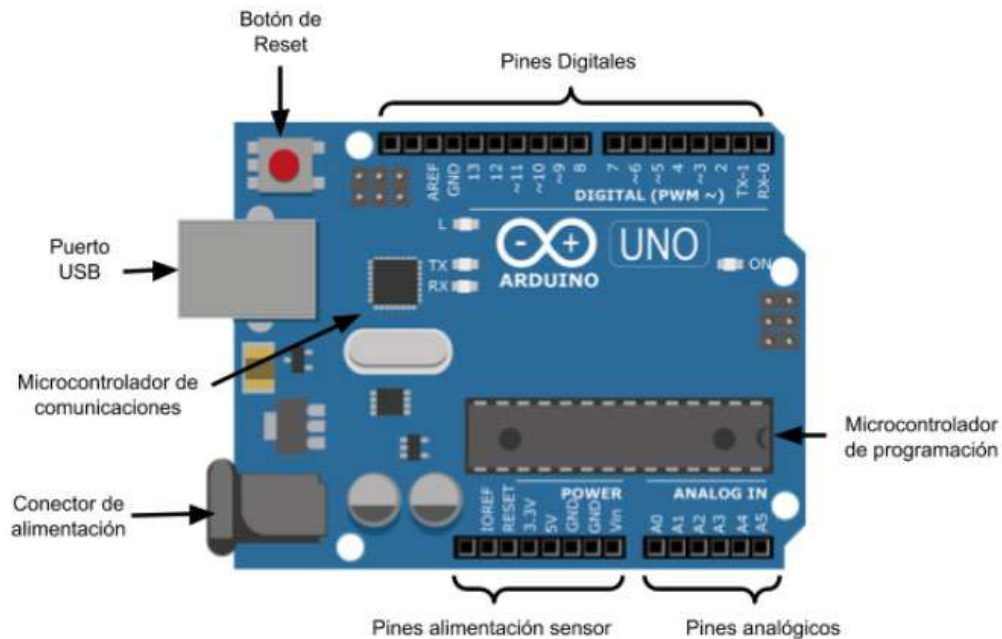


# Guía Básica de Arduino

## ¿Qué es Arduino?

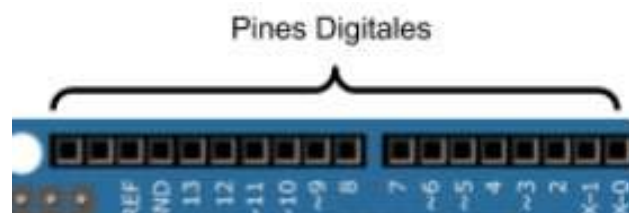
Arduino es una plataforma de desarrollo que parte del uso de una placa electrónica que tiene un microcontrolador, el cual se puede reprogramar. Tiene una serie de pines que permiten la comunicación entre los diferentes componentes de la placa y el microcontrolador.





## Pines Digitales:

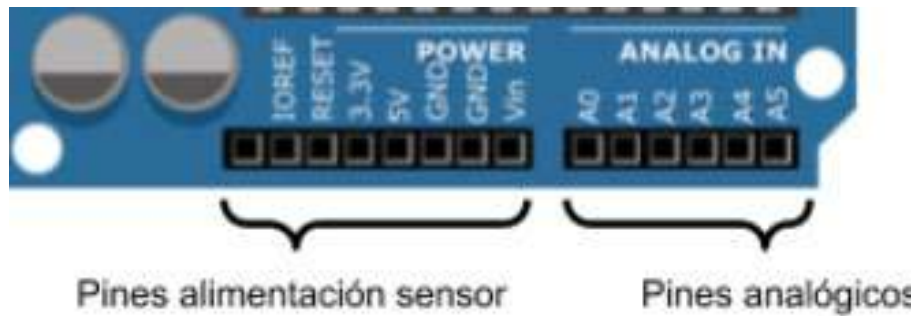
Son las conexiones digitales de los dispositivos conectados en la placa. La placa de Arduino cuenta con 14 pines digitales, que van del 0 al 13. Una señal digital solo puede tener dos estados: 0 (LOW, bajo, false): Indica 0V de tensión enviados desde la placa. 1 (HIGH, alto, true): Indica 5V de tensión enviados desde la placa. Por lo tanto, cuando ponemos un pin digital a valor HIGH, la placa suministra 5V de tensión por la salida que hayamos indicado, y si ponemos el valor a LOW suministrará 0V de tensión. (Ojo: Hay que tener en cuenta que los 5V no siempre son 5 ni los 0 siempre son 0) Los pines digitales de Arduino pueden ser usados tanto de entrada como de salida.





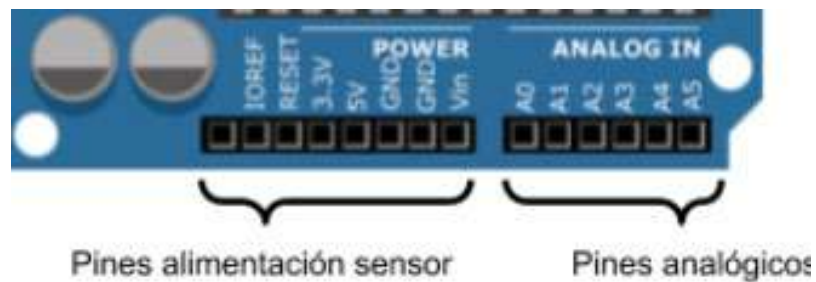
## Pines Alimentación Sensores:

Además de los pines de entrada y salida descritos anteriormente, Arduino dispone de pines que nos permiten alimentar componentes externos, concretamente uno con 5V y otro con 3,3V. También dispone de pines de tierra (GND).



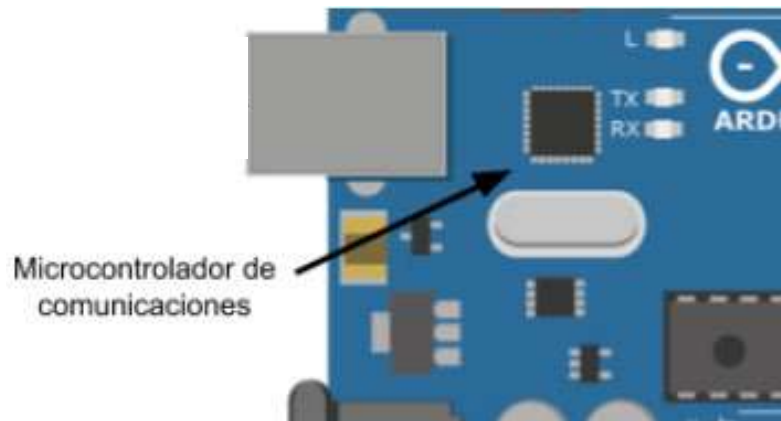
## Pines analógicos:

Los pines analógicos pueden medir un rango de valores de voltaje, a diferencia de los digitales que solo entienden dos valores: 0-1, o lo que es lo mismo, 0V o 5V. Con los pines analógicos vamos a poder leer valores intermedios entre 0V y 5V, representados con un valor entero comprendido entre 0 y 1023, ya que la información se representa en números de 10 bits, y también vamos a poder escribir en los pines valores comprendidos entre 0 y 255, ya que la información se representa en números de 8 bits. En el punto anterior hemos hablado sobre pines digitales, si te fijas en ellos verás que aparecen algunos con el símbolo “~” en la placa, este símbolo indica que pueden ser utilizados también como pines analógicos.



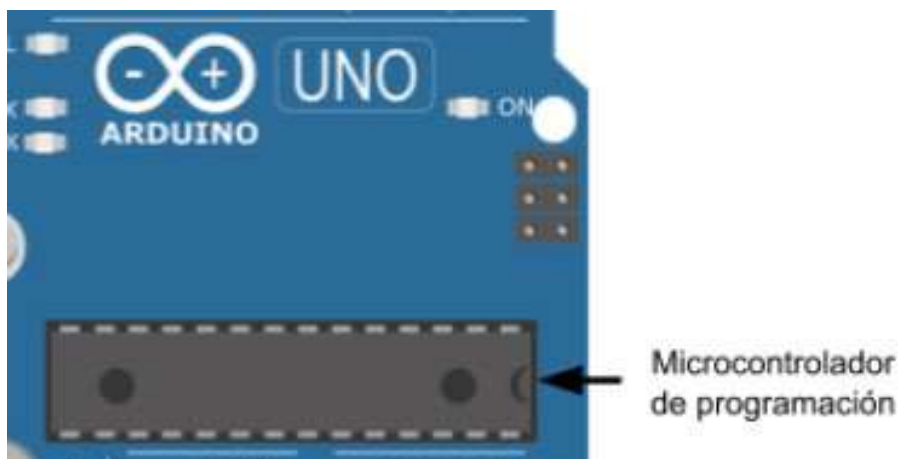
## Microcontrolador De Comunicaciones:

El microcontrolador de comunicaciones se encarga de gestionar las comunicaciones con todo lo que se conecta a la placa.



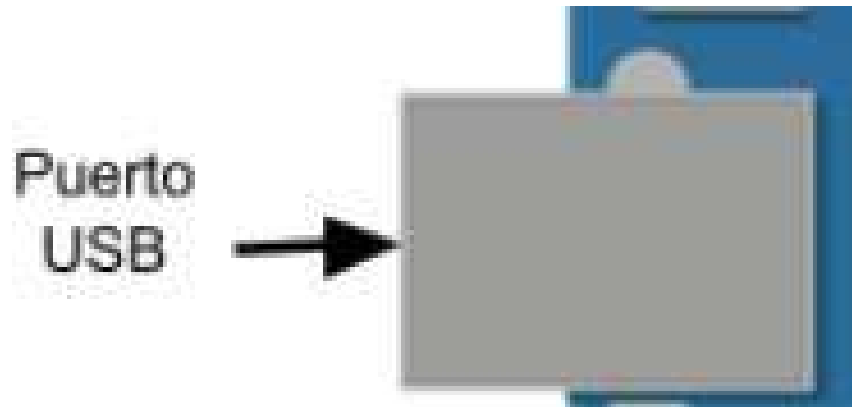
## Microcontrolador De Programación:

Este componente de la placa es el cerebro de la misma, es donde la placa almacena el programa que tiene que ejecutar y el que lo ejecuta. El microcontrolador de la placa se programa utilizando el IDE (Entorno de Desarrollo Integrado) de programación gratuito de Arduino. En los siguientes apartados explicamos cómo instalarlo y cómo ponerlo a funcionar.



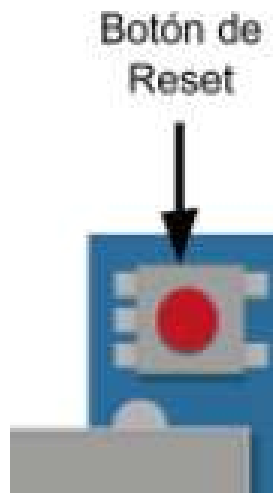
## Puerto Usb:

Es el puerto mediante el cual nos comunicaremos con la placa de Arduino. Sus funciones principales son: Alimentación y Cargar los programas en el microcontrolador. Envío de información desde la placa al ordenador.



## Botón Reset:

Permite reiniciar el programa que se ha cargado en el microcontrolador interrumpiendo la ejecución actual. Ten en cuenta que no borra el programa que se ha cargado, únicamente lo reinicia.



## Instalación del entorno de programación

Para desarrollar los programas en Arduino existen dos opciones. La primera es utilizar una página web, que cuenta con todo el entorno de desarrollo necesario y la otra manera es descargando un programa del entorno de desarrollo. Aquí en estos entornos es donde se pueden escribir y guardar los códigos de los programas que se desarrollan. La guía paso a paso de la instalación del entorno y cómo dar los primeros pasos en la plataforma web están disponibles en:

Página Web del editor Online: <https://create.arduino.cc/editor>

Para instalar en el computador disponible en:

<https://www.arduino.cc/en/Main/Software>



## Descripción de la estructura de un programa

La estructura de un programa en arduino tiene dos métodos principales donde se pone el código, estos son `setup()` y `loop()`:

En `setup()` es donde se codifican todas las sentencias de configuración, inicialización de los puertos de la placa, configuración de las comunicaciones y otros parámetros.

En `loop()` es el centro del programa donde se ejecuta de forma continua y repetitiva el código arduino.

```
4
5 void setup() {
6   |
7   }
8
9 void loop() {
10  |
11  }
12
```

## Links de fuentes de información

1. <https://arduino.cl/que-es-arduino/>
2. Libro: Aprendiendo Arduino en un fin de semana

# Arduino - Taller #1



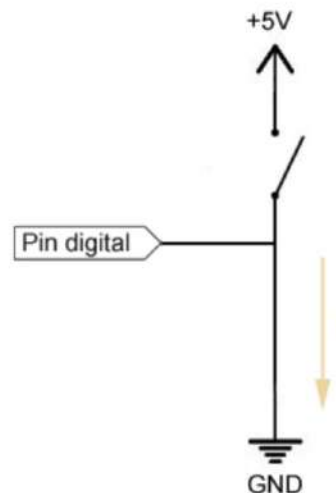
## Introducción

El objetivo principal de un pulsador es que al ser presionado, permite el paso de la corriente. Sin embargo, la lectura del estado de este dependerá del tipo de resistencia que empleamos: La Pull Down o Pull Up .

### Pull-Up

Cuando el pulsador no es presionado, su estado será HIGH, en cambio cuando ya es presionado mostrará un estado LOW.

Al analizar la imagen podemos observar que gracias a que el interruptor se encuentra abierto, ya que no se ha presionado el botón, podemos leer el el estado HIGH. Esto es gracias a que el pin se encuentra conectado directamente a los 5V.

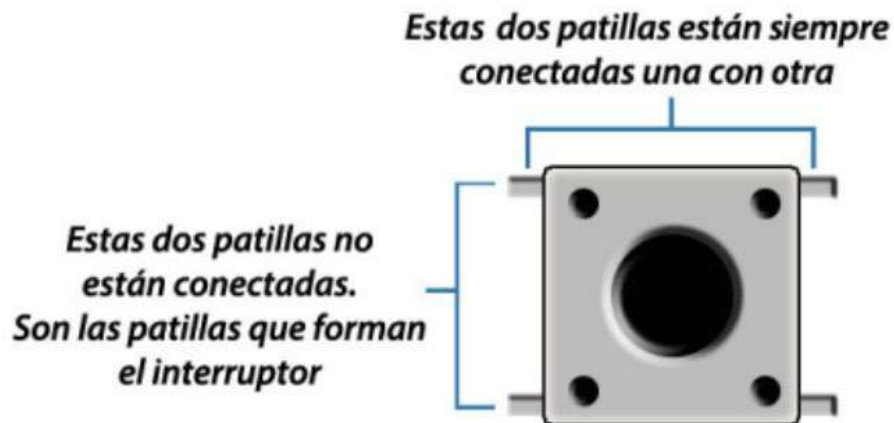
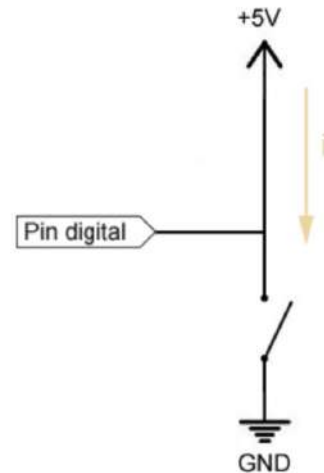




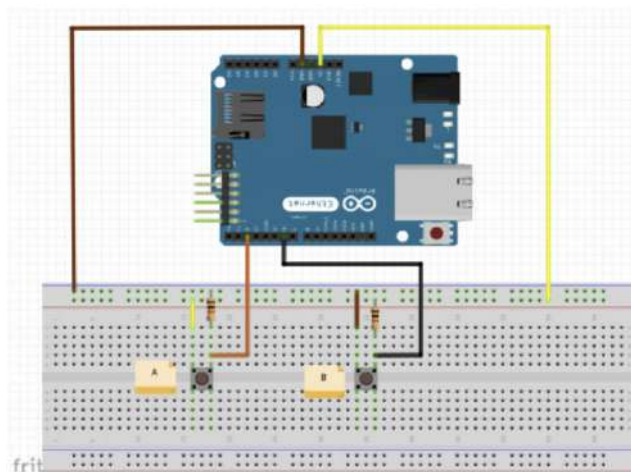
## Pull-Down

Cuando el pulsador no es presionado, su estado será LOW, en cambio cuando ya es presionado mostrará un estado HIGH.

Es parecido al caso que se vio en las resistencias Pull-Up, la diferencia es que el pin se encuentra conectado directamente a GND, por lo tanto marca un estado bajo cuando el circuito está abierto.



En este montaje vamos a tener dos pulsadores, uno con resistencia Pull-Up y el otro con resistencia Pull-Down.



## Código

Suba el siguiente código a la placa.

```
// Constantes para definir los pines
int DOWN = 2;
int UP = 6;
int LED = 13;
// Auxiliares
int valDown= 0;
int valUp= 0;
char msm[50];

void setup() {
  Serial.begin(9600);
  // Activamos los pines de entrada
  pinMode(DOWN, INPUT);
  pinMode(UP, INPUT);
  // Activamos los pines de salida
  pinMode(LED, OUTPUT);
}
void loop() {
  // Leemos el valor de los pines
  valDown = digitalRead(DOWN);
  valUp= digitalRead(UP);
  // Creamos un String
  sprintf(msm, "{\"a\":%i,\"b\":%i}", valDown, valUp);
  // Imprimimos el string creado
  Serial.println(msm);
  //Añadimos un retraso
  delay(1000);
}
```



## Monitor serial

Abra el monitor serial y presione los botones para observar las distintas señales que se generan.



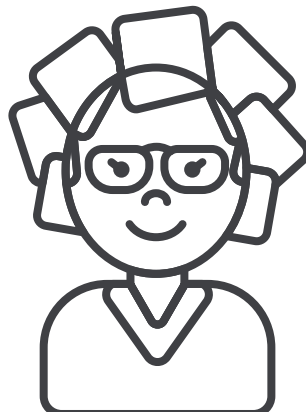
```

{"a":0,"b":1}
{"a":0,"b":1}
{"a":0,"b":1}
{"a":0,"b":1}
{"a":1,"b":1}
{"a":1,"b":1}
{"a":1,"b":0}
{"a":1,"b":0}
{"a":0,"b":0}
{"a":0,"b":0}
{"a":0,"b":1}
{"a":0,"b":1}
{"a":0,"b":1}

```

Autoscroll ☒ Mostrar marca temporal ☐ Sin ajuste de línea 9600 baudio Limpiar salida

**¡Has finalizado el Taller #1!  
Sigue con el excelente trabajo.**



# Arduino - Taller #2

## Introducción

El módulo joystick consta de cinco pines, donde VRx y VRy, que representan las posiciones (x,y) respectivamente, son pines analógicos.

Estos pines son analógicos dado que son valores continuos que puede tomar un sinnúmero de valores dentro de un límite.

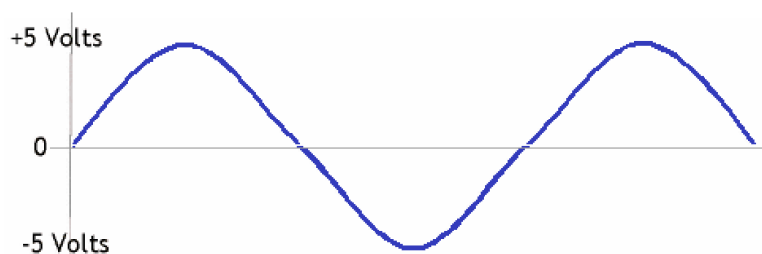
Podemos contrastar con los botones, cuyos pines son digitales ya que puede tomar valores fijos, que son 0 y 1.



Estos pines son analógicos dado que son valores continuos que puede tomar un sinnúmero de valores dentro de un límite.

Podemos contrastar con los botones, cuyos pines son digitales ya que puede tomar valores fijos, que son 0 y 1.

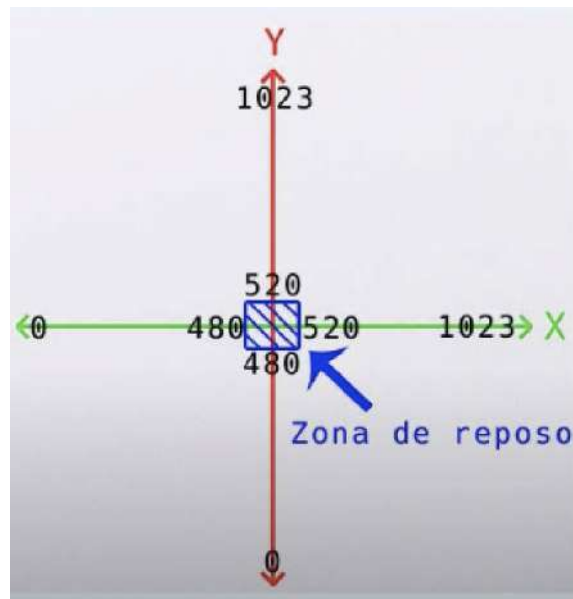
### Un evento analógico



### Un evento digital

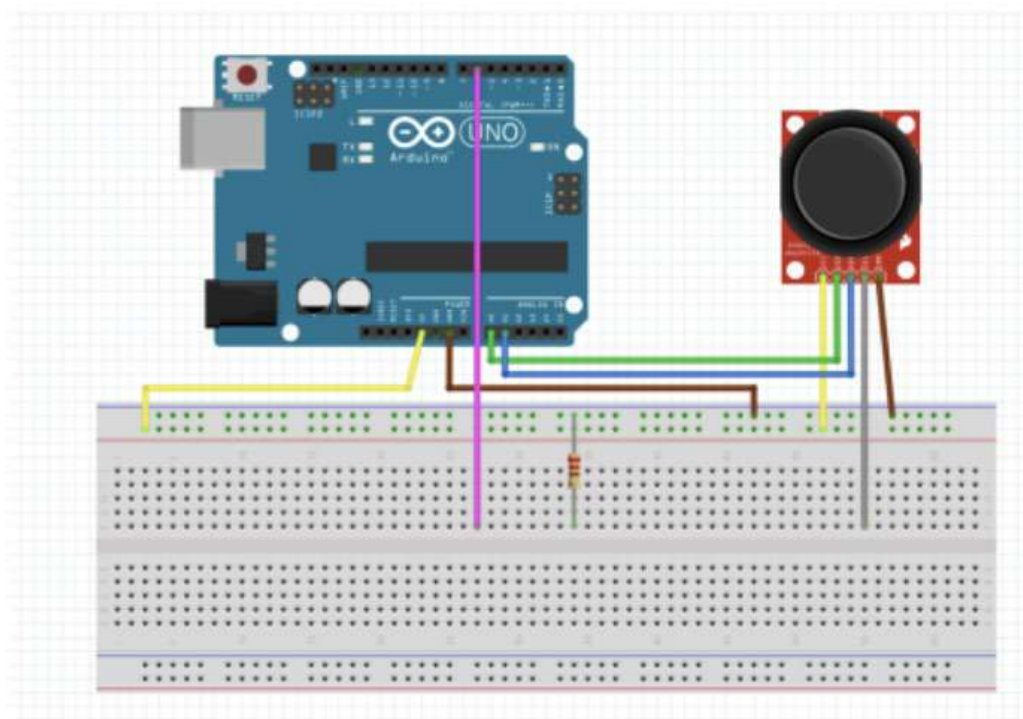


En este caso, cada pin analógico puede generar valores desde 0 a 1023, sin embargo en su estado de reposo se encuentra aproximadamente entre los siguientes valores:



## Montaje

En este caso, vamos a tener un joystick el cual va a generar señales que representan las posiciones (x,y)



## Código

Suba el siguiente código a la placa:

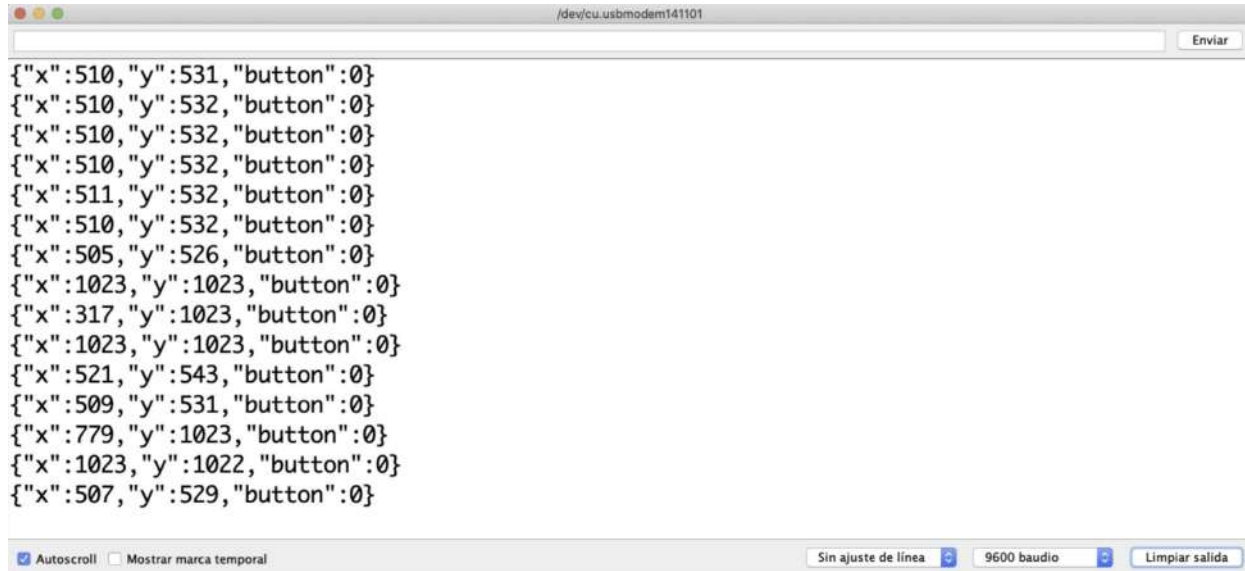
```
// Constantes para definir los pines
int xPos = A1;
int yPos = A0;
int BUTTON=3;

int LED = 13;
// Auxiliares
int x=0;
int y=0;
int sw=0;
char msm[50];
void setup() {
  Serial.begin(9600);
  // Activamos los pines de entrada
  pinMode(xPos, INPUT);
  pinMode(yPos, INPUT);
  pinMode(BUTTON, INPUT);
  // Activamos los pines de salida
  pinMode(LED, OUTPUT);
}
void loop() {
  //Leemos el valor de los pines analogos
  x = analogRead(xPos);
  y= analogRead(yPos);
  // Leemos el valor del pin
  sw = digitalRead(BUTTON);
  // Creamos un String
  sprintf(msm, "{\\"x\\":%i,\\"y\\":%i,\\"button\\":%i}", x, y, sw);
  // Imprimimos el string creado
  Serial.println(msm);
  //Añadimos un retraso
  delay(500);
}
```



## Monitor serial

Abra el monitor serial y mueva el joystick para observar las distintas señales que se generan.



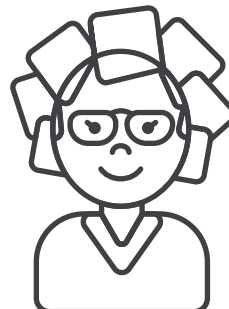
The screenshot shows the Arduino Serial Monitor window with the title bar indicating the port as /dev/cu.usbmodem141101. The main text area displays a series of JSON objects representing joystick coordinates and button states. The status bar at the bottom shows settings: Autoscroll is checked, Mostrar marca temporal is unchecked, Sin ajuste de línea is selected, 9600 baudio is set, and a Limpiar salida button is present.

```
{ "x": 510, "y": 531, "button": 0 }
{ "x": 510, "y": 532, "button": 0 }
{ "x": 510, "y": 532, "button": 0 }
{ "x": 510, "y": 532, "button": 0 }
{ "x": 511, "y": 532, "button": 0 }
{ "x": 510, "y": 532, "button": 0 }
{ "x": 505, "y": 526, "button": 0 }
{ "x": 1023, "y": 1023, "button": 0 }
{ "x": 317, "y": 1023, "button": 0 }
{ "x": 1023, "y": 1023, "button": 0 }
{ "x": 521, "y": 543, "button": 0 }
{ "x": 509, "y": 531, "button": 0 }
{ "x": 779, "y": 1023, "button": 0 }
{ "x": 1023, "y": 1022, "button": 0 }
{ "x": 507, "y": 529, "button": 0 }
```

## Referencias

1. <https://www.youtube.com/watch?v=okvUaG2BRBo&>
2. <https://aprendiendoarduino.wordpress.com/2018/10/16/joystick-arduino/>
3. <http://www.eveliux.com/mx/curso/conversion-analogico-digital.html>

**Has completado el Taller #2.  
¡Ánimos con el siguiente!**



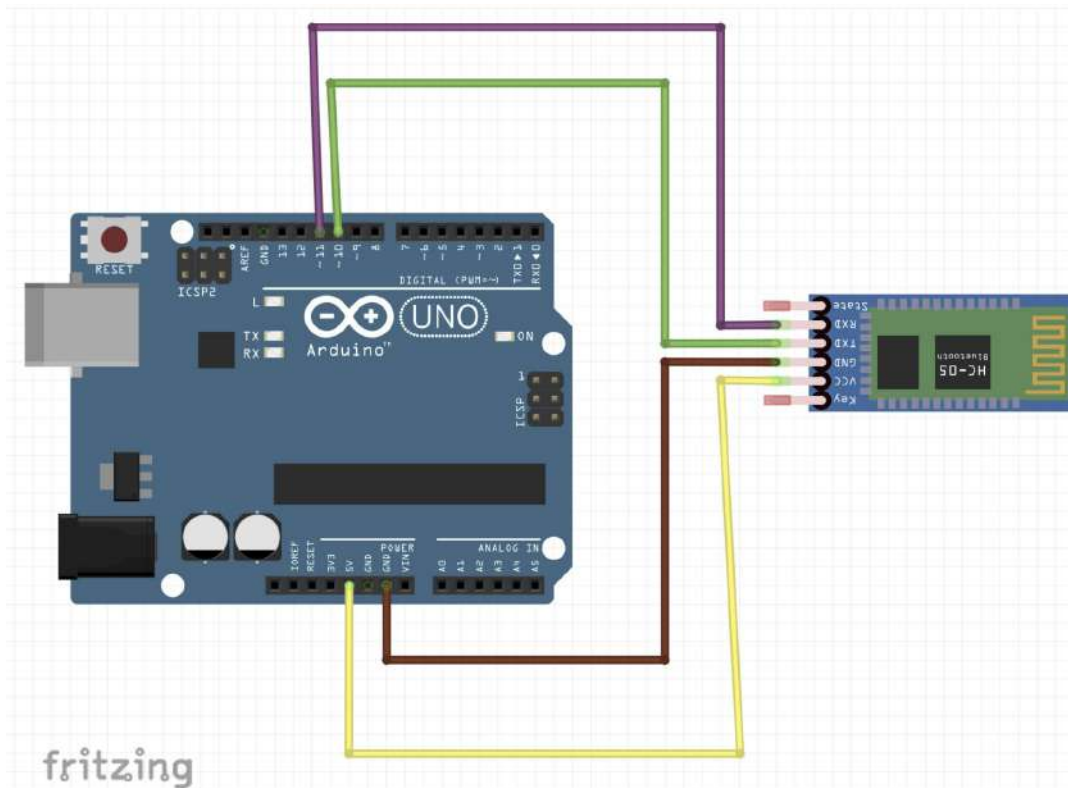
# Arduino - Taller #3

## Introducción

En el siguiente video se explica detalladamente el uso del módulo bluetooth y su configuración:

<https://www.youtube.com/watch?v=5SmKOUHhmWk&>

## Montaje

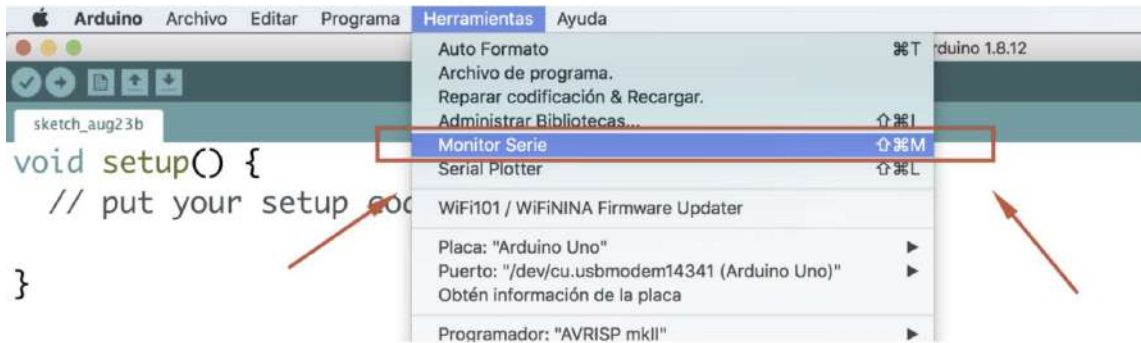
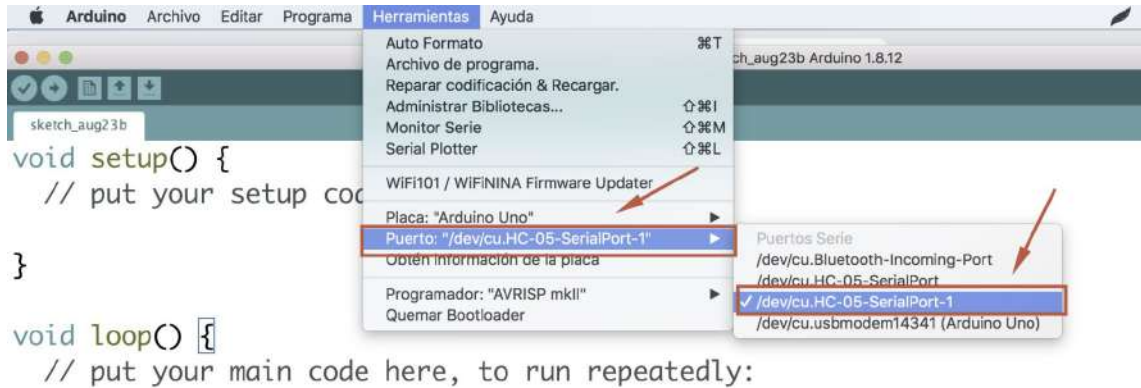


## Código

Suba el siguiente código:

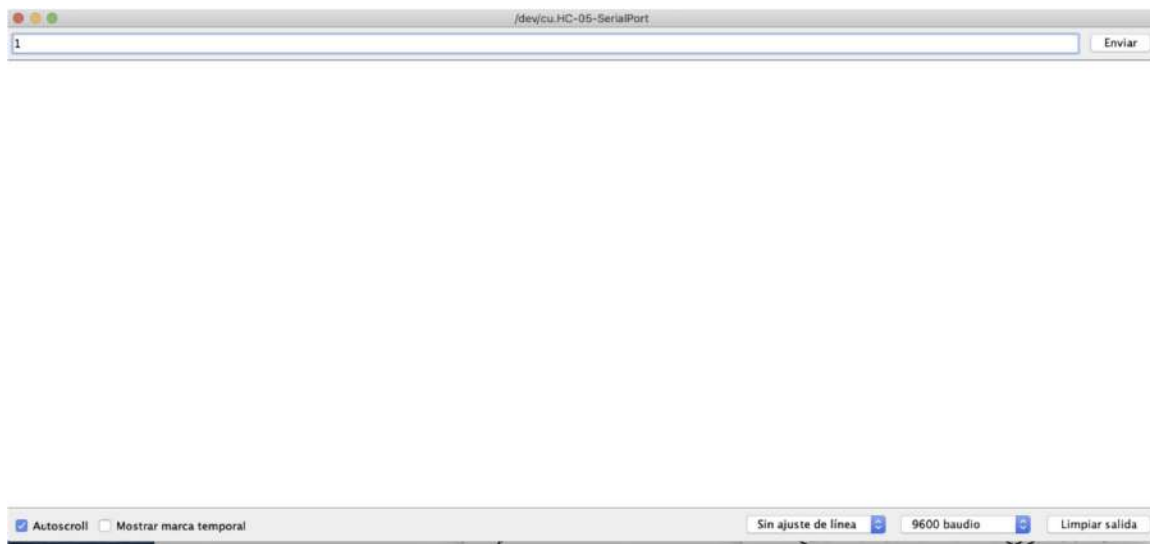
```
#include <SoftwareSerial.h> // Incluimos la librería
SoftwareSerial
SoftwareSerial BT(10,11); // Definimos los pines RX y TX del
Arduino conectados al Bluetooth
// Constantes para definir el pin de LED
int LED= 13;
//Auxiliar
char b;
void setup()
{
// Inicializamos el puerto serie
Serial.begin(9600);
// Inicializamos el puerto serie BT
BT.begin(9600);
// Activamos el pin de salida del LED
pinMode(LED,OUTPUT);
}
void loop()
Taller 3 2
{
if(BT.available()) // Si llega un dato por el puerto BT se envía al
monitor serial {
b=BT.read();
Serial.write(b);
// Cuando el valor es enviado, de acuerdo al protocolo
// se le añaden 48 unidades, por lo que las restamos
if(b-48 == 1){
// Activamos el pin del led
digitalWrite(LED,HIGH);
}
else if(b-48 == 2){
// Bajamos el pin del led
digitalWrite(LED,LOW);
}
}
if(Serial.available()) // Si llega un dato por el monitor serial se
envía al puerto BT {
BT.write(Serial.read());
}
}
```

Ahora conéctese al puerto del módulo HC-05 y abra el monitor serie como se muestra en las siguientes imágenes:

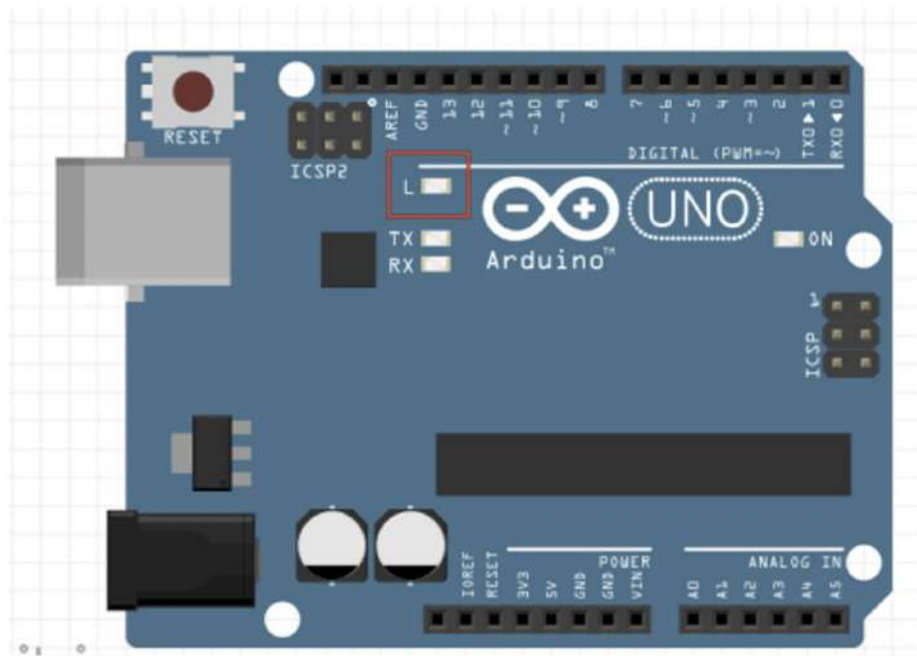


## Monitor Serie

Envíe 1 para encender el led del pin 13 y envíe 2 para apagarlo.



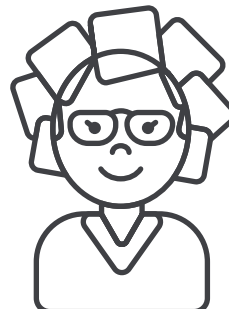
Este es el led del pin 13:



## Referencias:

1. <https://www.instructables.com/Easiest-Arduino-Bluetooth-Control-With-Android/>
2. [https://naylampmechatronics.com/blog/24\\_configuracion-del-modulo-bluetooth-hc-05\\_usa.html](https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05_usa.html)
3. <https://www.youtube.com/watch?v=5SmKOUHmWk&>

**Felicitaciones, has completado el último taller.**





Unity

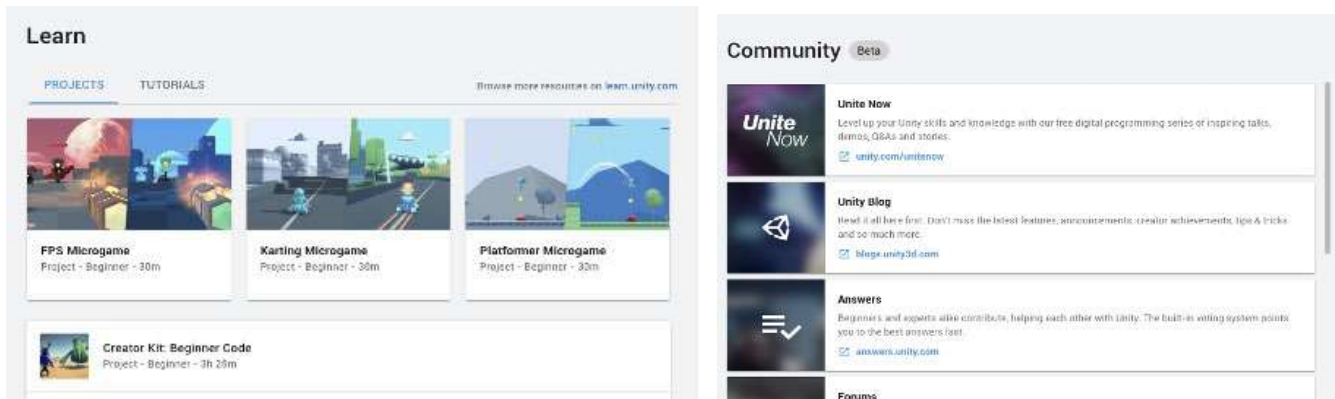
The image features a central purple oval with the word "Unity" in white. The oval is composed of three overlapping circles of varying shades of purple. The background is white with abstract geometric elements: two horizontal lines at the top right, a purple triangle, a purple square, and a purple diamond; a purple 'X' and a purple square at the bottom left; a purple circle and a purple diamond at the bottom right; and a purple circle and a purple diamond at the bottom center.



# Introducción a Unity

Para descargar Unity, es necesario instalar un asistente: Unity Hub. Este te muestra tu historial de proyectos y te permite crear nuevos. Lo puedes descargar desde este enlace: [https://unity3d.com/get-unity/download?\\_ga=2.88932571.1144251109.1602134903-1894545095.1601662740](https://unity3d.com/get-unity/download?_ga=2.88932571.1144251109.1602134903-1894545095.1601662740).

También cuenta con una barra de opciones en las cuales además de la pestaña principal de proyectos te sale:



- 1 **Learn:** puedes ver los tutoriales o practicar la herramienta con los proyectos predeterminados.
- 2 **Community:** simplemente es como un blog en donde puedes ver las últimas noticias, los foros, y consultar ayuda con los demás miembros de la comunidad de Unity.
- 3 **Install:** te sale la versión que tienes de Unity, y si lo quieres desinstalar o agregarle módulos. También revela la ubicación del archivo en el buscador de tu pc.

## Crear Proyectos

Al crear un nuevo proyecto en Unity, se puede escoger entre modo 2D y 3D. Ambos tienen sus diferencias ya que manejan algunos elementos distintos.

### Tipos de juegos:

**3D:** Utilizan materiales tridimensionales con texturas renderizadas en su superficie. La cámara se puede mover en todas las direcciones. (Usar Modo 3D)

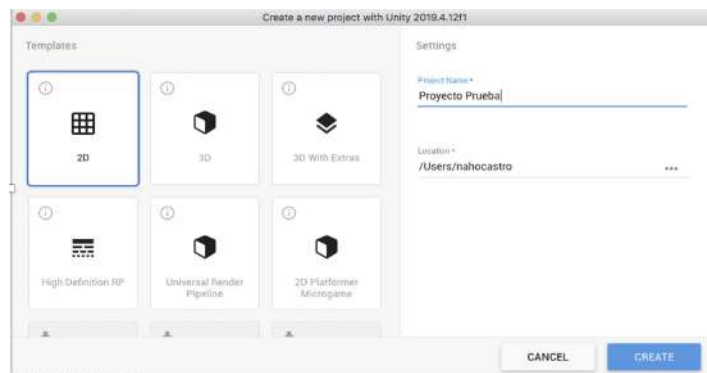
**2.5D:** Utilizan una vista ortográfica fija. (Usar Modo 3D)

**2D:** Utilizan gráficas planas (conocidas como sprites) sin dimensiones 3d.

Al iniciar [Unity Hub](#) se abre una pantalla así en donde se muestran los juegos que has creado. También ahí mismo puedes crear un nuevo proyecto.

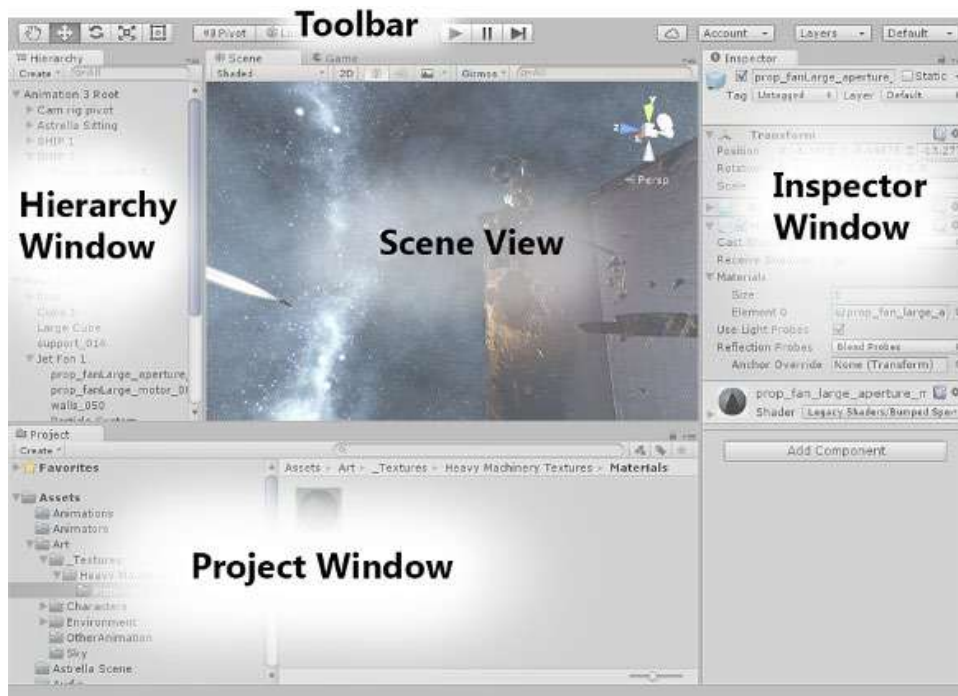


Para crear un nuevo proyecto seleccionamos donde dice [NEW](#) o [NUEVO](#) y aparece un panel con las opciones para escoger el tipo de proyecto (2d, 3d, etc).



## Interfaz

La interfaz de Unity está organizada en la siguiente distribución. En la siguiente página se explica con más detalle cada sección:



En la [ventana de proyecto \(Project Window\)](#) se encuentran los assets de librería, o los elementos como materiales y texturas, disponibles para ser usados. Deben ser importados al proyecto.

En la [ventana de escena \(Scene View\)](#) se puede observar como está quedando la escena que está contruyendo. Es practicamente un previsualizador. Muestra una perspectiva en 2D o 3D, según el modo de trabajo en el que se encuentre el proyecto.

En la ventana de jerarquía (Hierarchy Window) se organizan todos los elementos y assets del juego por orden alfabético y otro orden que desees (se puede cambiar en preferencias). También se puede usar el Parenting que es básicamente ligar algunos objetos a uno principal (el padre) y esos objetos ligados (los hijos) reciben las mismas configuraciones que el padre.

En la [ventana del inspector \(Inspector Window\)](#) se puede ver y editar todas las propiedades del objeto que esté seleccionado.

La [Barra de Herramientas](#) consiste en cinco controles básicos. Cada uno se relaciona con diferentes partes del Editor.



Transform Tools – se utilizan con la Scene View



Palancas(toggles) de Gizmo del Transform – afecta la visualización del Scene View



Play/Pause/Step Buttons – se usan para probar el juego en la previsualización



Play/Pause/Step Botones – utilizado con la vista del juego (GameView)



Diseño del Desplegable– controla el arreglo de todas las Vistas

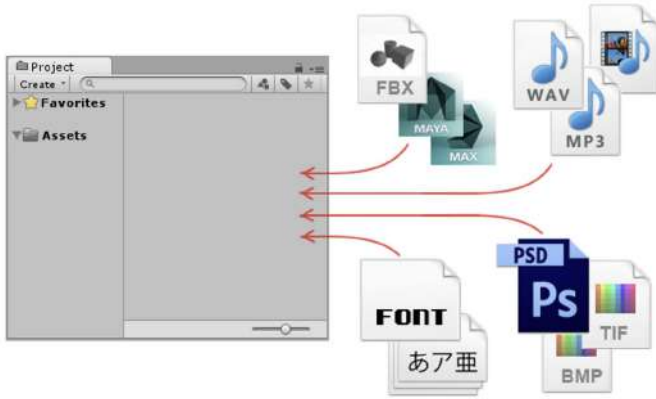


Layers Desplegable –controla qué objetos son mostrados en el Scene View

**Nota:** Además de este orden, es posible acomodar a tu gusto el espacio de trabajo.

## Assets

Son todos los objetos importados al proyecto. Vienen en una variedad de formatos ya que incluyen archivos de audio, de 3D, bitmaps, entre otros. Se recomienda guardarlos en una carpeta, clasificarlos según su tipo de formato y después importarlos al proyecto.



Unity también tiene una tienda para comprar e importar assets al proyecto:

<https://assetstore.unity.com/top-assets/top-free>

## Crear un juego

Unity utiliza un tiempo de ejecución estándar Mono para el scripting.

Unity soporta dos lenguajes nativamente:

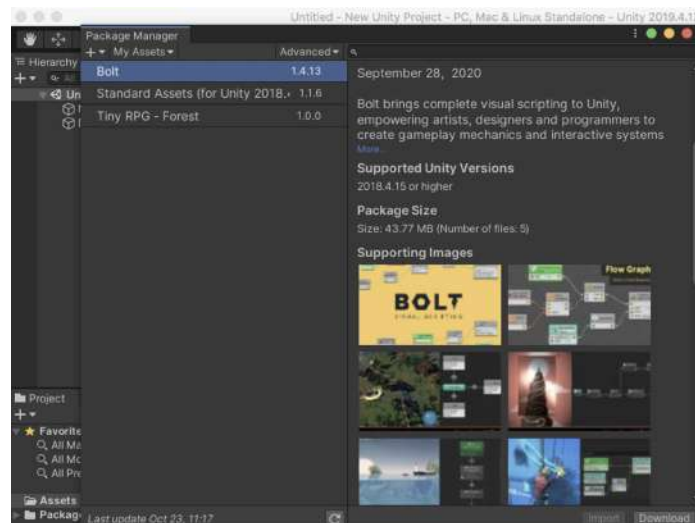
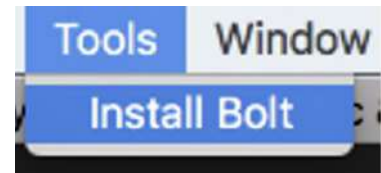
- C# (C-sharp), un lenguaje estándar de la industria similar a Java o C++;
- UnityScript, un lenguaje diseñado específicamente para uso con Unity y modelado tras JavaScript.

El comportamiento de los objetos de Unity es controlado mediante componentes adjuntos. Estos se pueden crear utilizando scripts.

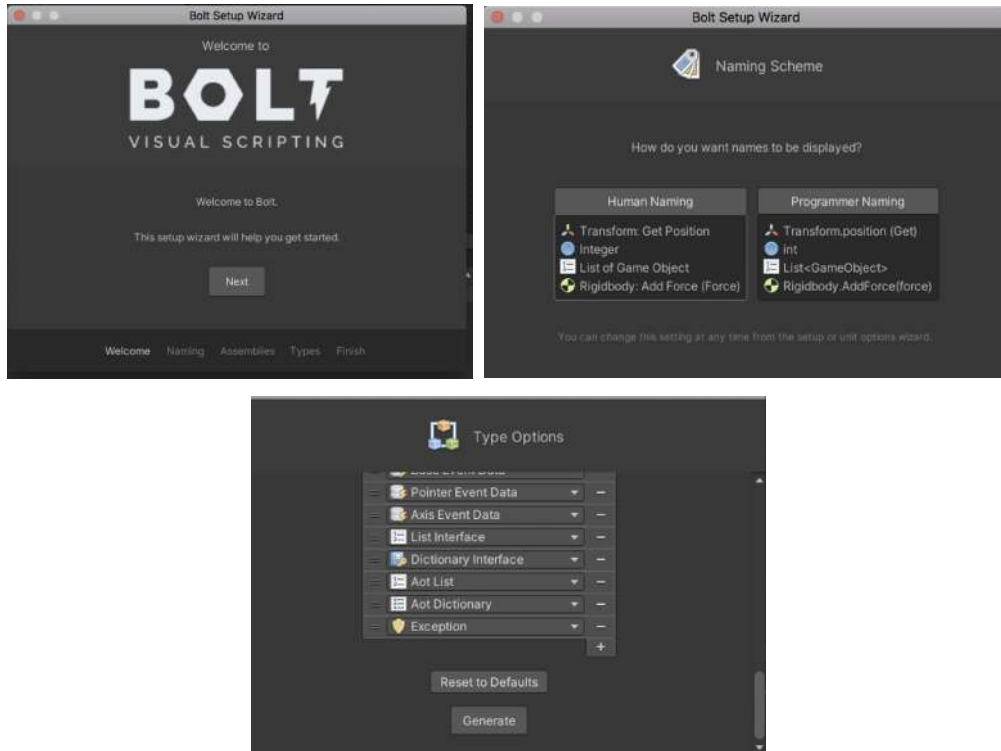
Sin embargo para hacer más fácil el proceso de aprendizaje utilizaremos el plugin Bolt en vez de los scripts en C# para realizar el curso.

## Bolt

Bolt se instala desde el Package Manager de Unity. Ahi se busca en la opción de My Assets, se descarga y luego se importa. Una vez finalizado este proceso se escoge la ventana Tools (Herramientas) en la parte de arriba, y se escoge Install Bolt.



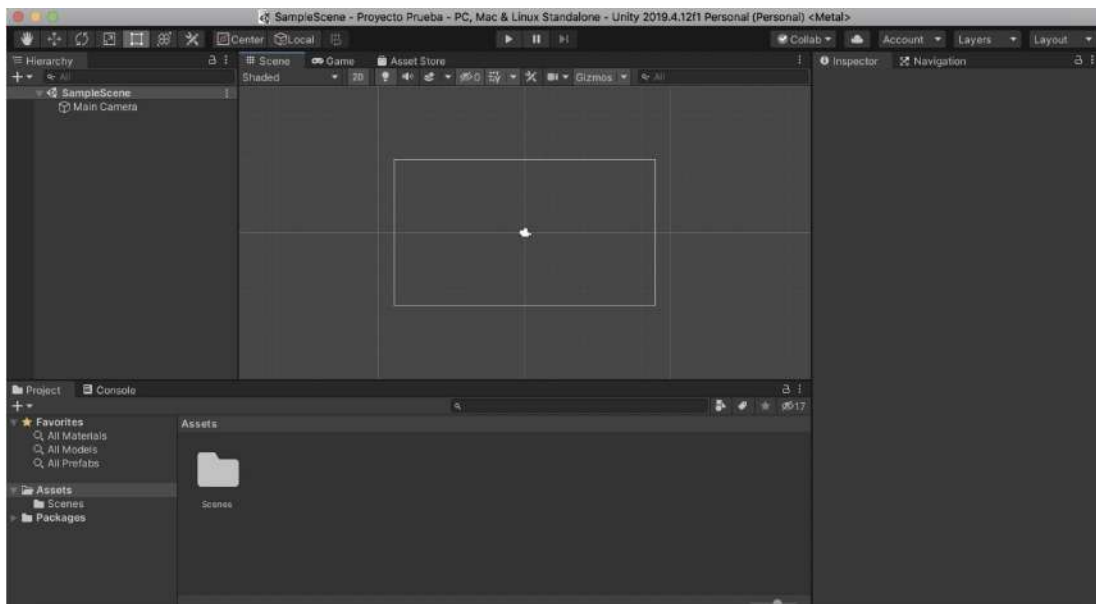
Al final aparece una ventana de bienvenida a bolt en donde se puede configurar. Escoger la opción de **Human Naming** ya que es más amigable al usuario. En las Assembly options dejarlo como está y seguir a la última página en donde se debe escoger Generate. Una vez listo este proceso, ya tienes bolt. Ahora hay que descargar el Bolt Kit: Platformer Tutorial Assets de igual manera como instalaste Bolt. (Esto es para practicar).



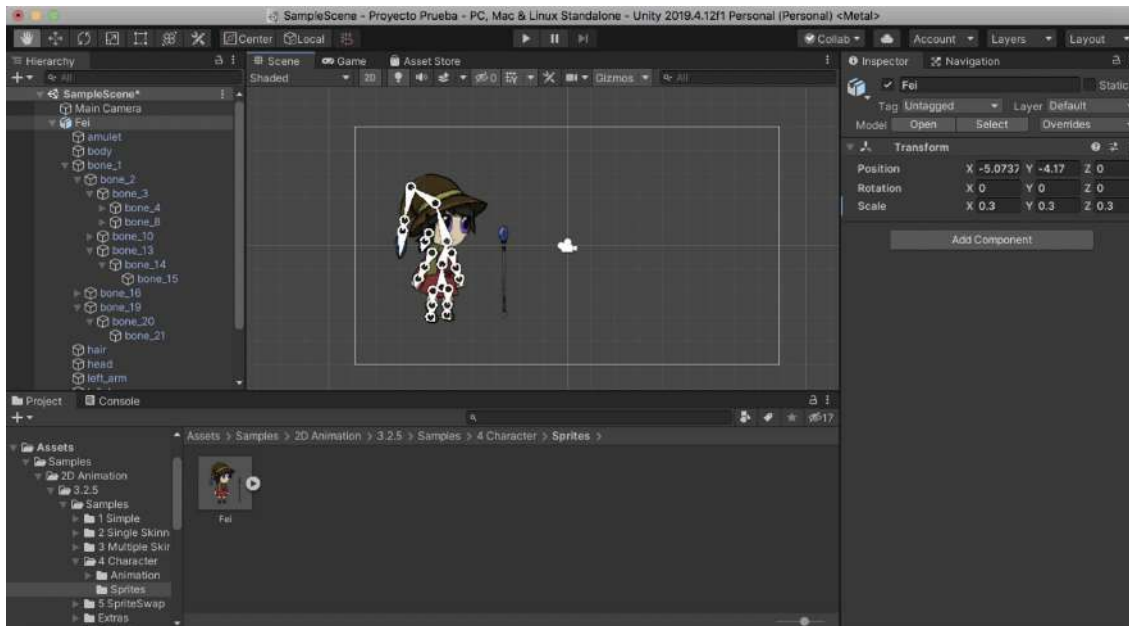
### La Interfaz

En el espacio de trabajo se pueden agregar los assets a la Escena, a esta se le puede cambiar la vista de 2D a 3D mediante este boton **2D**

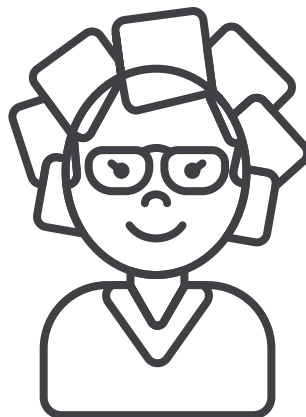
Se escoge el sprite de los assets que se importaron y se arrastra a la escena.



Listo, una vez terminado este recorrido puedes experimentar por tu cuenta y seguir el taller que hemos desarrollado para que practiques la herramienta en el desarrollo de video juegos.



**¡Una vez haya leído este documento  
estás listo para comenzar con tu primer  
juego en Unity!**



# Taller Juego Quiz

## Múltiples Respuestas en Bolt-Unity

### Set Up

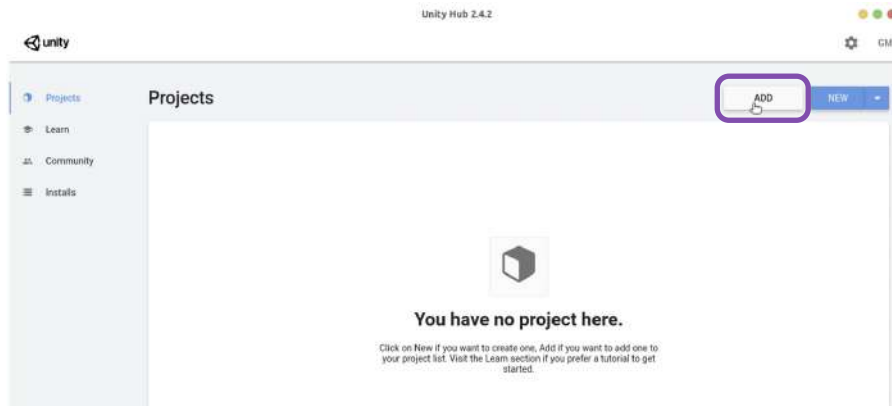
#### Requerimientos:

- Unity con Bolt instalado
- Archivo del proyecto de prueba descargado
- Visual Studio Code Instalado

### Desarrollo

En la siguiente guía explicaremos el funcionamiento de nuestro juego base y el uso de los componentes de BOLT en Unity.


- 1 Primero abrimos Unity Hub en nuestros computador.
- 2 Ahora abriremos el proyecto Multiple-resp-quiz, haciendo click en el botón [ADD](#).

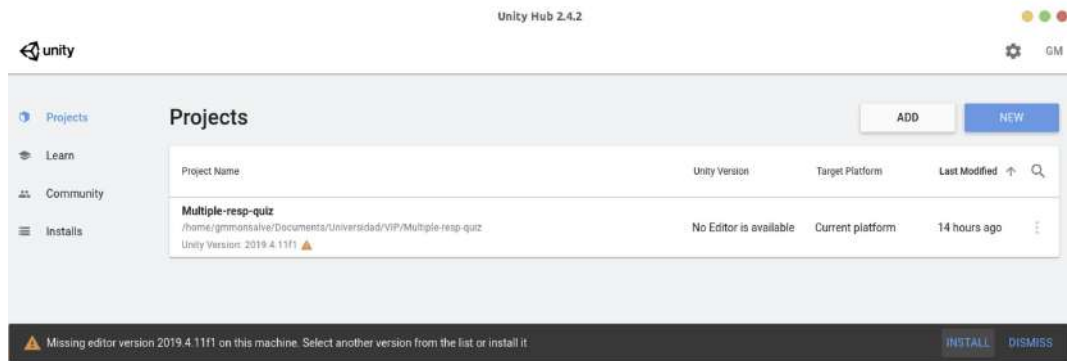


- 3 Siguiendo a esto aparecerá en tu pantalla el buscador de archivos en computador, busca donde tengas la carpeta de [Multiple-resp-quiz](#) que descargaste para realizar el tutorial.

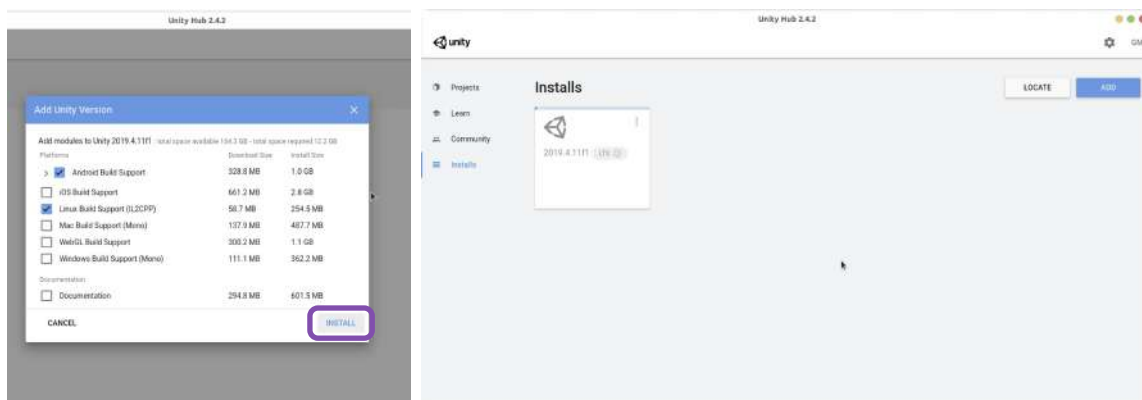




- 4 Ahora aparecerá el proyecto en Unity Hub con una alerta de versión. Hacemos click en la alerta  del proyecto. Seguidamente en la barra de notificación que aparecerá en la parte posterior hacemos click en **INSTALL**.



- 5 A continuación aparecerá una ventana emergente, en esta se listan las características de herramientas y la versión de Unity. Dejamos la configuración por defecto y hacemos click en **INSTALL**.



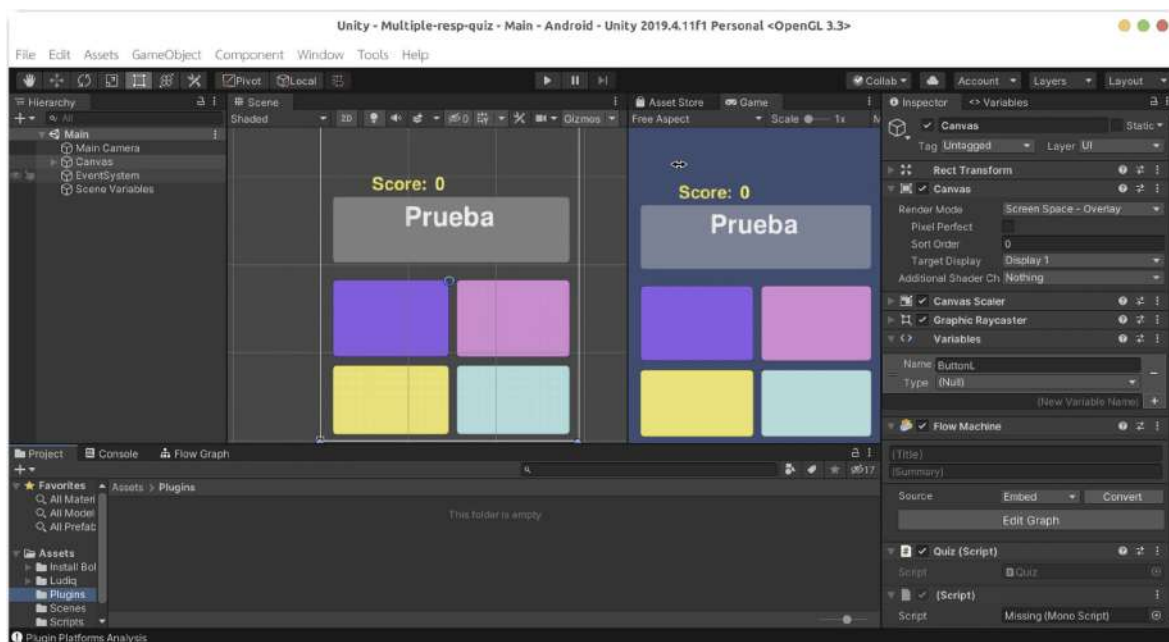
Ahora la versión requerida está instalándose, solo debemos esperar hasta que termine y estaremos listos para empezar.

- 6 Una vez instalada la versión volvemos al menú izquierdo en la opción **PROJECTS**. Ahora hacemos click en el proyecto.

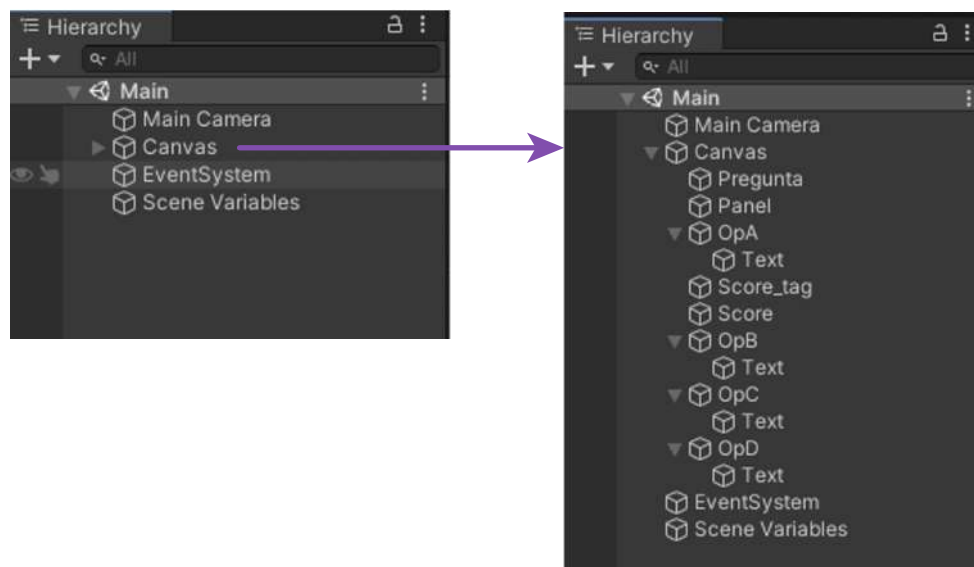




## Demo: Quiz Múltiples Respuestas

- 1 Una vez que el proyecto haya abierto se nos mostrará la ventana principal de Unity. Para ver la vista del proyecto hacemos click en el objeto Canvas el menú de HIERARCHY.

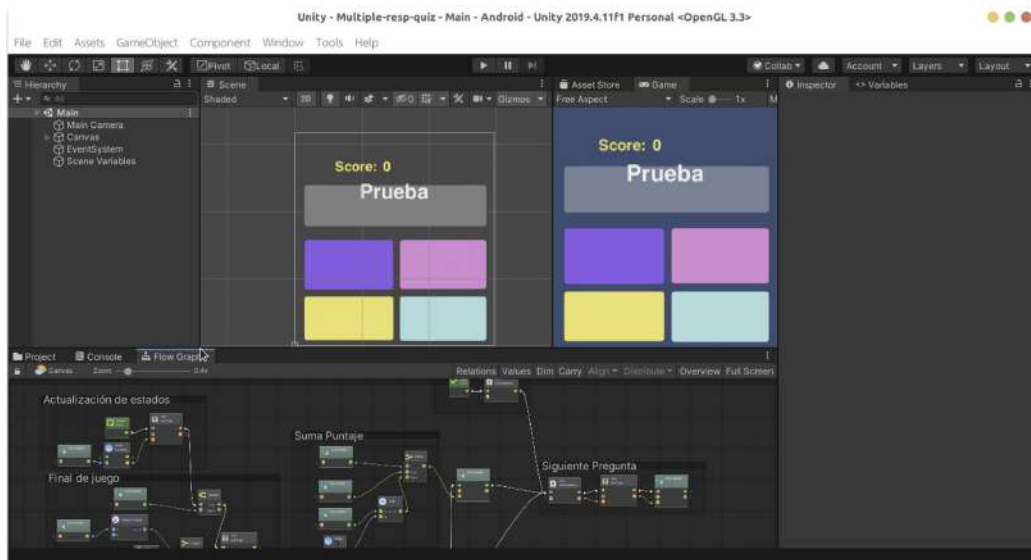


En este menú encontraremos todos los GameObjects que tiene nuestro proyecto. Si hacemos click al triángulo que apunta a Canvas se desplegarán todos los objetos dentro de este objeto. Esto funciona igual con todos los GameObjects.



- 2 ¡Probemos el juego! Haz click en  en la parte superior y prueba su funcionamiento en la pestaña  Game. Haz click en los botones para interactuar.

- 3 Ahora vemos el funcionamiento de este juego con Bolt. Ve a la pestaña Flow Graph del menú que nos aparece en la parte inferior.

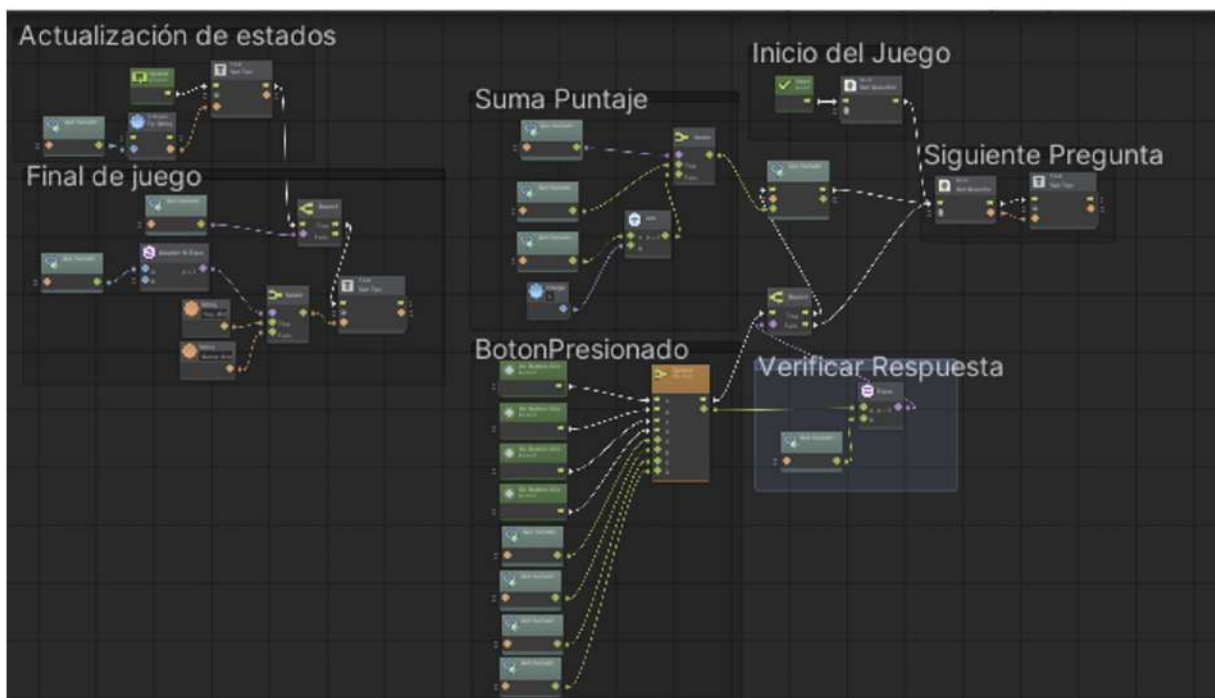


- 4 Hagamos click ahora en Full Screen **Full Screen**.

Ahora podremos observar los 7 bloques principales de la estructura de nuestro Quiz.

## Bloques agrupados

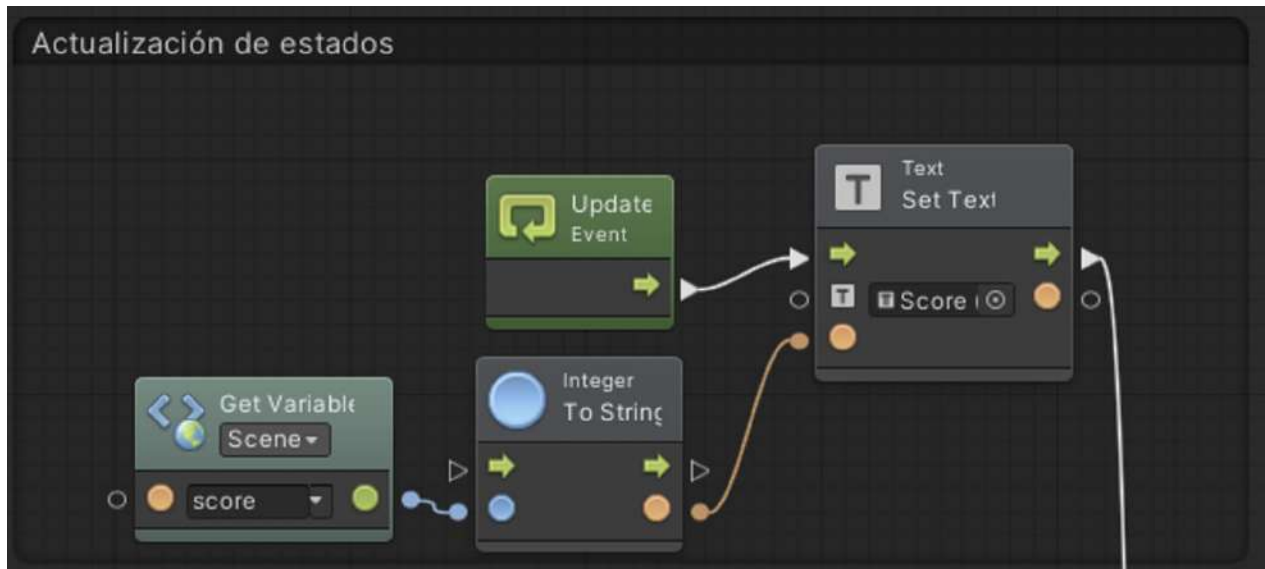
A continuación, podrás ver de manera general la estructura de la lógica implementada en nuestro juego de Unity con Bolt.



## Actualización de estados

En este encontraremos 4 bloques de código divididos de la siguiente forma.

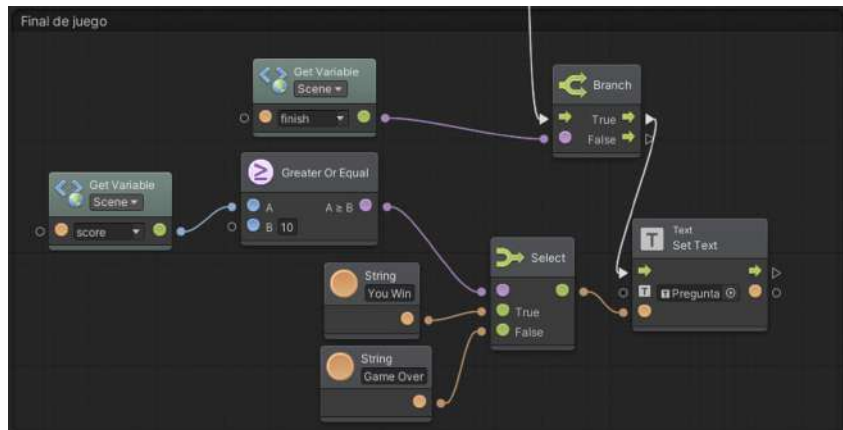
- Update (Event): Este bloque representa la acción de actualización de todo el juego. todo lo que se conecte a este bloque será actualizado de forma continua. En este caso Update es conectado al bloque de acción Set Text de forma que todo el tiempo realiza esta acción.



- Text Set Text (Score): Este bloque coloca el texto en el objeto Score que se le es dado a través del puerto inferior izquierdo.
- Integer to String: Convierte un valor numérico al formato de cadena (string).
- Get variable (Score): Este bloque obtiene el valor que tiene la variable global score.

## Final de Juego

Este apartado contiene la lógica en bloques necesaria para desplegar un mensaje de final del juego basado en la variable de escena finish.

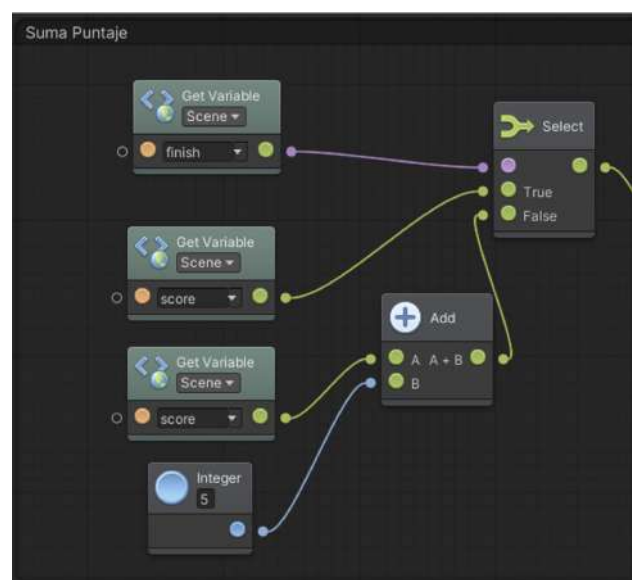


- **Branch:** Este es un bloque de control que permite según el valor de una variable, en este caso finish, decidir a donde llevar el flujo de la acción. Si finish tiene valor verdadero enviará el flujo del juego a accionar la función bloque Set Text.
- **Select:** Este otro bloque de control nos permite seleccionar un valor según una condición lógica que se conecta a través del puerto
- **String:** A través de este bloque podremos establecer valores fijos o constantes de tipo string en nuestro juego. Para este caso los usamos para definir las cadenas “You Win” y “Game Over” en nuestro juego.
- **Greater or Equal:** Por medio de este bloque podremos saber, dados dos elementos numéricos, si el primero es mayor o igual al segundo. La salida del bloque es precisamente verdadero o falso dependiendo de como se evalúe la condición.

## Suma Puntaje

Para que el sistema de puntaje funcione se utiliza principalmente un Select. Este bloque toma el valor de la variable de escena finish, si es verdadera entonces el juego habrá terminado y procederá simplemente tomar y devolver el valor de la variable score que ya se tiene. Por otra parte, si el valor de finish es falso se llamará el bloque Add.

El bloque Add tiene como funcionalidad tomar dos valores numéricos, en este caso el de la variable score y un valor entero fijo Integer, que luego procede a sumar para devolverlo a través de Select.

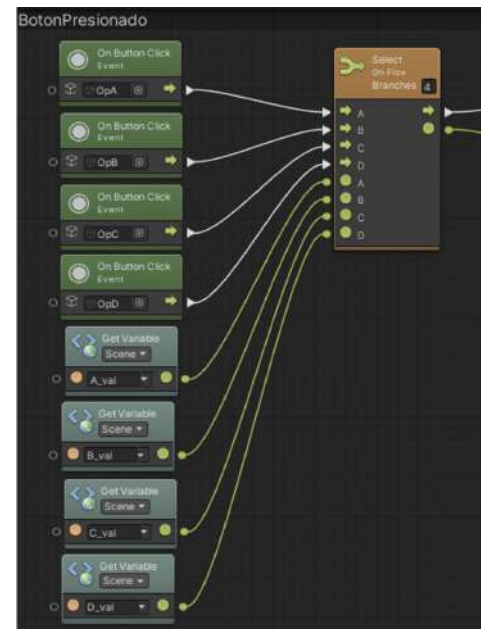


## Botón Presionado

En este apartado tenemos tres tipos de bloque principales. El primero es el On Button Click. Este bloque es un bloque de evento, cada vez que un botón en el UI es presionado envía una señal al Select on Flow con la cual se puede saber que botón ha sido presionado.

El segundo bloque son los de Get Variable cuya función es devolver el valor de una variable de escena especificado, para que esto ocurra debe ser llamada alguna de ellas por el bloque Select.

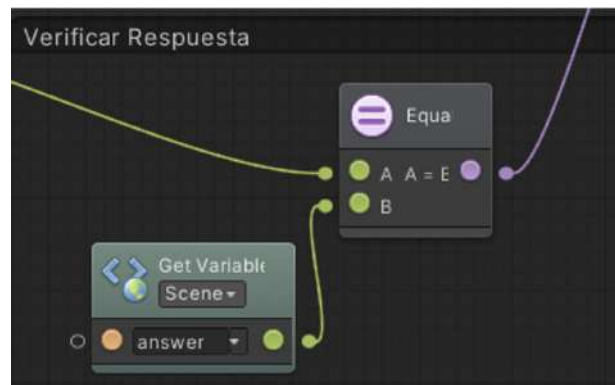
Finalmente podemos encontrar al bloque Select. Este bloque toma el flujo de los On Button Click y según el que haya sido presionado llamará a la respectiva variable de escena asignada para tomar el valor que contenga para luego retornarlo.



## Verificar respuesta

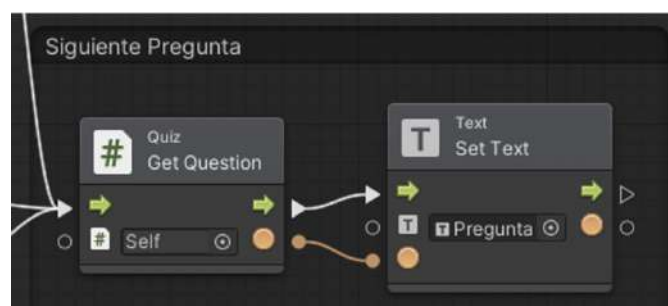
Una vez se obtiene la respectiva respuesta en este bloque se hace la comprobación de su valor. Para esto se usa el bloque Equal.

Este bloque recibe la información de la respuesta dada y la compara con la respuesta correcta. Luego de hacer la comparación devuelve el valor lógico verdadero o falso hacia la siguiente etapa.



## Siguiente pregunta

En este apartado se llama inicialmente al bloque que contiene el método Get Question del script Quiz. Este método se encarga de obtener las preguntas establecidas en el código y pasarlas a la función Set Text quien despliega el texto en el objeto de texto Pregunta.





## Inicio del juego

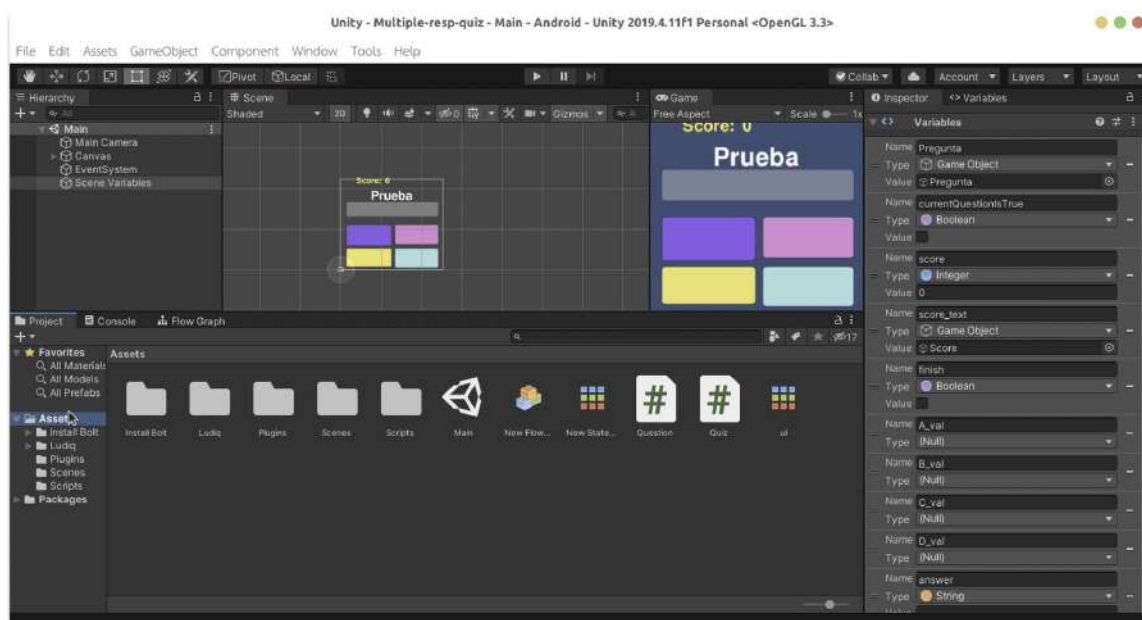
Aquí en este apartado se usa el bloque Start el cual es un bloque de evento que se ejecuta solo una vez al inicio de nuestro juego. Al activarse envía el flujo de acción hacia el bloque del método Set Questions del script Quiz el cual se encarga de cargar las preguntas al juego y sus opciones de respuesta.



## Unity Scripts y Variables de Escena

Ahora veremos un poco más acerca de los scripts y sus métodos usados en los bloques de Bolt.

- 1 Primero en el mismo menú inferior donde vimos nuestro grafo de bloques hacemos click en la pestaña Project. Seguidamente hacemos click en la carpeta de Assets. Aquí encontraremos dos scripts: Quiz y Question.



- 2 Vamos a hacer doble click en estos dos archivos. Para este caso estos archivos se abrirán en el programa Visual Studio Code.



## Archivo Question.cs

En este script tenemos la definición de la clase Question. A esta se le dan tres atributos:

- Statement [string]: Es el cuerpo de la pregunta que se va a realizar en el juego como por ejemplo "2+1 = ?"
- Options [List]: Es una lista que contendrá todas las opciones de respuesta para la pregunta.
- Answer [string]: Es la respuesta correcta a la pregunta.

También podemos encontrar un constructor para Question que permite crear un objeto pregunta según unos parámetros establecidos y posteriormente se asignan esos valores a los atributos. Además de esto contamos con los métodos get\_statement(), get\_answer(), get\_options(). Estos métodos nos permiten obtener los valores que tenemos almacenados en los respectivos atributos de cada método.

```
using System.Collections;
using System.Collections.Generic;

public class Question {
    public string statement;
    public List<string> options;
    public string answer;

    public Question(string statement, List<string> options, string answer)
    {
        this.statement = statement;
        this.options = options;
        this.answer = answer;
    }

    public string get_statement() {
        return statement;
    }

    public string get_answer() {
        return answer;
    }

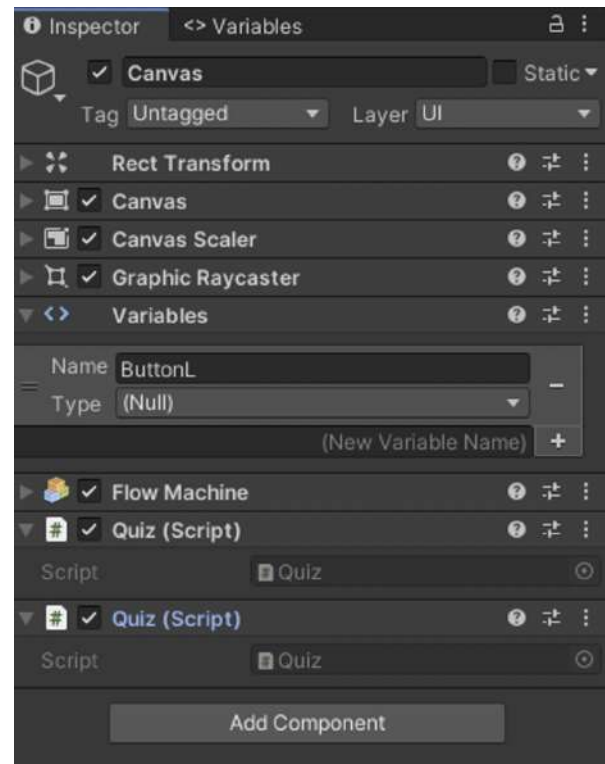
    public List<string> get_options() {
        return options;
    }
}
```

## Archivo Quiz.cs

Este archivo está ligado a nuestro juego a través del objeto Canvas. Si en nuestro menú de Hierarchy hacemos click en Canvas podremos ver las propiedades de este objeto en el ventana de inspector que aparece al lado izquierdo de la ventana.

En el contenido de este script podremos encontrar la definición de la clase Quiz. Esta cuenta con el atributo questions: una lista de preguntas cuya estructura definimos en el script Question.cs. Luego tenemos los métodos Start(), Update() que coinciden con nuestros bloques de eventos usados en Bolt.

Ahora quedan los métodos setQuestions(), getQuestions() y setOptions().



El método setQuestions() se encarga de crear las preguntas según los parámetros que hemos especificado en Question. Lo que se hace es crear una lista de opciones, seguidamente agregar a la lista questions (lista de preguntas) una nueva pregunta. Para crear esa nueva pregunta le brindamos al método constructor un statement, la lista de opciones options y la respuesta correcta a la pregunta.

En este método encontramos también la línea:

```
Variables.ActiveScene.Set("answer", pregunta.get_answer());
```

A través de esta podemos establecer el valor de cualquier variable de escena que tengamos en Unity Bolt. En este caso estamos dándole a la variable de escena "answer" el valor o el contenido de la respuesta de la pregunta que se obtiene a través de get\_answer().

Al igual que el método setQuestions() tenemos al método setOptions() que se encarga de darle a las variables de escena A\_VAL, B\_VAL, C\_VAL y D\_VAL que son las que almacenan las cuatro opciones de la pregunta.

Además de guardar los valores también en este método colocamos el texto de las opciones en nuestros botones tras cada pregunta. Para esto se utiliza la siguiente línea de código

```
GameObject.Find("OpA").GetComponentInChildren<Text>().text =
pregunta.get_options().ElementAt(0);
```

Aquí Buscamos a través de `GameObject.find()` el botón cuyo nombre sea OpA y seguidamente le asignamos un elemento de nuestra lista de opciones.

El método `setOptions()` es llamado en método `getQuestion()`. Este método se encarga de tomar una pregunta de la lista de preguntas (`questions`) de forma aleatoria y luego retorna el statement de esta pregunta.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using Bolt;
using Ludiq;

public class Quiz : MonoBehaviour
{
    // Start is called before the first frame update
    public List<Question> questions = new List<Question>();

    void Start()
    {
    }

    void Update()
    {
    }

    public void setQuestions() {
        List<string> options = new List<string>
        {"Juego", "Baile", "Lugar", "Personaje"};
        questions.Add(new Question("La cumbia en Colombia es un:", options, "Baile"));

        List<string> options2 = new List<string> {"5", "4", "6", "3"};
        questions.Add(new Question("2+1 = ? : ", options2, "3"));

        List<string> options3 = new List<string>
        {"Medellin", "Cartagena", "Barranquilla", "Bogotá"};
        questions.Add(new Question("La 'arenosa' es la ciudad de:", options3, "Barranquilla"));
    }

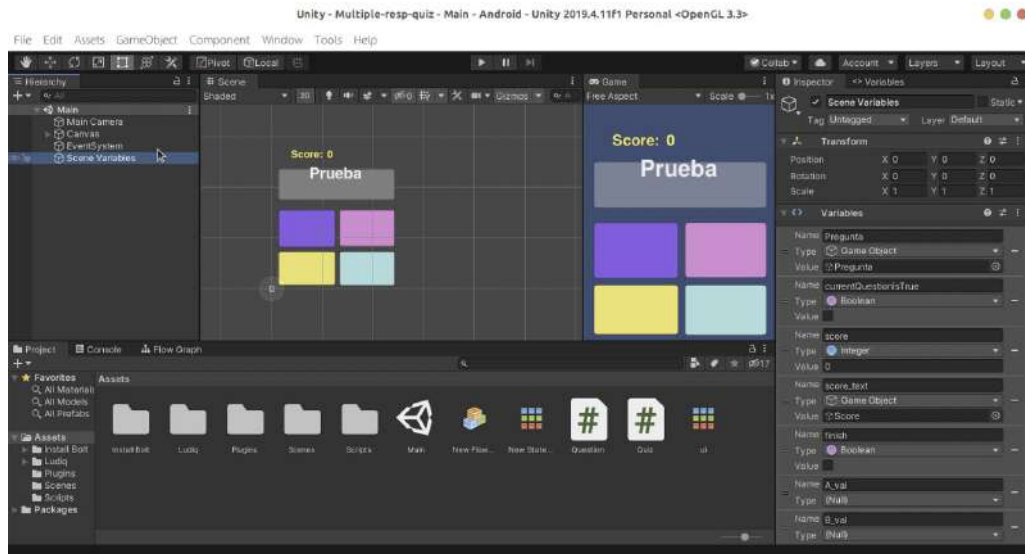
    public string getQuestion() {
        if(questions.Count == 0){
            string_question = "FINISH!";
            Variables.ActiveScene.Set("finish", true);
        }else{
            int randomNum = Random.Range(0, questions.Count);
            Question pregunta = questions[randomNum];
            string_question = pregunta.get_statement();
            Variables.ActiveScene.Set("answer",
pregunta.get_answer());
            setOptions(pregunta);
            questions.RemoveAt(randomNum);
        }
        return string_question;
    }

    public void setOptions(Question pregunta){
        Variables.ActiveScene.Set("A_val",
pregunta.get_options().ElementAt(0));
        Variables.ActiveScene.Set("B_val",
pregunta.get_options().ElementAt(1));
```

```
Variables.ActiveScene.Set("C_val",
pregunta.get_options().ElementAt(2));
Variables.ActiveScene.Set("D_val",
pregunta.get_options().ElementAt(3));
GameObject.Find("OpA").GetComponentInChildren<Text>().text =
pregunta.get_options().ElementAt(0);
GameObject.Find("OpB").GetComponentInChildren<Text>().text =
pregunta.get_options().ElementAt(1);
GameObject.Find("OpC").GetComponentInChildren<Text>().text =
pregunta.get_options().ElementAt(2);
GameObject.Find("OpD").GetComponentInChildren<Text>().text =
pregunta.get_options().ElementAt(3);
    }
}
```

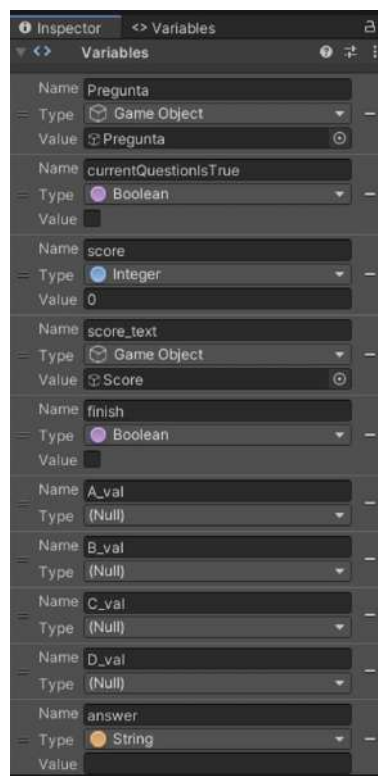
## Variables de escena

Nuevamente en nuestro menú Hierarchy que está en la parte izquierda hacemos click en el objeto SceneVariables.



En el lado izquierdo de la pantalla aparecerá, en la pestaña inspector, el listado de las variables de escena de nuestro juego.

Estas variables nos permitirán comunicarnos desde nuestros scripts hasta nuestros bloques de Bolt en todo el proyecto de nuestro juego.



**¡Felicidades,  
has completado toda la guía!**

**Esperamos que durante el desarrollo  
de este proyecto te hayas divertido y  
aprendido mucho sobre mundo de la  
programación para video juegos.**

**¡Muchas gracias por participar!**

