Financiado por la Unión Europea NextGenerationEU

Plan de Recuperación, Transformación y Resiliencia

Junta de Andalucía | Consejería de Desarrollo Educativo y Formación Profesional

CAPACITACIÓN TÉCNICA

AULAS DIGITALES INTERACTIVAS

Módulo 5: ADI STEAM

PCT #EcoDigEdu

Documentación complementaria:
5.7 Programación de microbit con Makecode

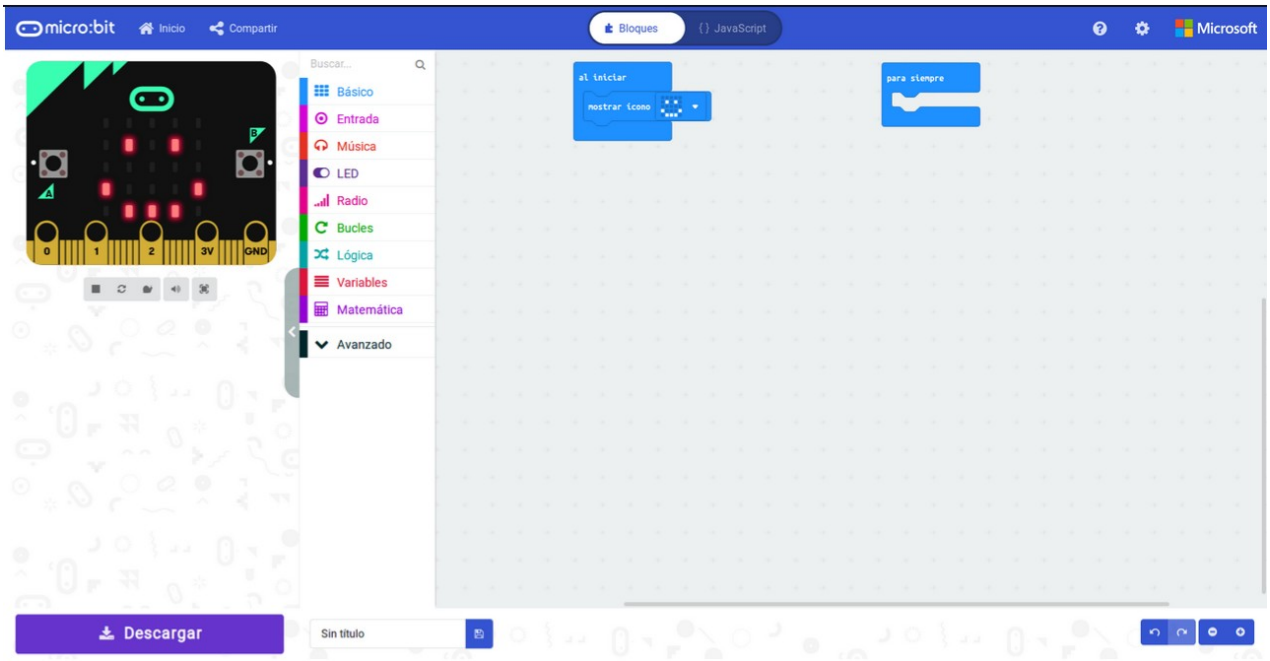
19/12/25

Índice

1. Software de simulación Makecode....3
1.1. Interfaz gráfica....4
1.2. Simulador virtual....5
1.3. Compatibilidad con extensiones....6
1.4. Descarga y Transferencia de código....6
1.5. Integración con GitHub y Compartición de Proyectos....7
1.6. Ejemplos y documentación de ayuda....8
2. Uso de botones y luces LED....8
3. Comunicación por radio....12
3.1. Transmisión de tipos de datos diferentes....14
4. Sensor de temperatura....18
4.1. Conversión entre unidades de temperatura....19
4.1. Diseñar una estación meteorológica....22
5. Sensores de movimiento....23
5.1. Giroscopio....26
5.2. Juego de dados por parejas....29
5.3. Juego de dados por equipos....30
6. Otras actividades....30
6.1. Código morse....30
6.2. Competición de reflejos....34
6.3. Cálculo mental....37
6.4. Llegar a 21....41

1. Software de simulación Makecode

El entorno de desarrollo Microbit Makecode es una plataforma en línea diseñada por Microsoft para programar la placa BBC micro:bit.



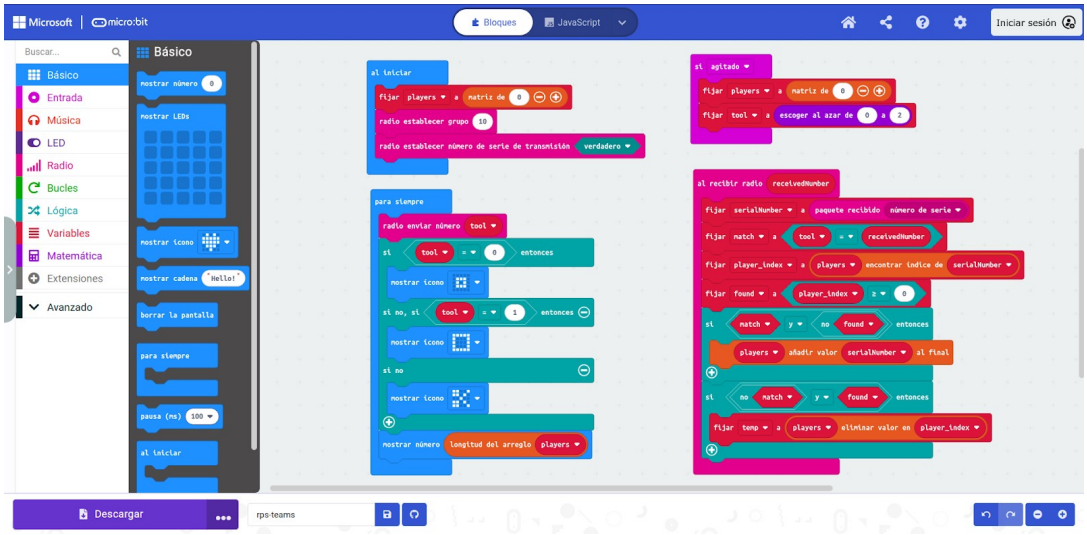
Algunas recomendaciones de uso del entorno de desarrollo Makecode:

- **Familiarízate con el entorno:** Antes de empezar, dedica tiempo a explorar la interfaz de Makecode. Conoce las diferentes pestañas: **<Bloques>**, **<JavaScript>** y **<Python>**. La interfaz de bloques es la más intuitiva para principiantes, ya que permite arrastrar y soltar comandos.
- **Empieza con ejemplos sencillos:** No te compliques al principio. Makecode tiene una gran cantidad de proyectos de ejemplo integrados. Comienza con proyectos simples como hacer que el LED de la Microbit muestre una cara sonriente, un corazón o tu nombre.
- **Usa las categorías de bloques:** Los bloques de código están organizados por categorías de colores (por ejemplo, **<Básico>**, **<Entrada>**, **<Música>**). Esto facilita encontrar los comandos que necesitas. Familiarízate con las funciones de cada categoría.
- **Utiliza el simulador:** El simulador es una de las herramientas más poderosas de Makecode. Te permite probar tu código en un Microbit virtual sin necesidad de tener el dispositivo físico. Usa el simulador para depurar errores y verificar si el código funciona como esperas.

- **Pasa de bloques a código:** Una vez que te sientas cómodo con los bloques, haz clic en la pestaña de **<JavaScript o Python>** para ver cómo se traduce tu código de bloques. Esto te ayudará a comprender la sintaxis de la programación basada en texto.
- **Utiliza comentarios:** Al escribir código más complejo, es una buena práctica añadir comentarios para explicar qué hace cada sección del programa. Aunque la programación de bloques es visual, los comentarios ayudan a ti y a otros a entender tu lógica.
- **Guarda tu trabajo:** Makecode guarda tus proyectos automáticamente en el navegador, pero siempre es recomendable guardar una copia de seguridad en tu ordenador. Para ello, haz clic en **<Guardar>** para descargar el archivo **<.hex>** que contiene tu código.
- **Comparte tus proyectos**: Puedes compartir tu proyecto pulsando el botón **<Compartir>** para generar un enlace único a tu proyecto. Esto es útil para compartirlo con compañeros, profesores o en redes sociales, ya que cualquiera con el enlace puede acceder al proyecto y duplicarlo para su uso. Para proyectos más avanzados o para colaboración, puedes publicar tu código directamente en **<GitHub>**. Haz clic en el botón **<GitHub>** dentro de la opción de **<Compartir>** para crear un repositorio con tu código. Esto facilita el control de versiones y el trabajo en equipo.
- **Busca proyectos en la web:** Hay una comunidad muy activa de Microbit. Busca proyectos en la página oficial de Makecode, esto te dará ideas y te inspirará para realizar proyectos más avanzados.
- **Prueba los complementos o extensiones:** Makecode tiene extensiones que añaden bloques de código para trabajar con sensores externos, como módulos de servo, sensores de distancia o módulos GPS. Explora las extensiones para ampliar las capacidades de tu Microbit.
- **No temas equivocarte:** La programación se basa en prueba y error. Si algo no funciona, revisa el código, prueba el simulador y pide ayuda a la comunidad.

1.1. Interfaz gráfica

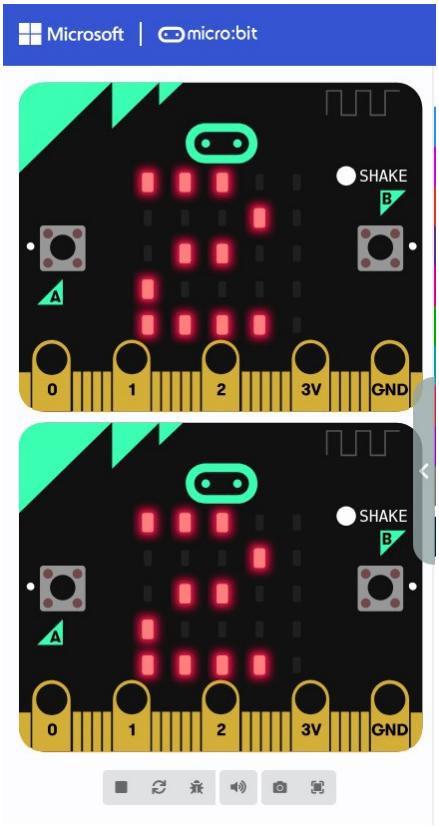
La interfaz gráfica de Makecode es muy intuitiva. Utiliza un sistema de bloques tipo drag and drop (arrastrar y soltar), similar a Scratch, lo que facilita la programación sin necesidad de escribir código.



También ofrece un editor de JavaScript y Python, permitiendo a los usuarios avanzar a lenguajes de programación textuales.

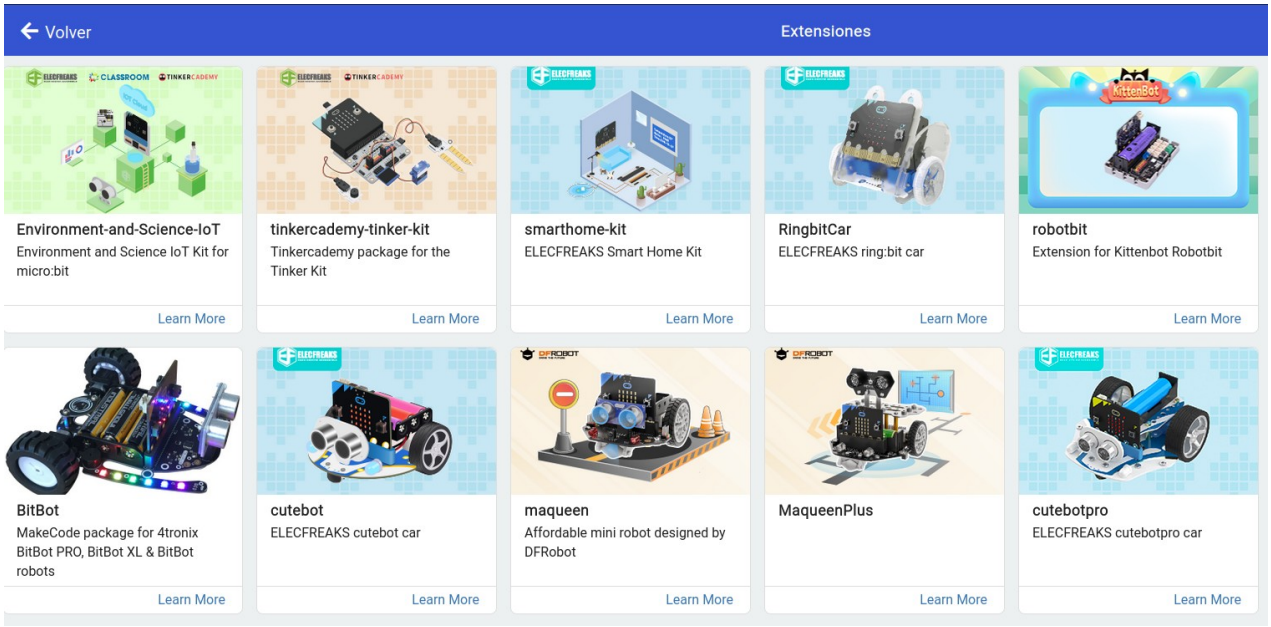
1.2. Simulador virtual

Antes de transferir el código a la micro:bit, se puede probar en un simulador interactivo que imita el comportamiento del hardware. Permite visualizar el resultado del código en los botones, LEDs, sensores y otros componentes de la placa.



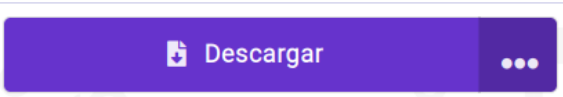
1.3. Compatibilidad con extensiones

Además de las funciones básicas incluidas en la placa Micro:bit, se pueden agregar bibliotecas para trabajar con sensores, motores, comunicación Bluetooth y otros módulos externos.

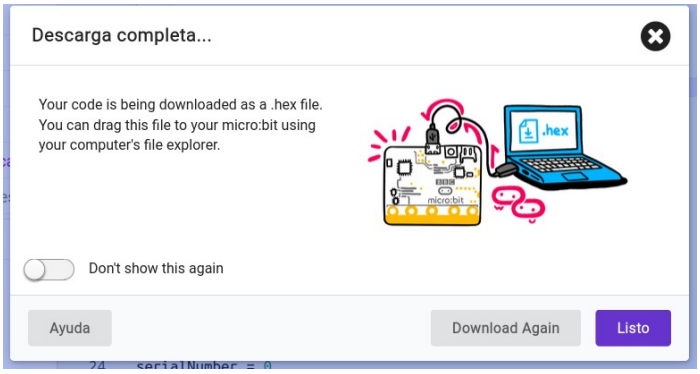


1.4. Descarga y Transferencia de código

El proceso para enviar el código generado a la placa Microbit es muy sencillo. Se descarga el programa pulsando el botón descargar.



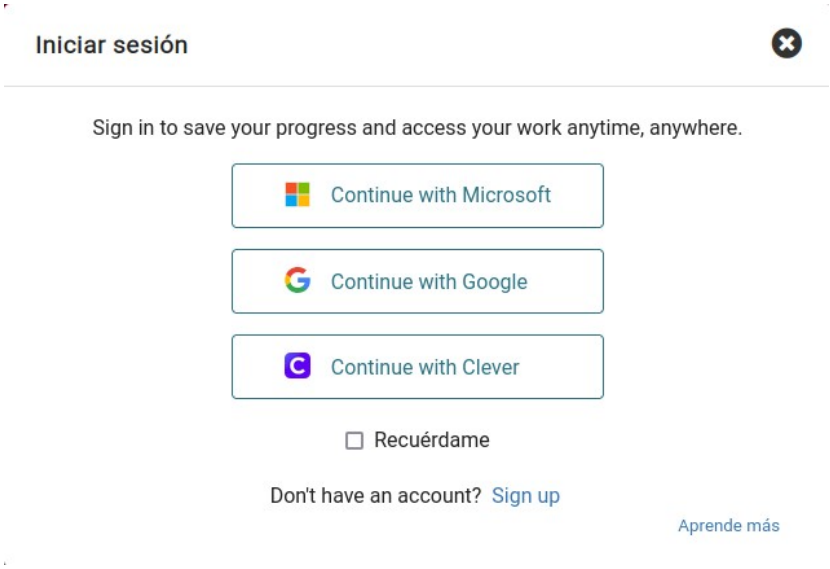
El código se descarga como un archivo .hex, que luego se transfiere a la micro:bit simplemente arrastrándolo a su unidad USB.



También permite la conexión directa vía WebUSB, eliminando la necesidad de descargar archivos manualmente como se explica en el siguiente **video**.

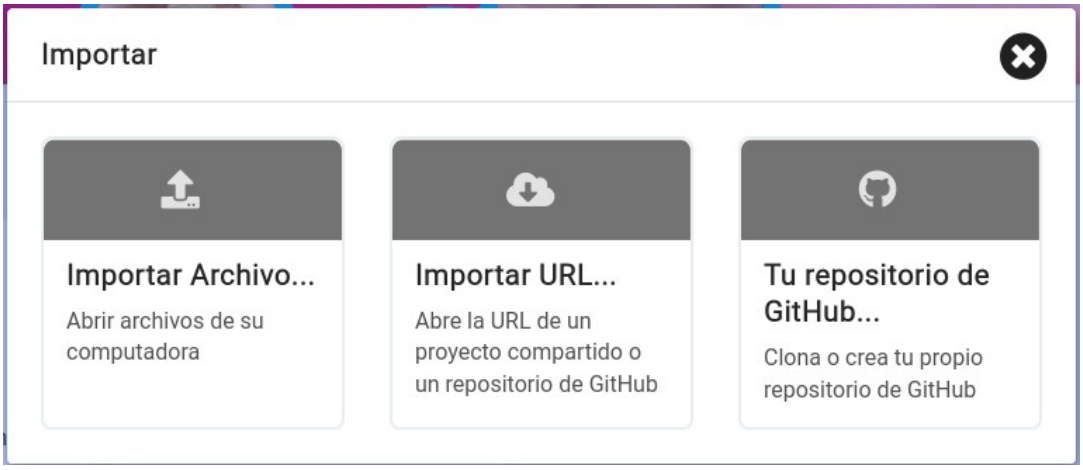
1.5. Integración con GitHub y Compartición de Proyectos

Permite crear una cuenta en la que guardar los proyectos.



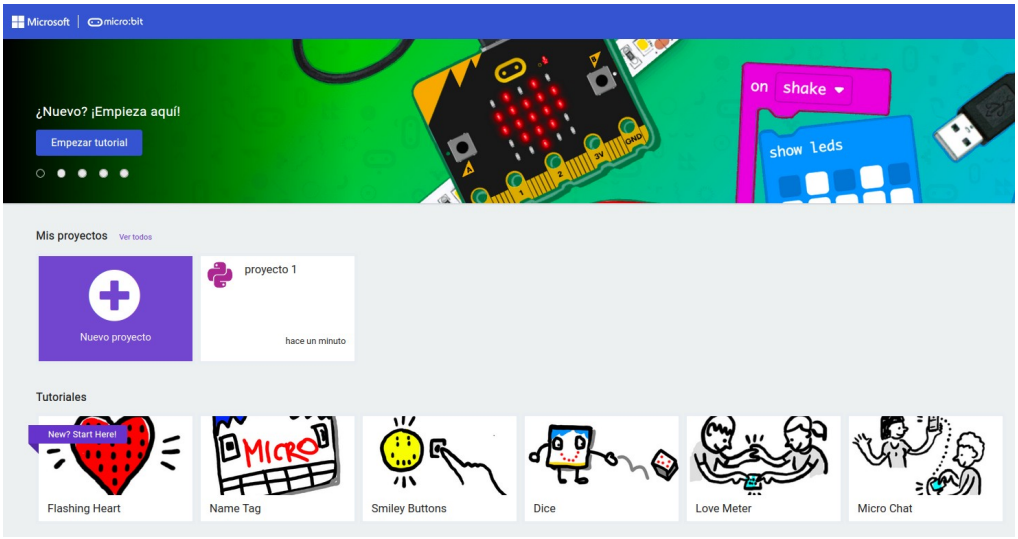
Una vez registrados, los proyectos se pueden guardar en nuestra cuenta o enviar a un repositorio colaborativo GitHub.

Posteriormente los proyectos se pueden importar desde un archivo local, archivo web o desde el repositorio GitHub.



1.6. Ejemplos y documentación de ayuda

La web Makecode Microbit incluye multitud de aplicaciones de ejemplo y tutoriales guiados.

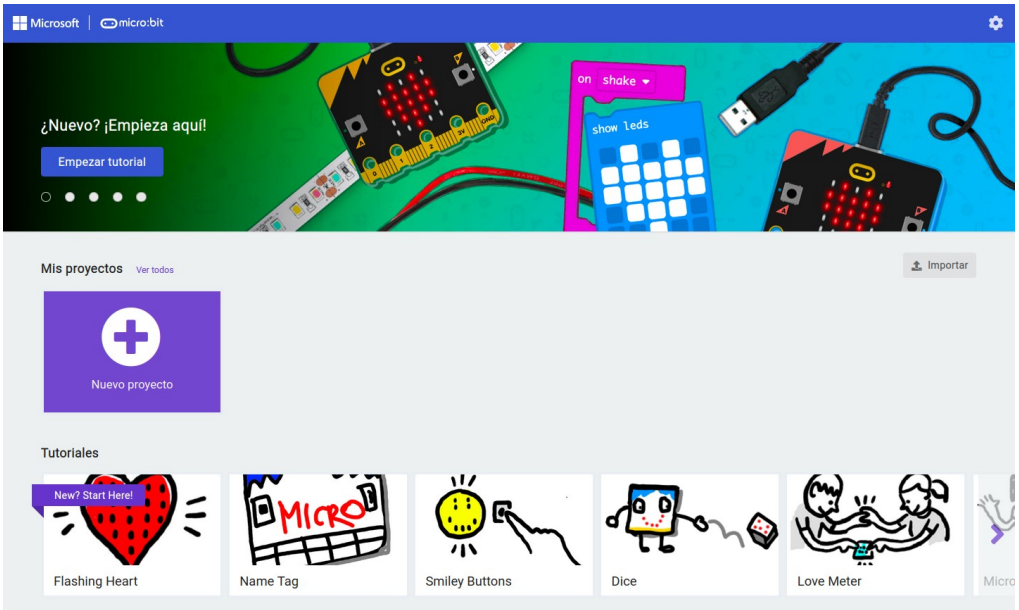


2. Uso de botones y luces LED

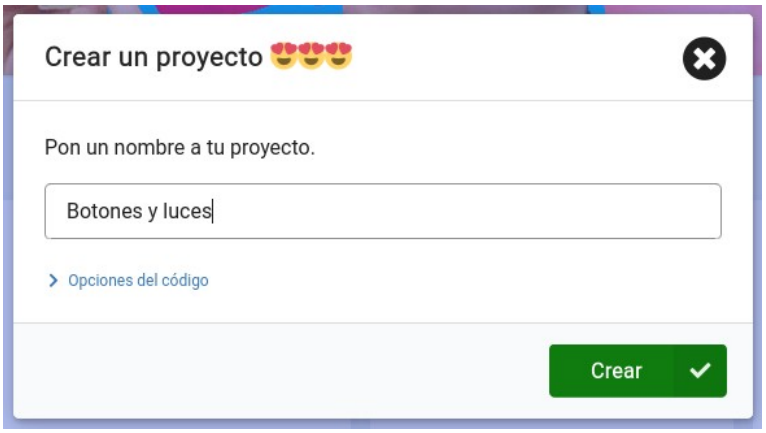
La manera más sencilla de simular el funcionamiento de una placa Microbit es utilizando la plataforma Makecode. Para ello vamos a crear un pequeño programa para la utilización de los botones y las luces LED que incorporan la placa.

Pasos a seguir

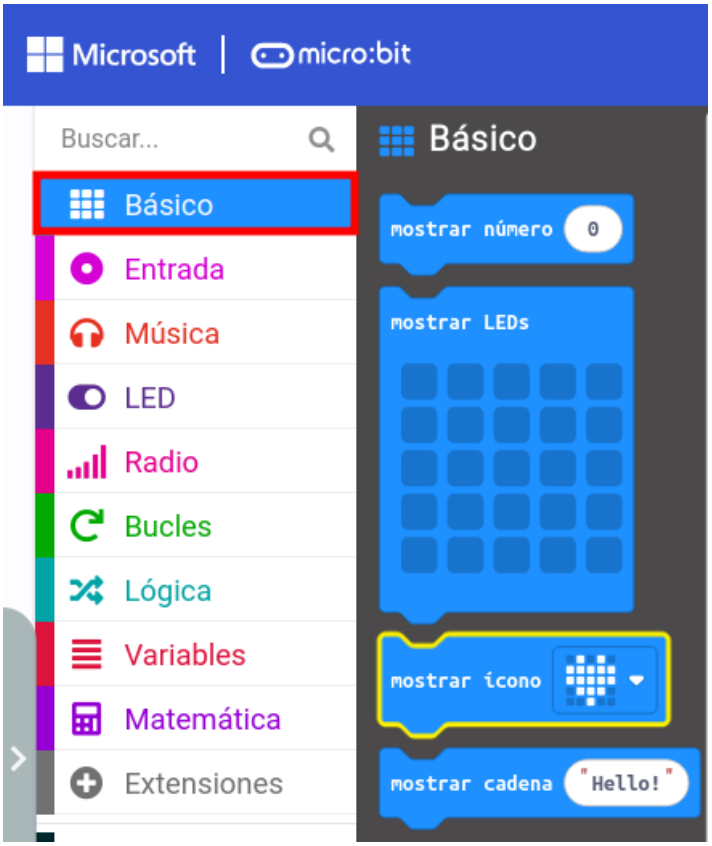
Entramos en la plataforma Microbit **Makecode** y creamos un proyecto nuevo.



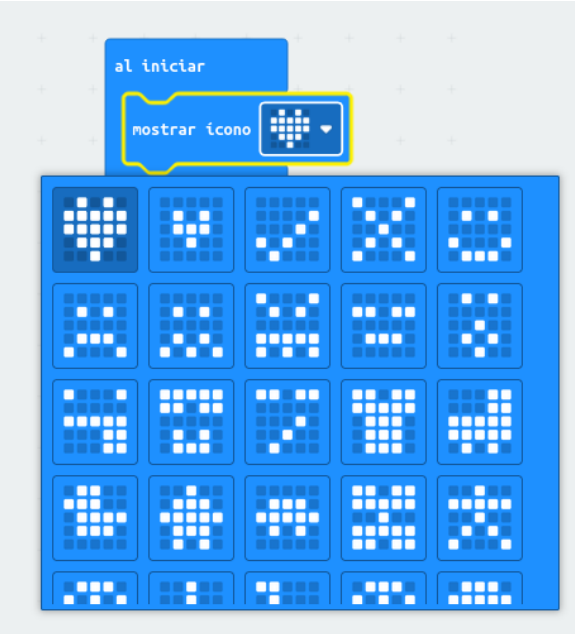
Le asignamos un nombre y pasaremos al entorno de desarrollo de Makecode.



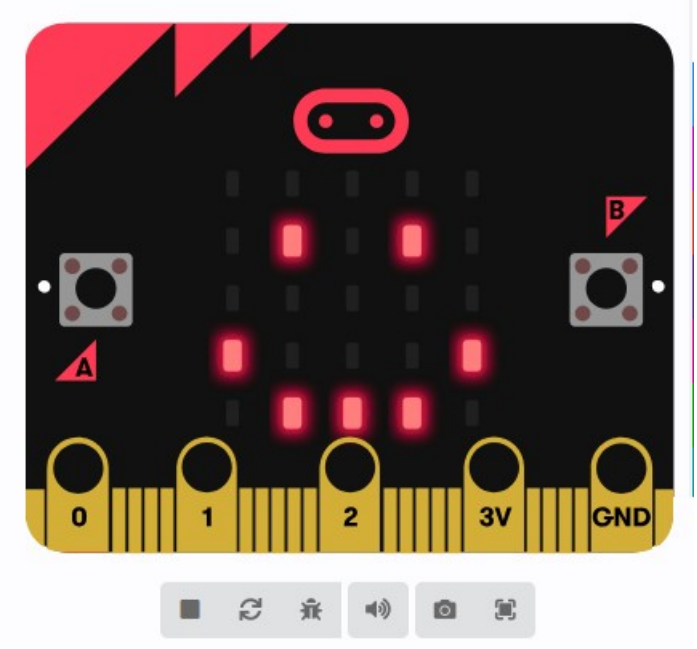
En primer lugar, vamos a mostrar una imagen predeterminada al iniciar el programa en la Microbit. Esta imagen permanecerá en la pantalla hasta que sea sustituida por otra mensaje. En el panel izquierdo elegimos la categoría “Básico” y el bloque “mostrar icono”.



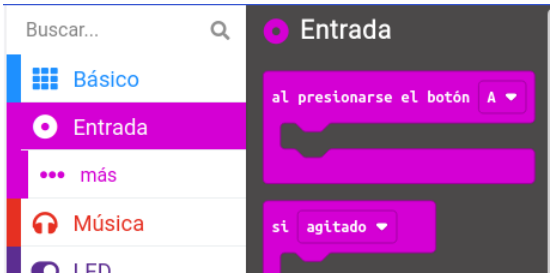
Arrastramos este bloque y lo colocamos dentro del bloque “al iniciar”. Si queremos cambiar la imagen desplegamos y elegimos otra imagen del conjunto de imágenes predefinidas.



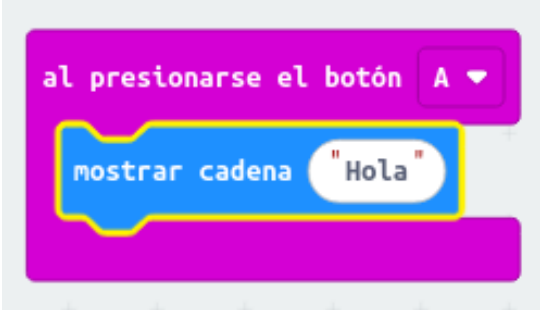
Si probamos a ejecutar el simulador, deberemos ver el icono al iniciar la ejecución.



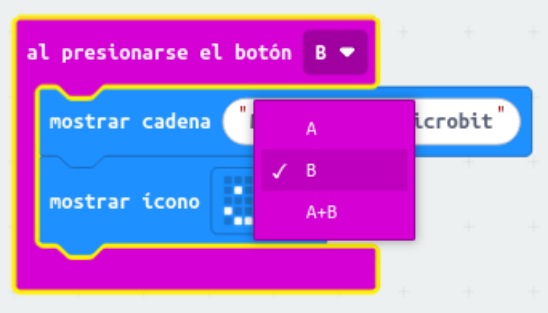
A continuación añadiremos un mensaje al pulsar cada uno de los botones. Para ello, arrastramos el bloque “al presionarse el botón <**A**> “ en la categoría “Entrada”.



Arrastramos el bloque al código del programa y añadimos dentro de el el bloque “mostrar cadena” de la categoría Básico. Cambiamos el texto inicial por “Hola”.



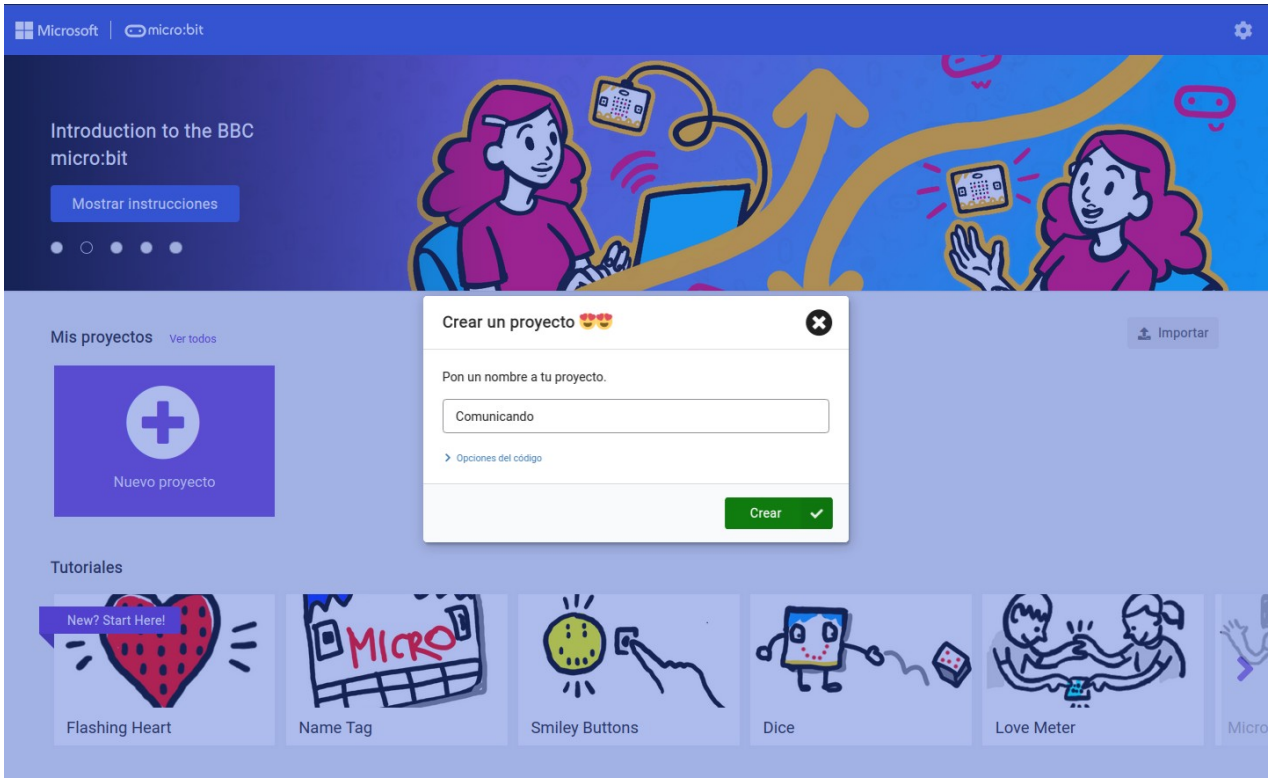
Repetiremos el proceso para añadir una funcionalidad al botón <**B**>, en este caso añadiremos un mensaje de texto y a continuación mostraremos otro icono. Recuerda cambiar la opción para que el bloque corresponda con la pulsación del botón <**B**>.



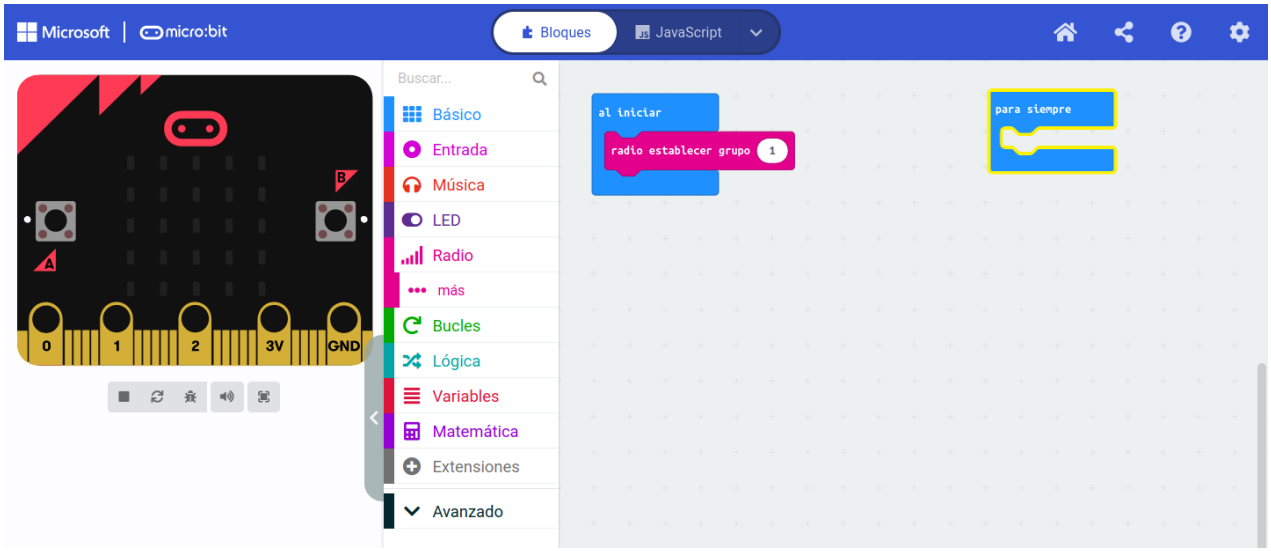
Puedes añadir más funcionalidad eligiendo también un bloque para la combinación de botones <**A+B**>.

3. Comunicación por radio

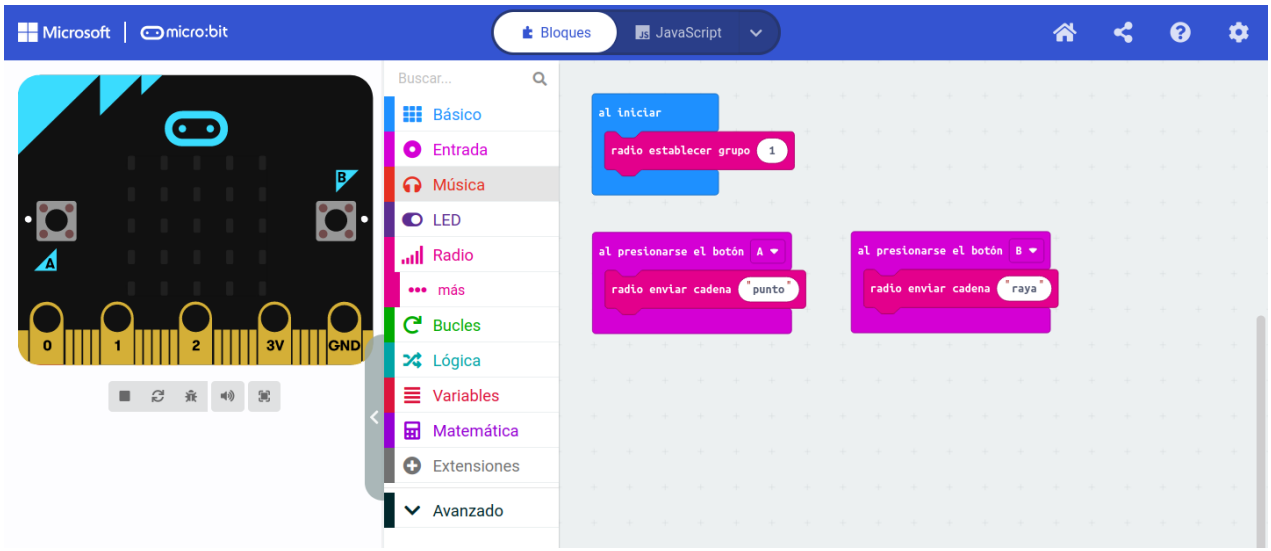
Creamos un nuevo proyecto para probar la comunicación entre Microbits a modo de señal Morse. Por ejemplo se le puede asignar el nombre "comunicando".



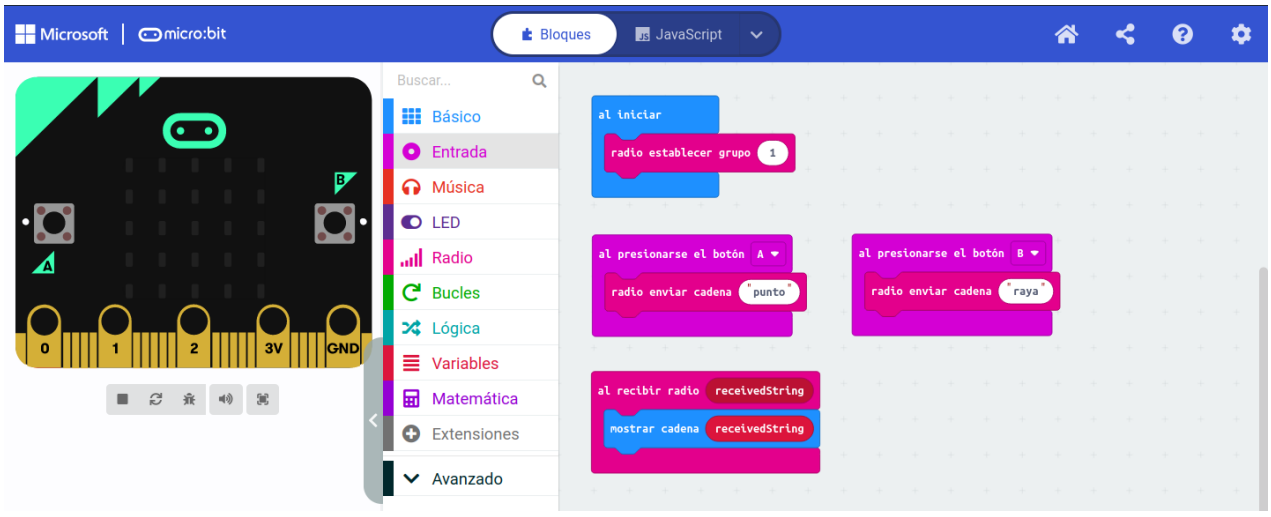
En esta proyecto, para la comunicación entre Microbits, cabe señalar que el código será común para simular ambas tarjetas. Es necesario al iniciar el código establecer el grupo de comunicaciones que se usará (en esta ocasión el grupo 1 por ejemplo).



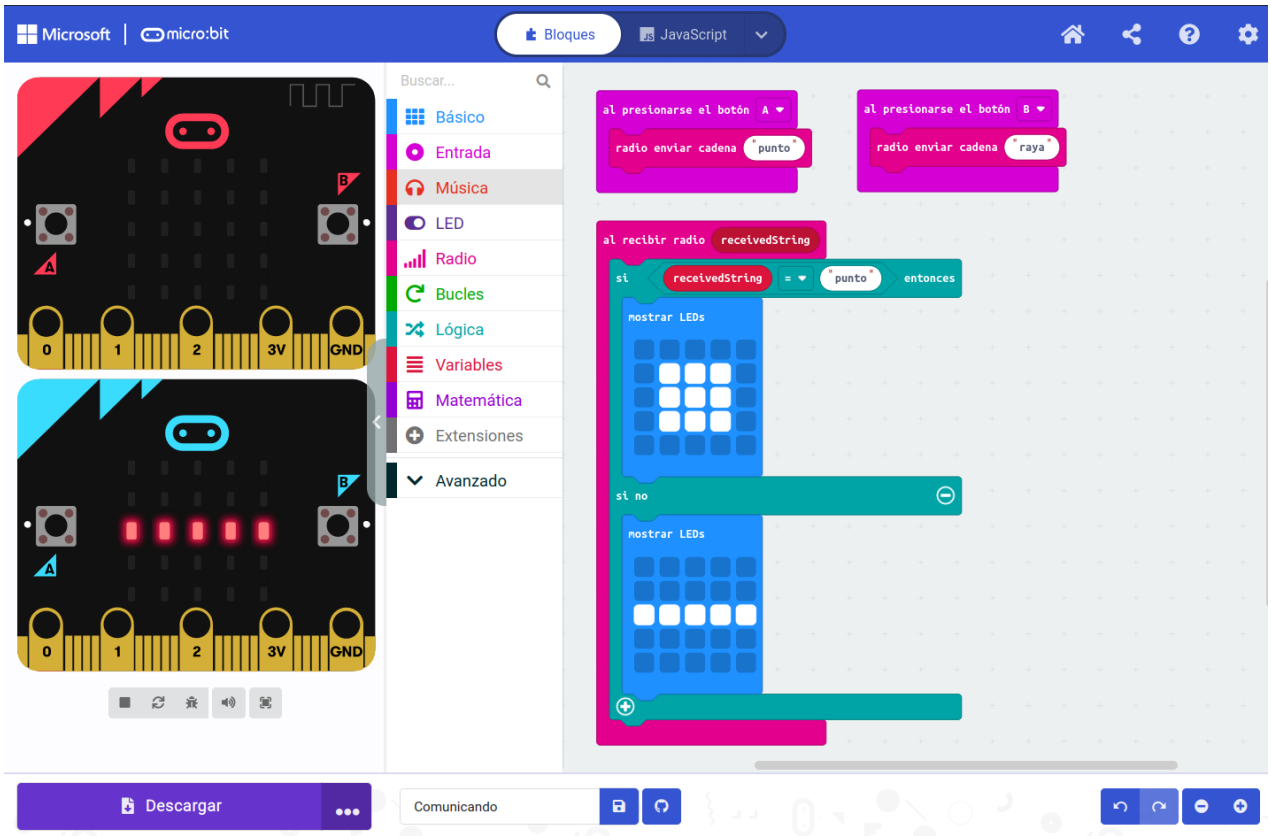
El código al pulsar <**A**> será enviar "punto" y al pulsar <**B**> será enviar "raya".



Para probar su funcionamiento, al recibir una señal de texto se muestra por pantalla. En esta ocasión, probamos su funcionamiento con el simulador.



Cambiamos el código, para que al recibir el texto "punto" se muestre un punto por pantalla y un sonido corto, mientras que si se recibe "raya" se muestra una línea por pantalla y un sonido más largo.

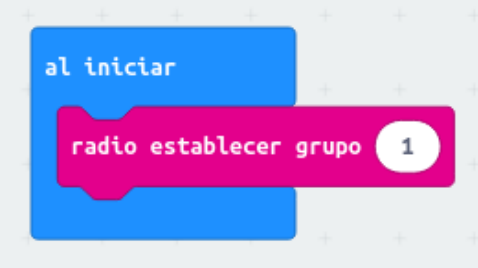


3.1. Transmisión de tipos de datos diferentes

A continuación, vamos a probar un programa similar de transmisión por radio, pero en esta ocasión se transmitirán tanto números como texto. También añadiremos un código de confirmación de envío.

Pasos a seguir

Lo primero que debemos hacer para utilizar las funciones de Radio, es definir al iniciar el canal de radio que se va a utilizar (entre 1 y 255). Se debe elegir el mismo número en ambas microbit.

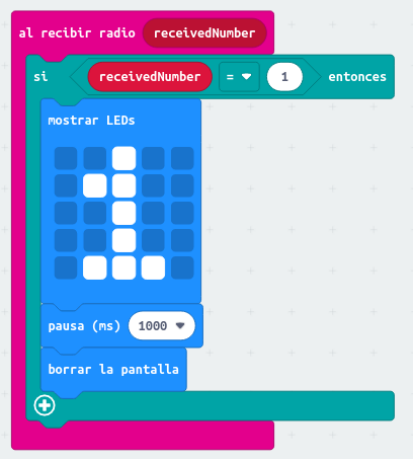


A continuación vamos a añadir un evento "al pulsar botón <**A**>", que envíe por radio el número 1, y muestre un icono ✔ en la pantalla durante 1 segundo.

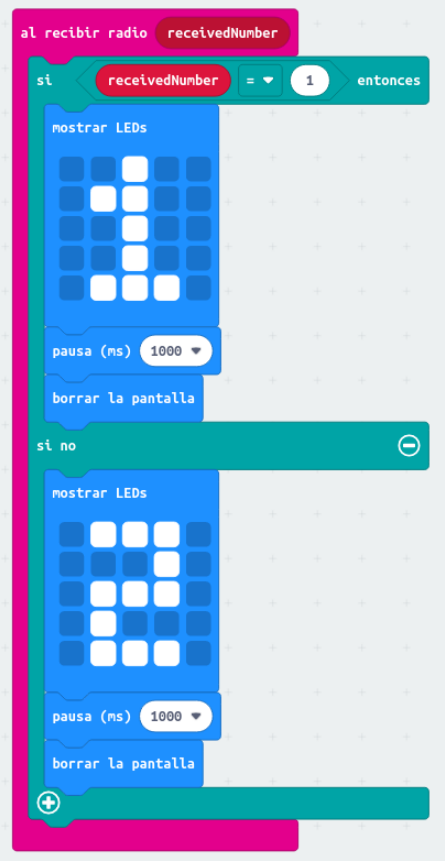
Hacemos lo mismo para el botón <**B**>, pero en esta ocasión enviamos el número 2.

Por último, al pulsar los dos botones <**A+B**> se envía el mensaje de texto "hola".

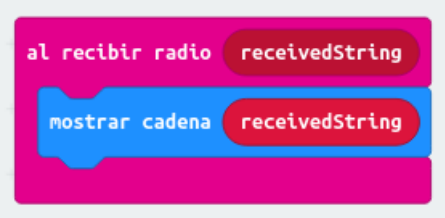
Ahora vamos a crear el código que recibirá la otra microbit. En este caso lo haremos en el mismo programa, para poder probar el funcionamiento en el simulador. Comenzamos creando el evento "al recibir radio" con un valor numérico. Dentro añadimos una condición "si el número = 1" y mostramos el valor 1 por pantalla durante 1 segundo.



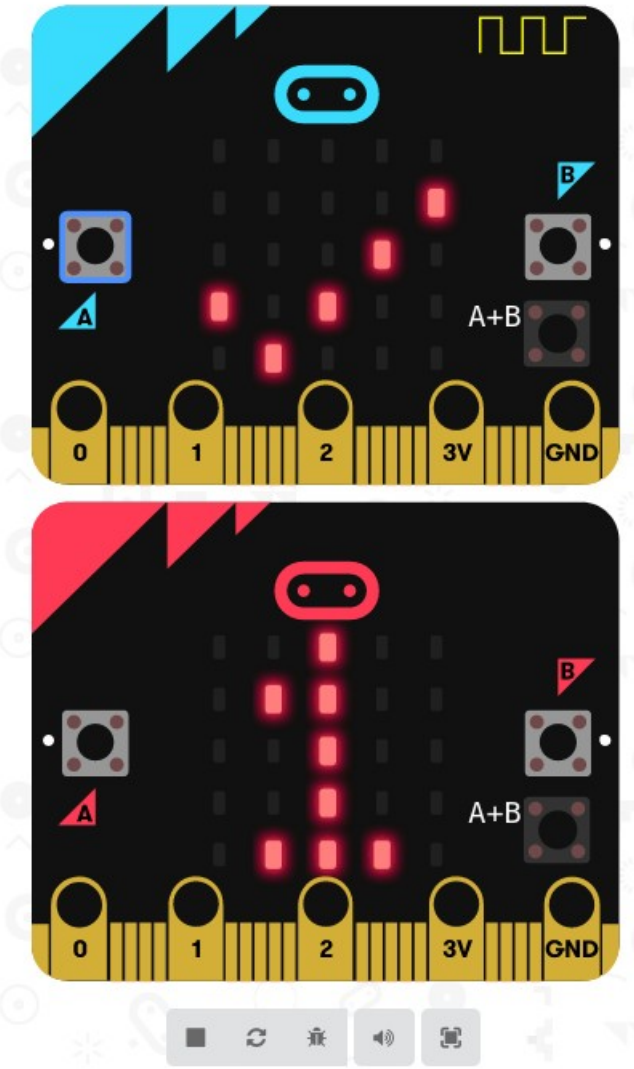
Pulsamos en el icono "+" para añadir la opción "si no". En este caso mostramos por pantalla un "2" durante 1 segundo.



Por último añadimos el evento "al recibir radio" un valor de cadena de texto, y lo mostramos por pantalla.

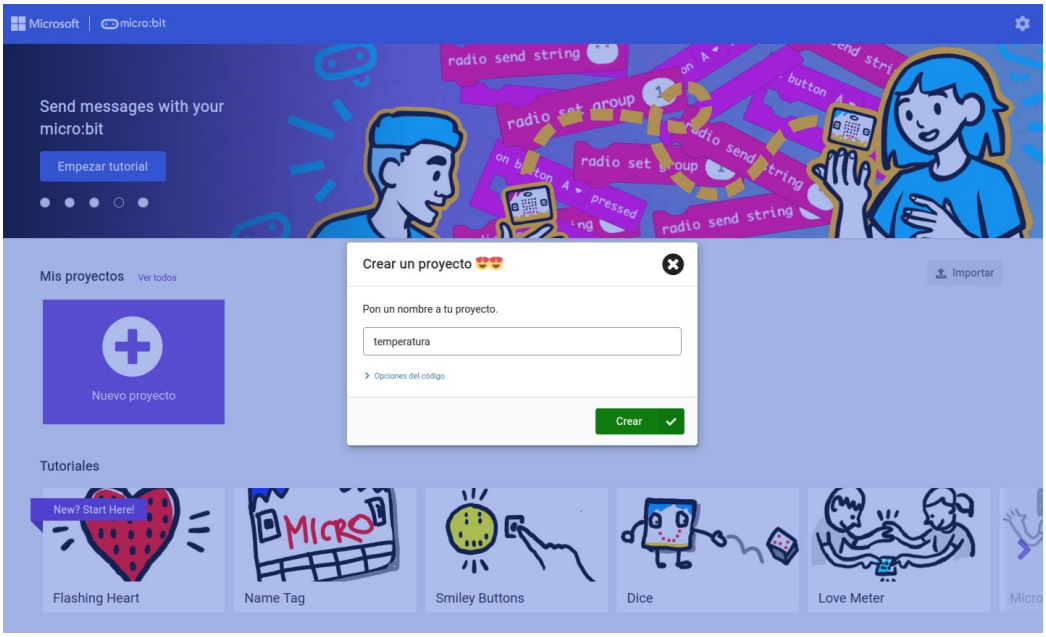


Al probar la microbit, en esta ocasión el programa detecta que se ejecuta código de comunicación por radio, y aparece una segunda microbit con la que probar como funciona la comunicación.

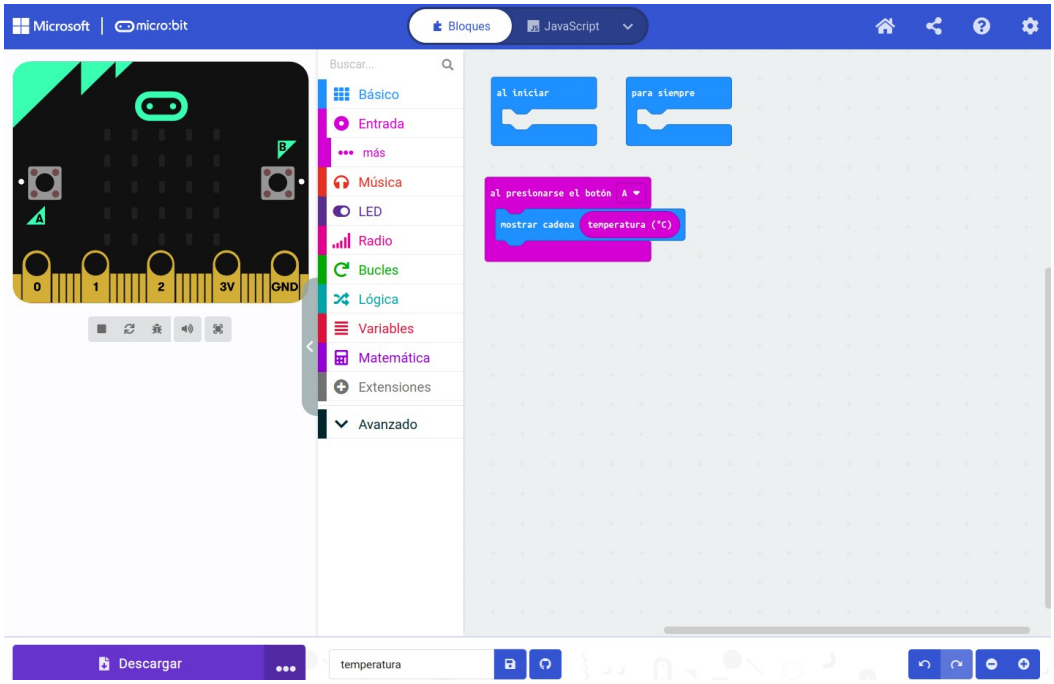


4. Sensor de temperatura

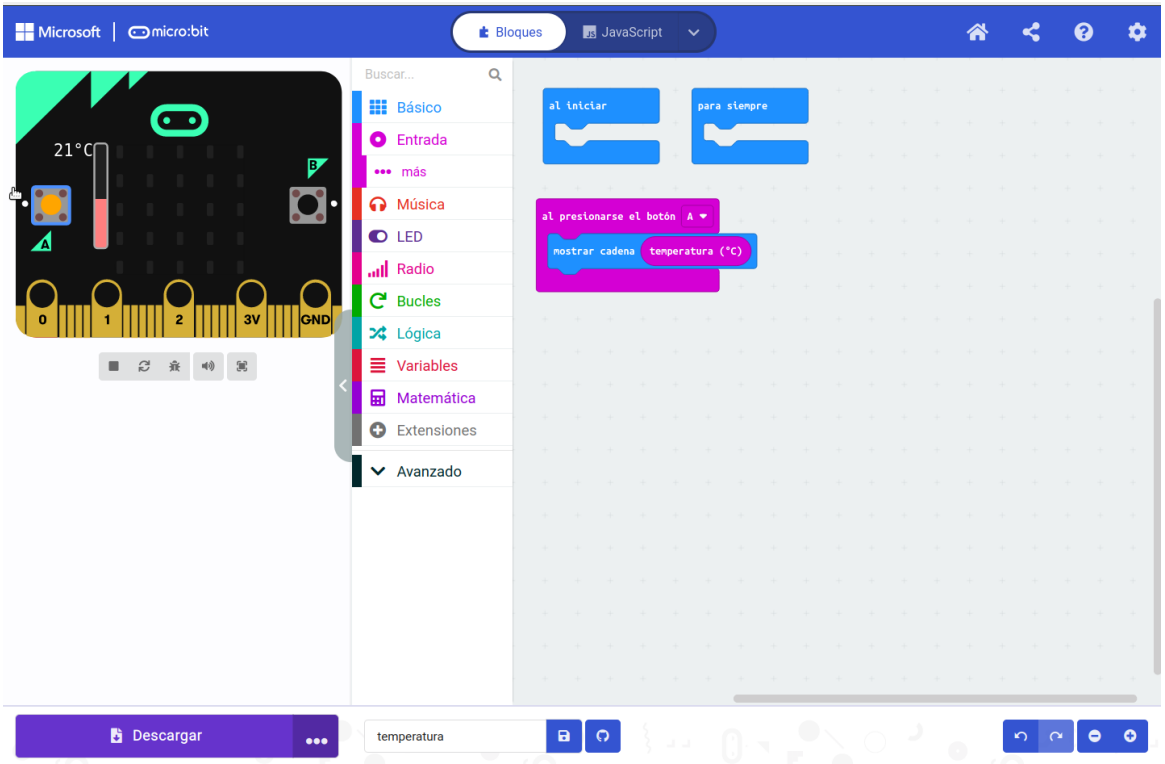
Creamos un proyecto nuevo, vamos a utilizar el sensor de temperatura y mostrarlo por pantalla.



Creamos un primer programa para que al presionar el botón <**A**> (bloques Entrada), se muestre por pantalla (bloques Básico) el valor de la temperatura (bloques Entrada).



Por último pulsamos el botón de ejecución del simulador para probar nuestro programa.

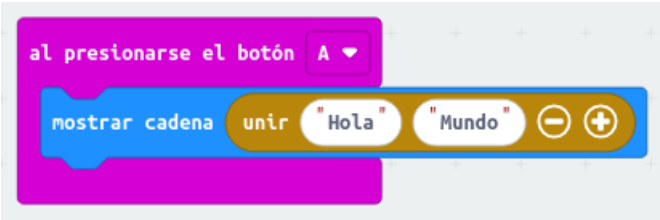


4.1. Conversión entre unidades de temperatura

Al programa anterior le vamos a añadir la funcionalidad de convertir el valor de temperatura en grados Celsius a grados Farenheit y grados Kelvin, aprovechando la funcionalidad del botón <**B**> y el botón <**A+B**>

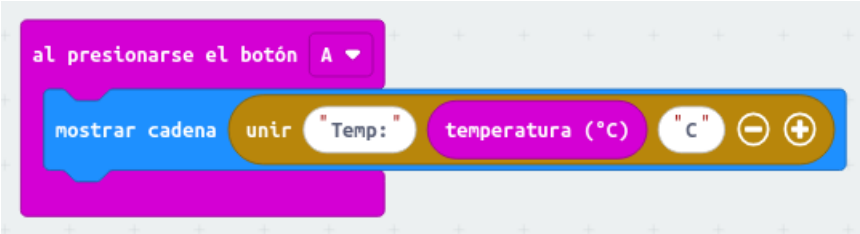
Pasos a seguir

En primer lugar, vamos a mostrar un texto junto con la temperatura actual en grados centígrados. Para ello, vamos a utilizar un bloque de unir texto para mostrar 3 campos de texto. Se añade un nuevo campo pulsando <**+**>.



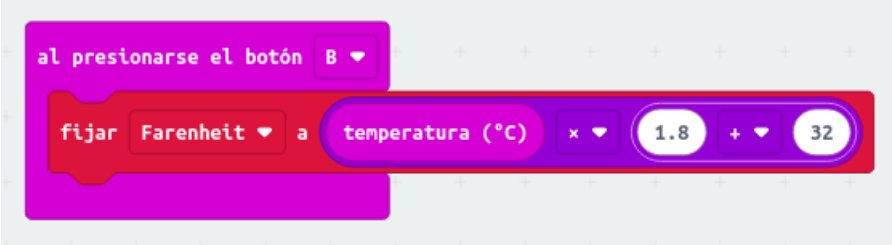
Cambiamos el primer campo de texto por "Temperatura", el segundo mostramos el valor obtenido con el sensor de temperatura de la Microbit que ya teníamos (también puedes encontrar este valor en el apartado "Entrada").

Por último añadimos el carácter "C" para indicar que son grados centígrados (el carácter º no se puede mostrar por led).

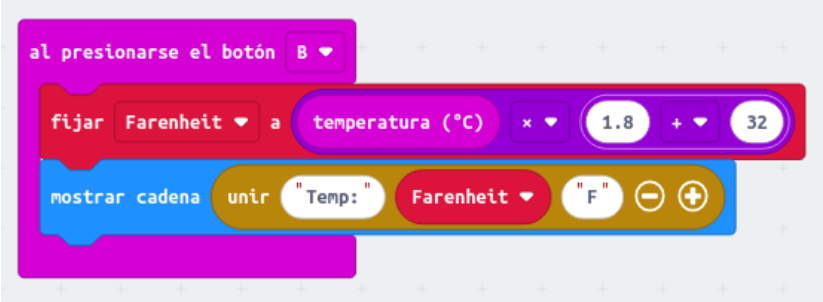


A continuación vamos a convertir los grados Centígrados a Farenheit al pulsar el botón <**B**>. Lo primero que hacemos es crear el evento y crear una variable llamada "farenheit".

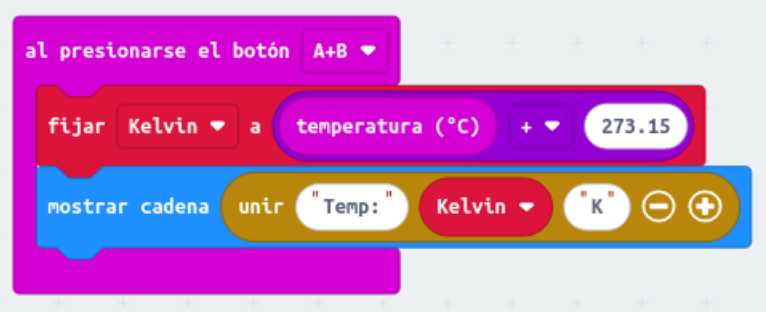
Para convertir de grados Centígrados a Farenheit hay que realizar la siguiente operación F = (C * 1.8) + 32 . Usamos los bloques de suma y multiplicación del grupo Matemáticas y el valor de entrada del sensor de temperatura.



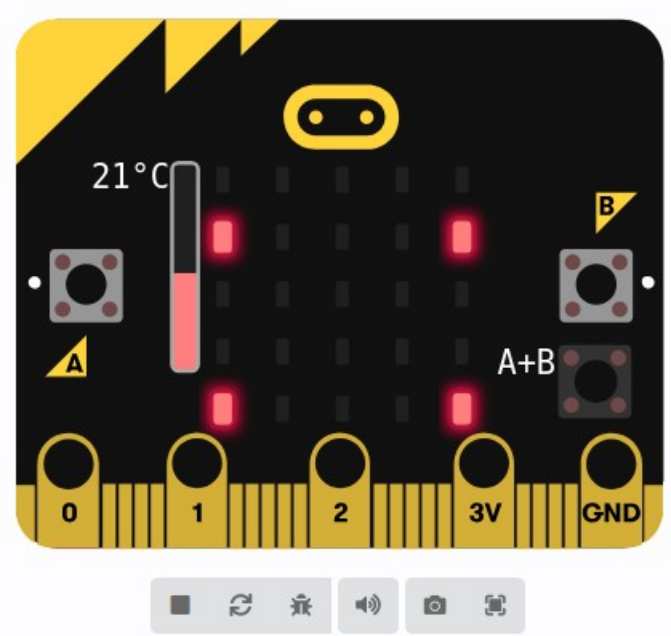
Finalmente mostramos el valor de la variable de forma similar a lo realizado con los grados centígrados, indicando con una F que la temperatura está expresada en grados Farenheit.



Para la combinación de botones <**A+B**>, realizaremos la misma operación pero en esta ocasión haremos la conversión a grados Kelvin, usando la fórmula : K = C + 273,15 . El resultado se muestra por pantalla indicando con K la unidad de temperatura está en grados Kelvin.



Al probar el código, podemos ver como aparece en el simulador un nivel para variar el valor de la temperatura que leerá el sensor.



Explicación detallada de esta actividad en el siguiente **vídeo.**

4.1. Diseñar una estación meteorológica

Se va a crear una modificación de los ejercicios anteriores, creando una estación meteorológica interior/exterior. Para ello, se va a utilizar la comunicación por radio, de forma que al pulsar el botón A se muestre el valor de la temperatura del microbit, mientras que si se pulsa el botón B se muestre el valor de la temperatura del otro microbit (microbit satélite).

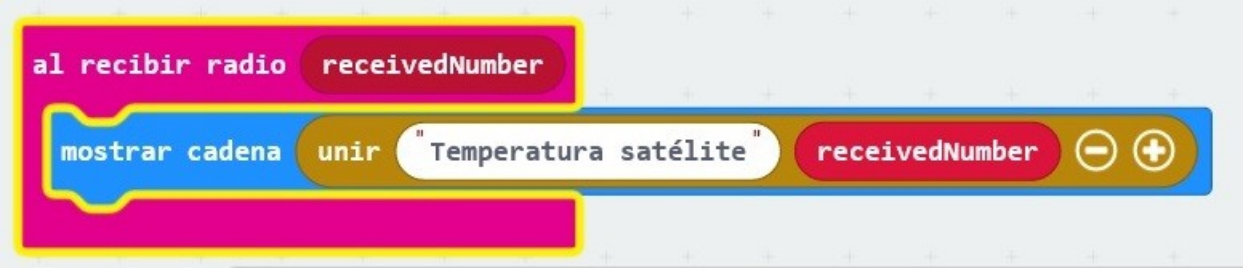
Pasos a seguir

En primer lugar, establece un canal de radio, será común para la base como para el satélite.

A continuación, si en la microbit base se pulsa el botón <**A**> se muestra por pantalla la temperatura de la microbit base.

Si en la microbit base se pulsa el botón <**B**> se envía un mensaje (texto) a la microbit satélite, que al recibir texto, envía a la base el número de su temperatura.

Al recibir la base el valor numérico de la microbit satélite, se muestra por la pantalla.



5. Sensores de movimiento

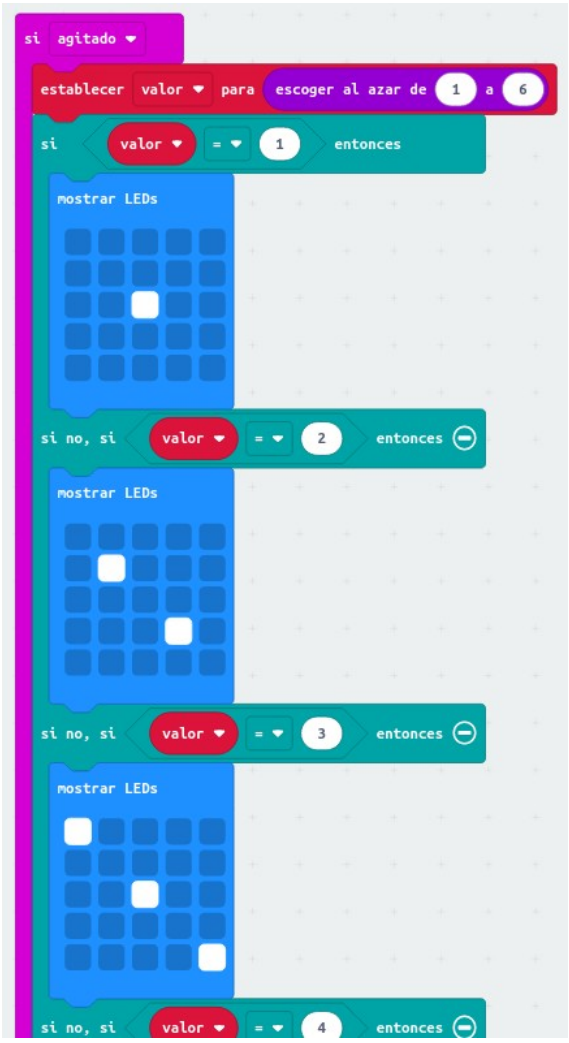
En este ejemplo vamos a simular el lanzamiento de dos dados, mostrando el número obtenido. En este ejercicio vamos a probar el sensor de movimiento (agitación).

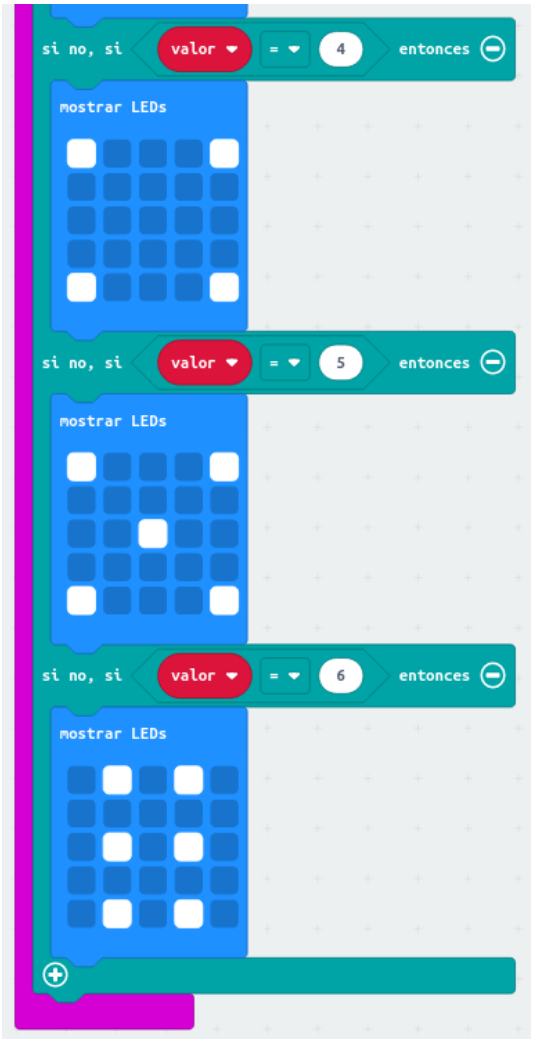
En primer lugar, creamos un nuevo proyecto llamado "Lanza los dados".

Al iniciar el programa, creamos e inicializamos las variables "tirada", "valor" y "suma".

Como vamos a utilizar el sensor de agitación, en la categoría “Entrada” añadimos al código el evento “Al agitar la microbit”

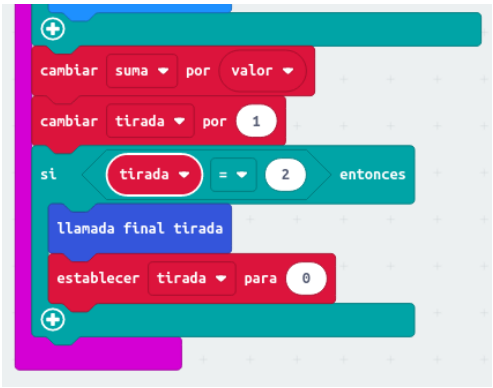
A continuación, elegimos un número al azar entre 1 y 6 y lo asignamos a la variable "valor". Según el valor obtenido mostramos por pantalla el símbolo similar al de un dado.





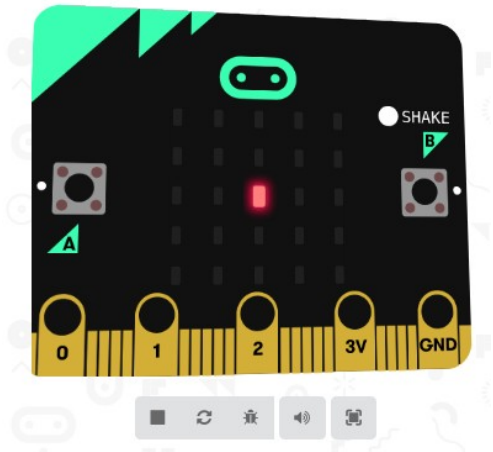
Modificamos el código anterior y añadimos a la variable suma el valor obtenido.

También incrementamos el número de tirada. Si es la segunda tirada, llamamos a la función final de tirada y reinicializamos el número de tirada a 0.



La función final de tirada, muestra 2 veces la suma obtenida y reinicializa las variables, para comenzar una nueva ronda de tiradas.

Al probar la microbit, tenemos el botón de agitar "shake" para simular el agitado de la placa.



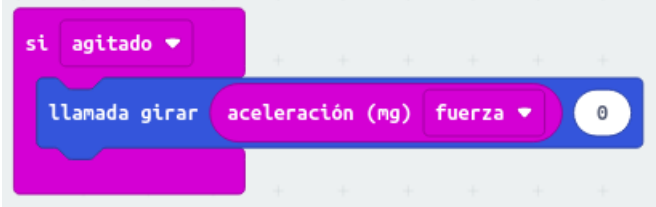
5.1. Giroscopio

Además del sensor de agitación, Microbit también tiene un sensor de giro (giroscopio). Para probarlo, vamos a crear un nuevo programa que simule el funcionamiento de una ruleta de la fortuna. Para ello, crea un nuevo proyecto llamado "Ruleta".

Pasos a seguir

En primer lugar, vamos a crear una función llamada "girar" a la que le pasaremos dos valores numéricos: fuerza y giro.

Añadimos el evento “si agitado”. En caso que se ejecute el evento, comenzamos a girar la ruleta llamando a "girar" con el valor del acelerómetro como fuerza y el valor inicial de giro a 0.



En cada momento vamos a mostrar la flecha según el valor de la variable giro. Hay que tener en cuenta la flecha se muestra para 8 posiciones: 0 norte, 1 noreste, 2 este.... Cuando el valor de giro sea mayor a 8 comenzamos la cuenta de nuevo, es decir si giro vale 8 es como si fuese 0 y deberá mostrar la flecha "norte".

Esto lo solucionamos utilizando el bloque "resto", el número de la variable giro lo dividimos entre 8, y nos quedamos con el resto (números del 0 al 7).

A continuación volvemos a llamar a la función girar, haciendo en esta ocasión que el valor de fuerza sea menor, y el valor de giro mayor. Esto es lo que llamamos recursividad: llamamos a una función desde la misma función, creando un bucle de llamadas.

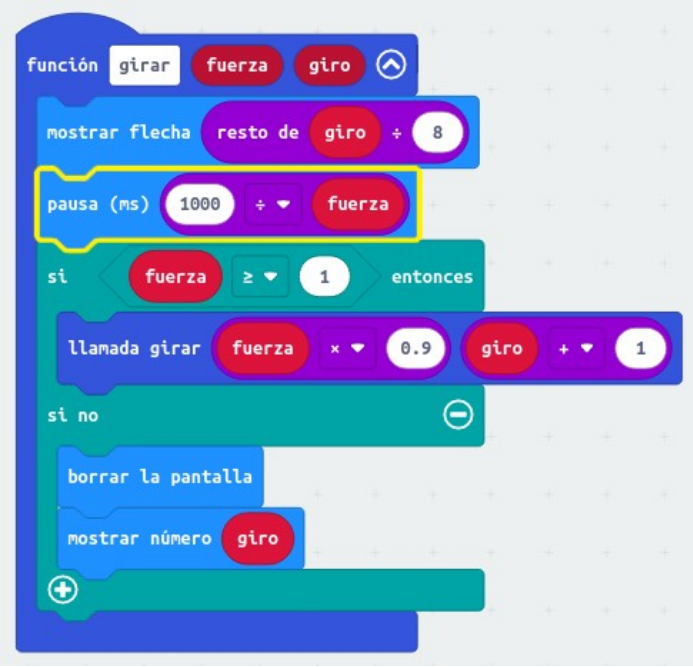
El valor de fuerza lo multiplicamos por 0,9 lo que hará que cada vez sea un valor más pequeño. El valor de giro se incrementa de 1 en 1.

Es muy importante que en las llamadas recursivas controlemos que en algún momento terminen, de lo contrario entraremos en un bucle infinito y el programa nunca funcionará.

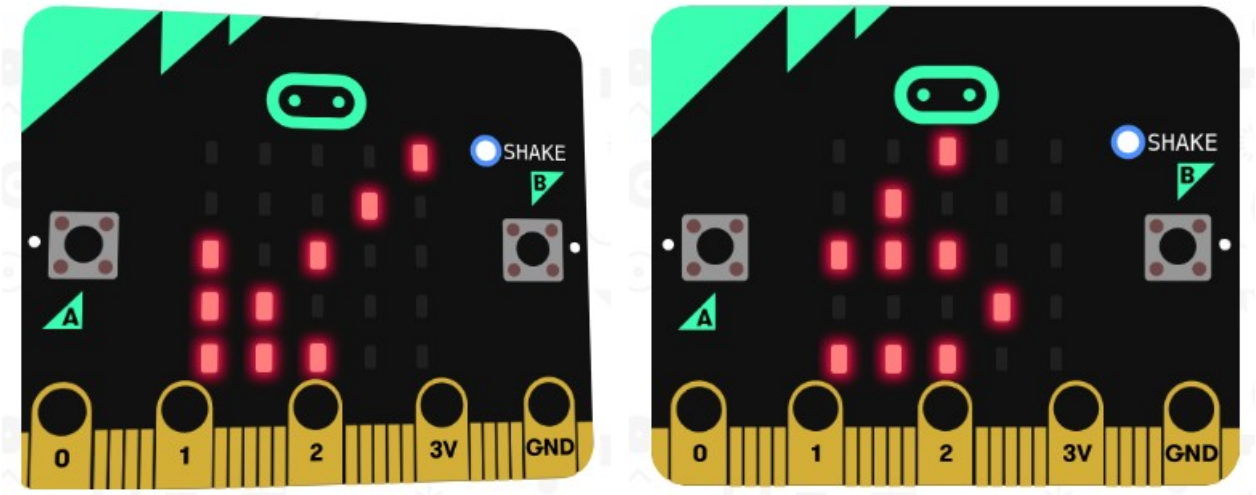
En esta ocasión nos aseguramos que se repita la llamada mientras el valor de fuerza sea mayor que 1, cuando sea menor a 1, dejaremos de llamar recursivamente a la función "girar" y el programa terminará.

Vamos a completar la condición "Si" con la opción "Si no" (cuando fuerza sea menor a 1). Mostraremos el número total de incrementos de giro, para ver que valor hemos obtenido. Para que se vea mejor, limpiamos previamente la flecha de la pantalla.

Por último vamos a hacer que la flecha gire de forma más lenta a medida que la fuerza disminuye. Para esto hacemos una pausa de un número divido entre el valor de fuerza, por ejemplo "1000 / fuerza". A medida que el valor de fuerza sea menor, la pausa será mayor.



Ya podemos probar el código de nuestra ruleta de la fortuna microbit.



5.2. Juego de dados por parejas

Vamos a realizar una versión modificada del juego de dados, para jugar por parejas aprovechando la comunicación por radio. Para realizar esta versión ampliada del juego:

Pasos a seguir

Define dos variables "suma_mia" y "suma_rival" a 0.

Al realizar el final de tirada (dos tiradas por ejemplo), se almacena el resultado en "suma_mia" y envía el resultado al rival.

El rival almacena el número obtenido en la variable "suma_rival".

Tanto al finalizar la tirada como al recibir un número del rival se llama a la función "comparar resultados".

La función comparar resultados solo funciona si suma_mia y suma_rival es mayor de 0.

Si suma_mia es mayor se muestra ✔ y si suma_rival es mayor se muestra ✖.

En caso contrario (empate) se muestra —.

5.3. Juego de dados por equipos

Si decides realizar la versión del juego por equipos:

Se define una variable jugadores con el número de personas por equipo. Se define una variable juego a 0

Se define la variable nuestro_canal con un valor común para todos los jugadores de un equipo. Se define la variable canal_envío con el canal por el que envía la suma un equipo al otro equipo. Se define otra variable canal_recepcion con el canal por el que se espera recibir la información del equipo rival.

Los jugadores de cada equipo comparten su propio canal.

Cada miembro del equipo hace su tirada y se calcula el total del equipo.

Si la variable juego es 0 y todos los jugadores han realizado su tirada, un miembro del equipo puede enviar la suma al equipo rival a través del canal canal_envío. El valor de la variable juego para ese equipo pasa a 1.

La suma recibida por el equipo contrario se recibe por canal_recepcion y almacenada en una variable.

Cuando ambas cantidades hayan sido enviadas y recibidas por el equipo rival se hace la comparación, se indica el equipo ganador y se comienza de nuevo, pasando la variable juego a 0.

6. Otras actividades

Existen multitud de proyectos, ejercicios y ejemplos para realizar con tu placa Microbit en la **página oficial Microbit.org** y en la página de proyectos de **Microsoft Makecode Microbit**.

6.1. Código morse

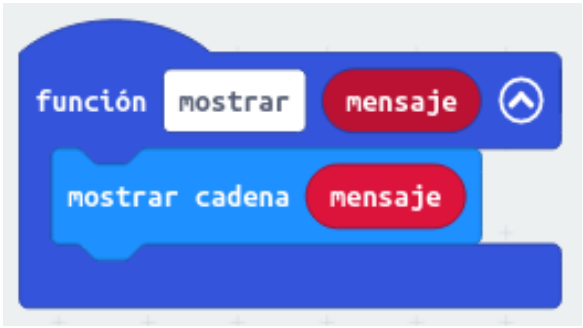
En este ejercicio vamos a crear un sistema de emisión y recepción morse.

A ·–	J ·–––	S ···	1 ·————
B –···	K –·–	T –	2 ··–––
C –·–·	L ·–··	U ··–	3 ···––
D –··	M ––	V ···–	4 ····–
E ·	N –·	W ·––	5 ·····
F ··–·	O –––	X –··–	6 –····
G ––·	P ·––·	Y –·––	7 ––···
H ····	Q ––·–	Z ––··	8 –––··
I ··	R ·–·		9 ————·
			0 —————

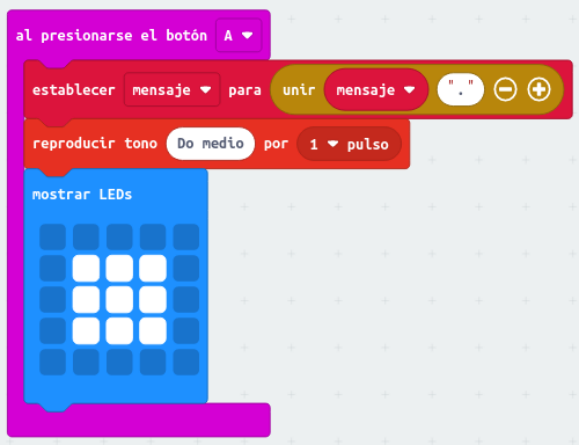
En primer lugar, creamos un nuevo proyecto llamado "Morse".

Al iniciar el programa se asigna la emisora de radio 1 y se crea una variable "mensaje" con valor inicial " " (texto vacío). También vamos a enviar un mensaje de texto vacío (así el simulador comenzará funcionando por pareja).

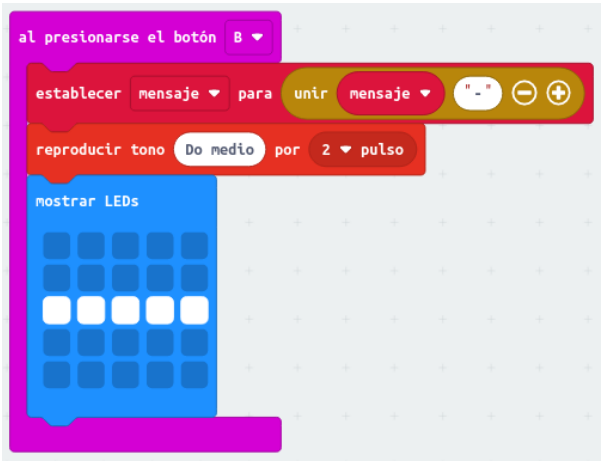
A continuación se crea una función llamada "mostrar", donde se pasa un valor de texto llamado "mensaje". El mensaje se muestra por pantalla.



Al pulsar el botón <**A**>, se une al mensaje de texto el mensaje y el carácter ".", suena un pitido (cualquier nota, por ejemplo "do medio") durante 1/2 pulso y muestra un círculo (o parecido) por pantalla.



Al pulsar el botón <**B**>, se une al mensaje de texto el mensaje y el carácter "-", suena un pitido (cualquier nota, por ejemplo "do medio") durante 1 pulso y muestra una línea (o parecido) por pantalla.



Al pulsar los botones <**A+B**>, se muestra el mensaje montado, se envía por radio (es un texto) y se borra el valor de la variable "mensaje".

Crea una función llamada "sonar" que recibe un valor de texto.

Dentro de la función "sonar" crea un bucle para index de 0 a longitud de mensaje-1.

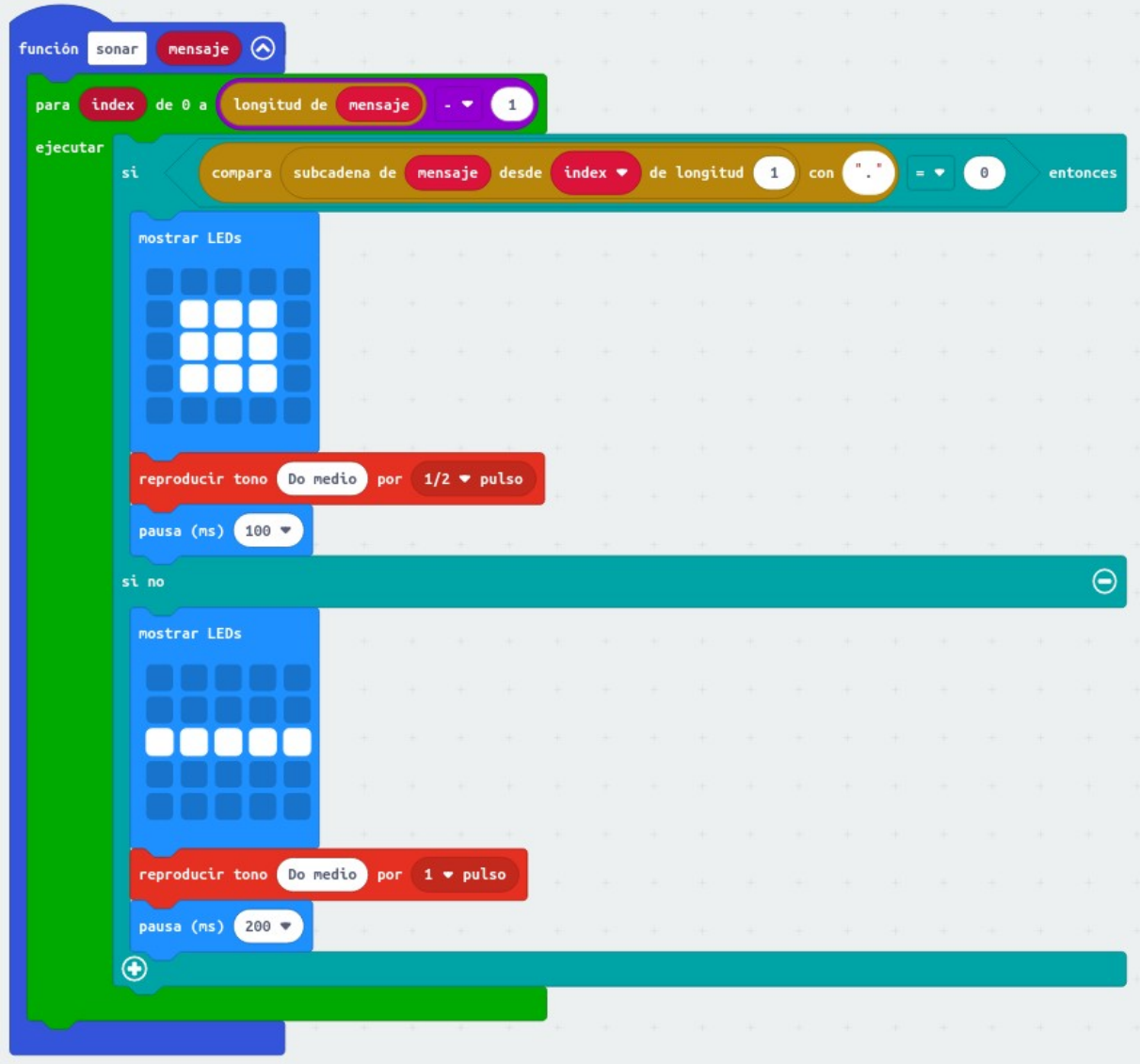
Si compara subcadena de "mensaje" desde "index" de longitud 1 con "." es igual a 0.

Muestra un punto grande y suena un pitido (cualquier nota, por ejemplo "do medio") durante 1/2 pulso.

Espera 100 ms si no, muestra una línea grande.

Suena un pitido (cualquier nota, por ejemplo "do medio") durante 1 pulso.

Espera 200 ms.



Al recibir el mensaje (recuerda que es texto), muestra el mensaje por pantalla y haz que suene el mensaje.

Crea una función llamada "iluminar" mensaje (muy parecida a la de sonar) pero en esta ocasión no hay sonido (obviamente) y ...

si es un punto se muestra completamente la pantalla iluminada durante 500 ms,

o si no (es una línea) se muestra completamente la pantalla iluminada durante 1000 ms.

En ambos casos pasado el tiempo de espera se borra la pantalla y se espera 100 ms.

La función "iluminar" mensaje se ejecuta al pulsar el botón A+B (ojo, no al recibir).

6.2. Competición de reflejos

En esta actividad queremos hacer una versión ampliada del juego “prueba tus reflejos”. Puedes ver el funcionamiento del juego y su desarrollo en el siguiente **vídeo**.

En esta ocasión, desarrollaremos la versión del juego para la competición por parejas.

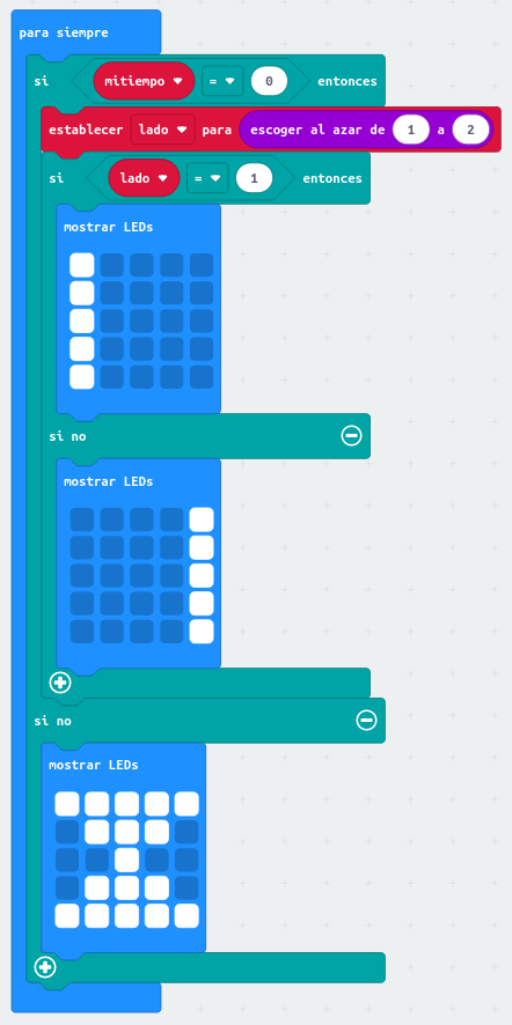
Al iniciar el programa ocurre lo siguiente:

Se define el canal de radio 1 y se envía por mensaje el número 0.

Se crean las variables "mitiempo", "sutiempo", "mispuntos", "suspuntos", "lado" y se inicializan a 0.

Se crea la variable "tiempo" y se inicializa con el tiempo de ejecución en milisegundos (recuerda que es un valor de entrada).

En todo momento se ejecuta el siguiente código:

<table>
<tr>
<td> 
</td>
<td>Si el valor de "mitiempo" aún está vacío (aún no he acertado).

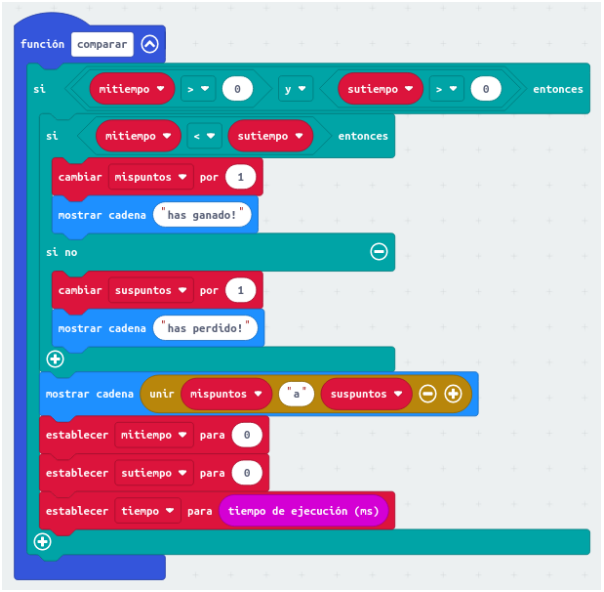
Se elige un número al azar (1 o 2) y se almacena en "lado".

Si el número es 1
se muestra por pantalla una barra en el lado izquierdo,
si no (el número es 2).

Se muestra una barra en el lado derecho,

si no, si el valor de "mitiempo" es mayor que 0 (porque ya he acertado)

se muestra por pantalla un reloj a modo de espera al rival .</td>
</tr>
</table>



Creamos una función llamada comparar a la que no le pasamos valores.

La función solo funciona si la variable "mitiempo" es mayor a 0 y la variable "sutiempo" también es mayor a 0.

Si "mitiempo" es menor que "sutiempo".

Se suma 1 a la variable "mispuntos".

Se muestra el mensaje "has ganado!"

si no, si "mitiempo" es mayor que "sutiempo"

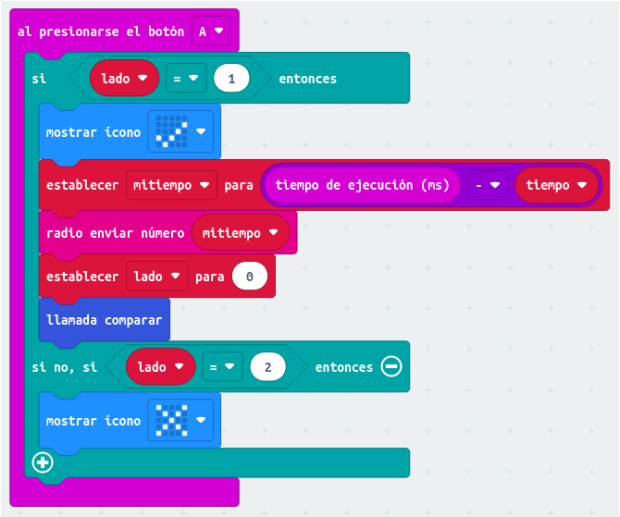
se suma 1 a la variable "suspuntos".

Se muestra el mensaje "has perdido!".

En ambos casos, se muestra un mensaje uniendo "mispuntos" " a " "suspuntos".

Se inicializan las variables "mitiempo" y "sutiempo" a 0.

Se inicializa la variable "tiempo" al tiempo de ejecución en milisegundos.



Cuando se pulsa el botón <**A**> debe ocurrir lo siguiente:

Si el lado es 1, hemos acertado y entonces mostramos el icono ✔.

Guardamos en la variable "mitiempo" el tiempo actual en milisegundos menos el valor de la variable "tiempo".

Enviamos ese valor por radio.

La variable "lado" la ponemos con valor a 0 a la espera de comenzar de nuevo (y que mientras tanto no ocurra nada al pulsar el botón <**A**> o <**B**>).

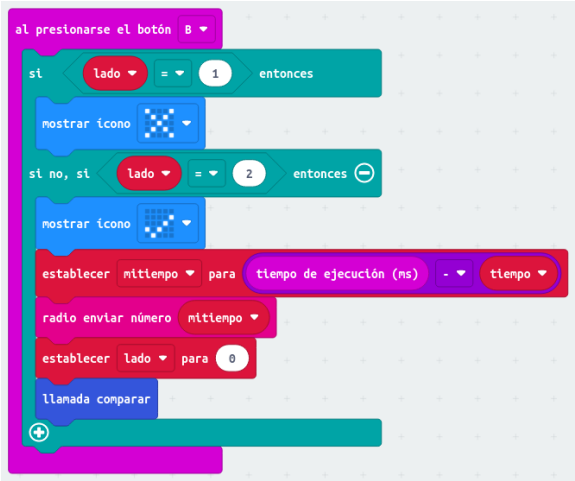
Llamamos a la función comparar para ver si hemos ganado o perdido.

Si el lado es 2, no hemos acertado y entonces

mostramos el icono ✖

en cualquier otro caso no ocurre nada.

Cuando se pulsa el botón <**B**> debe ocurrir lo mismo que al pulsar el botón <**A**>, pero con las condiciones al revés: si el lado es 1 no hemos acertado y si el lado es 2 hemos acertado.



Al recibir un número por radio

se guarda en la variable "sutiempo"

y se llama a la función comparar

6.3. Cálculo mental

En esta actividad vamos a hacer un juego de cálculo mental simulando a un famoso concurso televisivo de preguntas y respuestas.

El juego consiste en mostrar una operación matemática al azar y el jugador debe adivinar su resultado en el menor tiempo posible.

- Se jugará por parejas y ganará el que resuelva mentalmente la operación matemática en menor tiempo.
- Con cada pulsación del botón <**A**> se suma 10 a la respuesta, y con cada pulsación del botón <**B**> se suma 1. Al pulsar <**A+B**> se comprueba la respuesta, y si no es correcta volvemos a comenzar.
- Cuando ambos jugadores han acertado, se muestra un mensaje indicando quién es el ganador que ha resuelto la operación en menos tiempo.

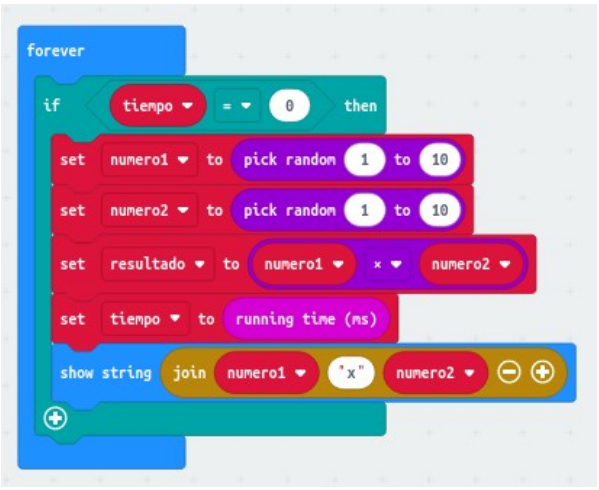
Pasos a seguir

Al iniciar el programa ocurre lo siguiente:

Se define el canal de radio 1 y se envía por mensaje el número 0.

Se crean las variables "mitiempo", "sutiempo", "numero1", "numero2", "resultado", "respuesta" y "tiempo" y se inicializan a 0.

En todo momento se ejecuta el siguiente código:



Si el valor de "tiempo" aún está vacío (aún no he acertado).

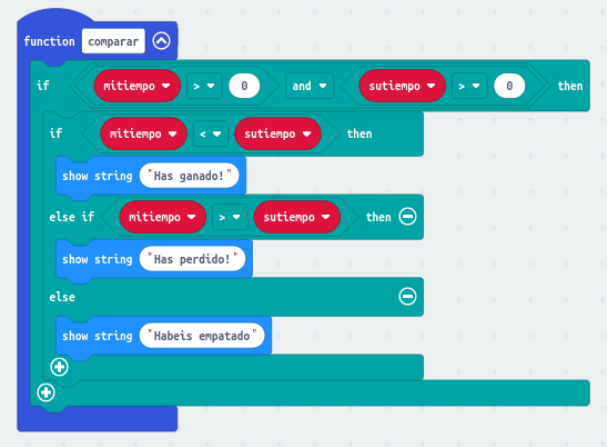
Se elige un número al azar de 1 a 10 y se almacena en "numero1".

Se elige de igual forma otro número al azar para "numero2".

Se calcula "numero1" x "numero2" y se almacena en la variable "resultado".

Se almacena en la variable "tiempo" el valor actual de tiempo de ejecución en milisegundos.

Se muestra por pantalla el texto "numero1" "x" "numero2".



Creamos una función llamada comparar a la que no le pasamos valores.

La función solo funciona si la variable "mitiempo" es mayor a 0 y la variable "sutiempo" también es mayor a 0.

Si "mitiempo" es menor que "sutiempo".

Se muestra el mensaje "has ganado!".

Si no, si "mitiempo" es mayor que "sutiempo".

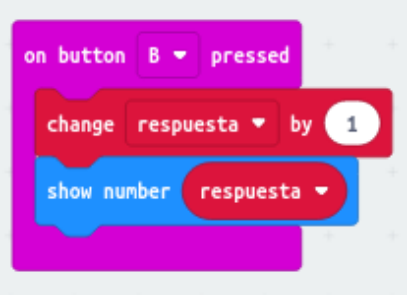
Se muestra el mensaje "has perdido!"

Si no se muestra el mensaje "habeis empatado!"

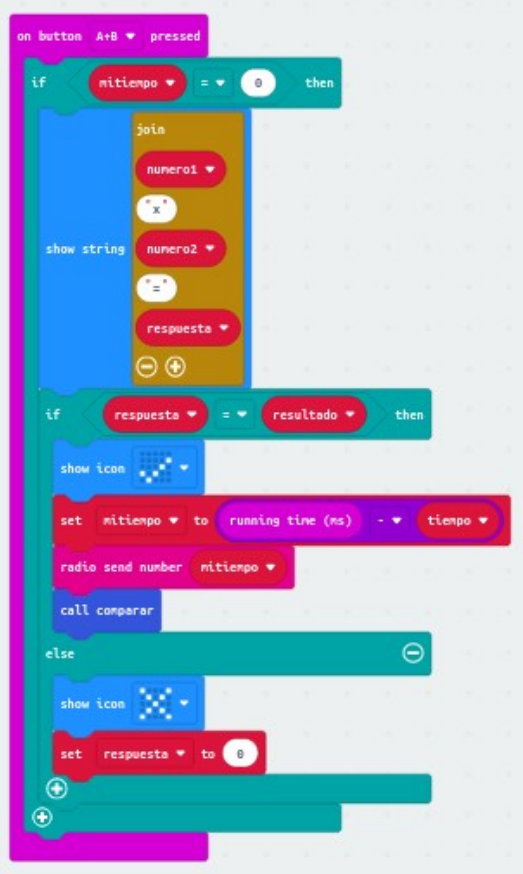
Cuando se pulsa el botón <**A**> debe ocurrir lo siguiente:

cambiamos el valor "respuesta" sumándole 10

y mostramos el valor de "respuesta" por pantalla.



Cuando se pulsa el botón <**B**> debe ocurrir lo mismo que al pulsar el botón <**A**>, pero añadiendo 1 a "respuesta".



Al presionar <**A+B**>

Solo funciona si el valor de mitiempo es 0

Se muestra un mensaje compuesto de "numero1" "x" "numero2" "=" "respuesta"

Si "respuesta" es correcta (es igual a "resultado")

Se muestra ✔

Se calcula "mitiempo" como la resta del tiempo de ejecución actual - "tiempo"

Se envía por radio "mitiempo"

Se llama a la función comparar

Si la respuesta no es correcta

Se muestra ✖

Se inicializa el valor de "respuesta" a 0.

Al recibir un número por radio

se guarda en la variable "sutiempo"

y se llama a la función comparar .

6.4. Llegar a 21

Vamos a hacer un juego similar al famoso juego de cartas Blackjack.

Descripción de la actividad

- El juego consiste en pedir números que pueden tener valor entre 1 y 11. Se piden cartas hasta que se decida parar o se supere el valor 21.
- Se jugará por parejas y ganará el que tenga más puntos sin pasarse de 21. En caso de pasarse de 21 automáticamente se pierde.
- Con cada pulsación del botón <**A**> se pide un nuevo número que se suma al valor de la partida que se juega.
- Con la pulsación del botón <**B**> se planta la partida y se envía la puntuación obtenida.

Pasos a seguir

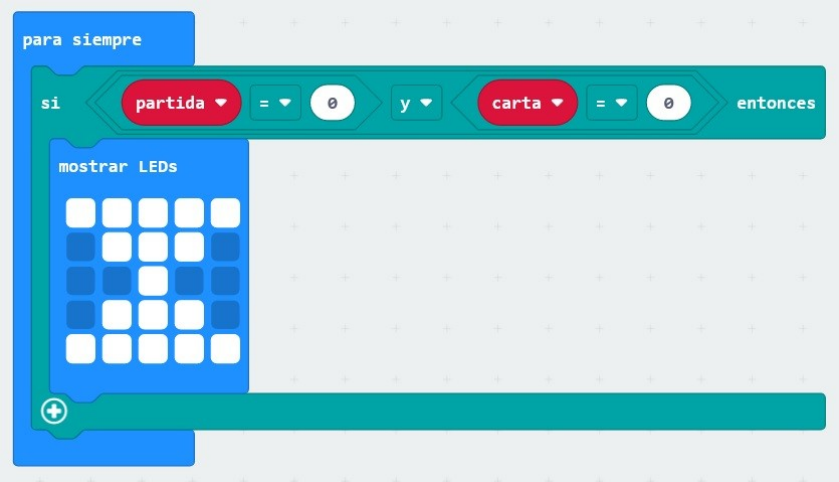
- Al iniciar el programa ocurre lo siguiente:

Se define el canal de radio 1 y se envía por mensaje el número 0.

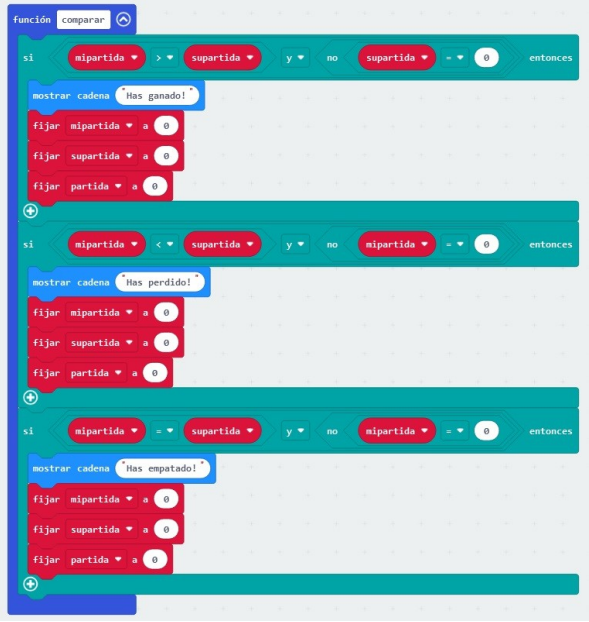
Se crean las variables "carta", "partida", "mipartida", "supartida" y se inicializan a 0.

- En todo momento se ejecuta el siguiente código.

  Si el valor de "carta" es 0 y el valor de "partida" es 0, aún no ha comenzado la partida y se muestra el símbolo de espera.



- Creamos una función llamada "comparar" a la que no le pasamos valores.



Si "mipartida" es mayor que "supartida" y el valor de "supartida" no es 0:

Se muestra el mensaje "has ganado!"

Vuelve a dar el valor a "mipartida", "supartida", "partida" a 0.

Si "mipartida" es menor que "supartida" y el valor de "mipartida" no es 0:

Se muestra el mensaje "has perdido!".

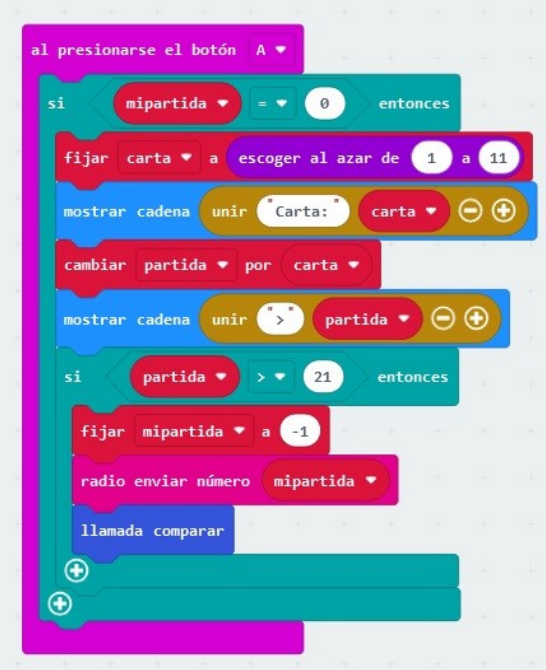
Vuelve a dar el valor a "mipartida", "supartida", "partida" a 0.

Si "mipartida" es igual que "supartida" y el valor de "mipartida" no es 0:

Se muestra el mensaje "habéis empatado!"

Vuelve a dar el valor a "mipartida", "supartida", "partida" a 0

- Cuando se pulsa el botón <**A**>, se va a pedir un nuevo número. Por ello, debe ocurrir lo siguiente:



Si el valor "mipartida" es 0 (aún seguimos jugando),

Dale a la variable "carta" un valor aleatorio entre 1 y 11.

Muestra el mensaje "Carta:" y el número almacenado en la variable "carta".

Suma a la variable "partida" el valor conseguido en "carta".

Muestra el mensaje ">" y el número almacenado en la variable "partida".

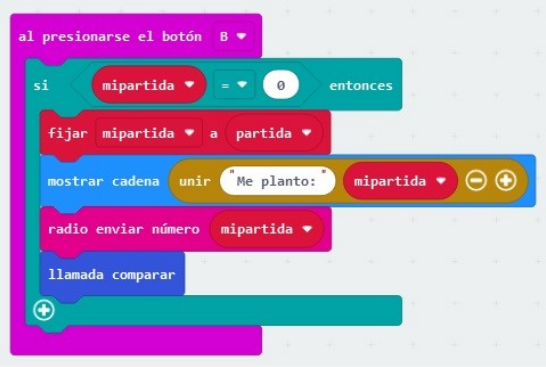
Si "partida" > 21 entonces:

Guarda en "mipartida" el valor -1.

Envía por radio el valor de "mipartida".

Llama a la función "comparar".

- Cuando se pulsa el botón <**B**> nos plantamos y enviamos al contrincante la puntuación alcanzada:



Si el valor "mipartida" es 0 (aún seguimos jugando)

Guarda en la variable "mipartida" el valor de "partida".

Muestra el mensaje "Me planto" y el valor de "mipartida".

Envía por radio el valor de "mipartida".

Llama a la función comparar.

- Al recibir un número por radio, se guarda en la variable "supartida" y se llama a la función comparar.

Programa financiado por el Ministerio de Educación, Formación Profesional y Deportes y el Mecanismo de Recuperación y Resiliencia