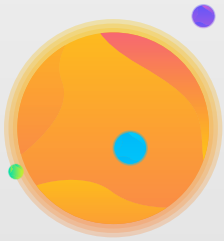
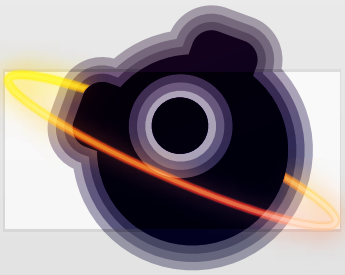
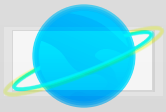






ROBOTEACH MASTER

CUARTO MÓDULO


PRIMERA
CLASE

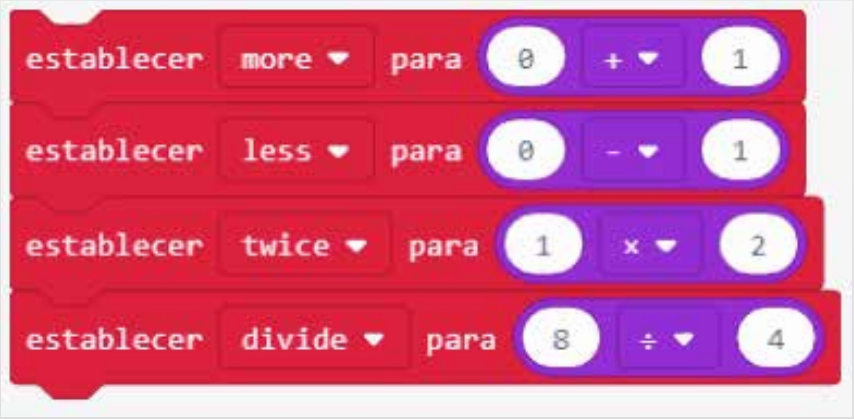
Operaciones Bàsicas

Cuando hablamos de operaciones nos referimos a ejecuciones o maniobras metódicas y sistemáticas sobre cuerpos, números, datos, etcétera, para lograr un determinado fin.

Las operaciones básicas de la matemática son cuatro la suma, la resta, la multiplicación y la división donde se obtiene un nuevo elemento a partir de dos elementos dados. Las operaciones básicas de las matemáticas nos sirven mucho para la vida cotidiana, y también nos sirve para el colegio para los ejercicios que nos pongan.

Ejemplo:

En micro:bit las operaciones se muestran de la siguiente manera:



Operadores aritméticos

Los siguientes operadores aritméticos trabajan con números y devuelven un Número:

- adición: 1 + 3
- sustracción: 1 - 3
- multiplicación: 3 * 2
- División entera: 7 / 3
- módulo está disponible a través de la biblioteca de matemáticas

Valores numéricos: 0, 1, 2, 6.7, 10.083…

Sólo números por sí mismos. A veces estos se llaman literales numéricos.

Enteros: números enteros

Punto flotante: números con una parte fraccionaria.

Los números también pueden tener su parte fraccionaria. El punto decimal está entre los dígitos del número. Pero, los números de punto flotante tienen el punto decimal en cualquier punto entre los dígitos, como: 3.14159 o 651.75.

Operación binaria aritmética (+, -, *, /)

Las operaciones para la aritmética básica: sumar, restar, multiplicar y dividir.

Recordatorio (%)

Este es un operador extra para la división. Puede averiguar cuánto queda si un número no se divide en el otro número de manera uniforme.

Sabemos que 4/2 = 2, por lo que 2 se divide en 4 uniformemente. Pero, 5/2 = 2 con un resto de 1. Entonces, la operación de resto, 5% 2 = 1, da el número que queda de una operación de división.

ACTIVIDAD

Crearemos un juego donde pondremos en práctica las operaciones básicas. El juego se llama ¡Saludo!

¡Saludo! es un juego de matemáticas simple en el que los jugadores seleccionan una carta numérica de un mazo (sin mirarla) y la sostienen en su frente como en un "saludo". Otro jugador decide si hacer una suma o producto de las dos cartas y luego anuncia el resultado. Basándose en la carta que tiene el jugador contrario y el resultado anunciado, cada jugador intenta averiguar qué carta tiene.

Objetivos para esta actividad.

Esta es una actividad grupal, ya sea para un salón de clases o solo para amigos reunidos.

- ¡Descomponga los pasos necesarios para codificar y reproducir el saludo! Juego de matemáticas utilizando micro: bits.
- Cree un programa para mostrar un número aleatorio, en un rango apropiado, en la Matriz de LED de un micro: bit e instale correctamente el programa en un micro: bit.
- Opcionalmente, cree un programa para mantener el puntaje en el juego usando un micro: bit e instale correctamente el programa en un micro: bit. Opcionalmente, haga una diadema para sostener el micro: bit y la batería en la frente durante el juego.
- Juega el micro: bit ¡Saludo! Juego de matemáticas, utilizando micro: bits, en grupos de 3.

Materiales

- 2 o 3 micro: bits con paquetes de baterías y cables USB
- Papel y lapiz
- Opcional, materiales para crear bandas para la cabeza para sostener el micro: bit

Codificando el juego

Dos jugadores

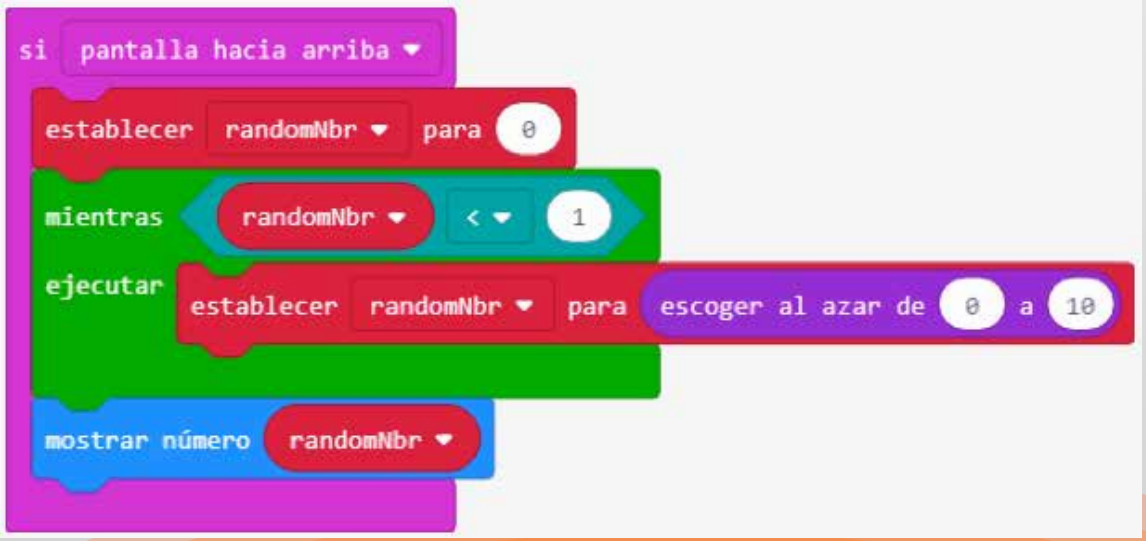
- Dos de las micro:bit están programados para mostrar un número aleatorio dentro de un rango específico basado en un "saludo". Un saludo es un evento / aporte acordado que el estudiante puede hacer que suceda sin necesidad de ver su matriz LED.
- Incluya un bucle para verificar que el número sea mayor que 0 y / o 1 si esos números no se incluirán en el juego.
- Si el número superior en el rango es mayor que 9, incluya un bucle para mostrar el número varias veces o proporcione otra forma de volver a mostrar el número.

Tres jugadores

Si el uso de micro:bit, está programado para llevar la cuenta en el juego utilizando el A y B botones para mostrar la puntuación basada en un evento diferente, como por ejemplo, A+B el botón. También debería haber una manera de restablecer las puntuaciones para la siguiente ronda de juego.

Hacer las tarjetas de números

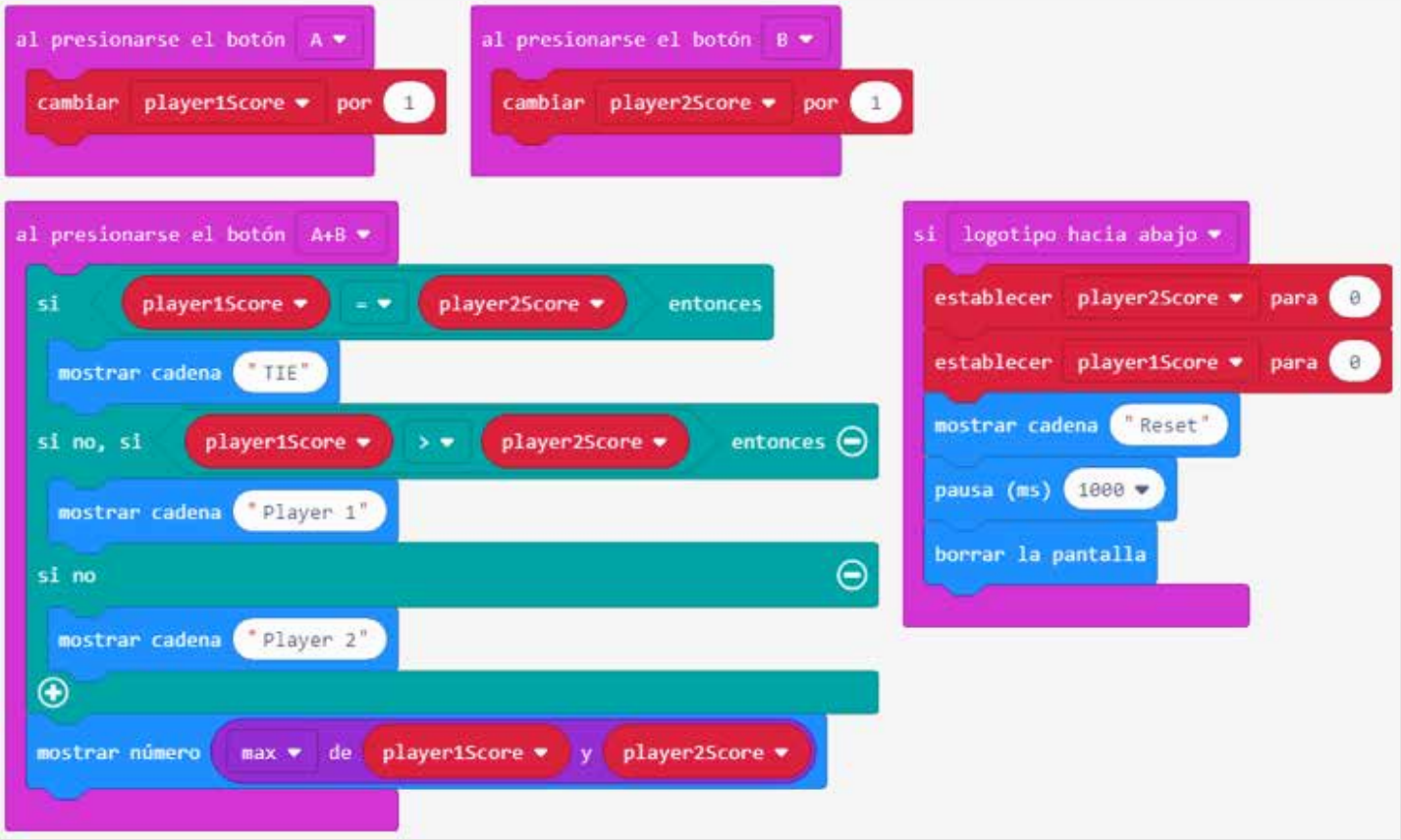
Elija un número aleatorio entre 0 y 9.



Elija un número aleatorio entre 1 y 9.

Anotador

El programa de anotador agrega un punto para un jugador cuando se presiona el botón A o B. Presionando A+B muestra el ganador actual y la puntuación. Las puntuaciones se restablecen al dar vuelta el micro: bit



Juego de juego

Antes de que comience el juego, decida cuántas rondas se jugarán o por cuánto tiempo se jugarán. También, decida cómo se determina el empate: ¿ambos jugadores obtendrán un punto o ninguno de los jugadores obtendrá un punto?

1. Los jugadores 1 y 2 se sientan o se colocan uno frente al otro con sus micro:bit en la frente para que puedan ver la matriz de LED en el micro:bit del otro jugador. Los jugadores no deben poder ver la matriz de LED en su propio micro:bit. NOTA: Es divertido crear una banda de algún tipo que sostenga el micro:bit en la frente y que la acción que

selecciona el número sea algo relacionado con el movimiento de la cabeza.

2. El jugador número 3 se sentará o se pondrá de pie para poder ver las micro:bit del jugador 1 y del jugador 2.

3. El jugador 3 dirá "¡Saludo!" Y los jugadores 1 y 2 deberían "saludarse" con sus micro:bit que mostrarán un número aleatorio en la matriz de LED de sus micro:bit.

4. El jugador 3 se suma rápidamente (si está haciendo un juego de suma / resta) o multiplica (si está haciendo un juego de multiplicación / división) los números que se muestran en las micro:bit del jugador 1 y el jugador 2. El jugador 3 debe decir el resultado. Si desea enfatizar el vocabulario matemático, requiera que el Jugador 3 use el vocabulario matemático adecuado: "La suma es ..." O "El producto es ...".

5. Los jugadores 1 y 2 intentan ser los más rápidos para descifrar y decir el número en su micro: bit. El primer jugador en identificar correctamente el número en su micro: bit "gana" la ronda. Si se usan ceros en un juego de multiplicación, un jugador que puede ver un cero en el otro micro: bit, puede decir "cualquier número" como una respuesta correcta. Si desea enfatizar el vocabulario matemático adecuado, el jugador debe decir todo el dato matemático y no solo el número. Por ejemplo, si juega un juego de multiplicación de datos:> el micro: bit del jugador 1 muestra un 2> el micro: el bit del jugador 2 muestra un 6> * el jugador (1 o 2) que dice correctamente "2 veces 6 (o 6 veces) 2) igual a 12 "ganaría la ronda".

6. El jugador 3 luego se suma a la puntuación:> Si se programó un tercer micro: bit para mantener la puntuación, el jugador 3 debe presionar el botón A si el jugador 1 ganó la ronda o el botón B si el jugador 2 ganó la ronda. > Si solo se están usando dos micro: bits, el jugador 3 debe mantener la puntuación en el papel.

7. Al final de una serie de rondas (o finalización del tiempo especificado), el puntaje total para ambos jugadores debe anotarse en papel. Si el tercer micro: bit fue programado para mantener el puntaje, el jugador 3 debería mostrar los puntajes del micro: bit y anotarlos en una hoja de puntaje. Esta hoja suele tener los nombres de los jugadores, no los números de los jugadores, ya que pueden cambiar de posición.

Si el tiempo lo permite, los jugadores deben cambiar de posición (por ejemplo, el Jugador 1 se convierte en Jugador 2, el Jugador 2 se convierte en Jugador 3 y el Jugador 3 se convierte en Jugador 1) y repite la misma cantidad de tiempo / número de rondas hasta que todos tengan la oportunidad de ser Jugadores. 3. Si el tiempo no lo permite, juegue el juego nuevamente en otro día, cambiando de posición.

Variaciones

- Si hay suficientes personas para jugar en grupos de 4, el jugador 3 diría "¡Saludo!" Y la suma o el producto de las dos cartas, como de costumbre. El jugador 4 sería el guardián de la puntuación y el árbitro en cualquier vínculo 0 tendrá un micro: bit adicional con un número mostrado y los jugadores deben calcular usando los tres micro: bits con números mostrados.

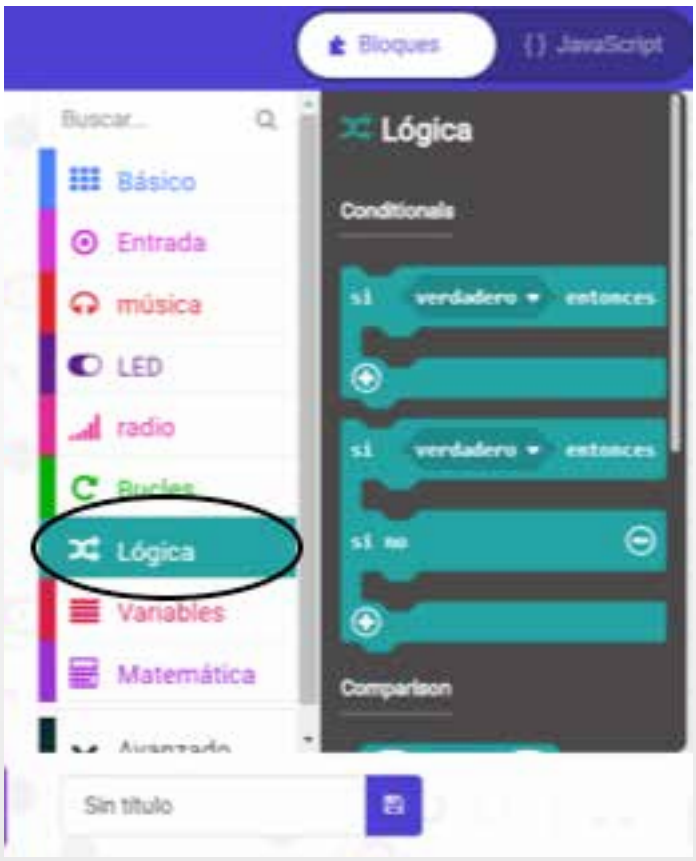
- Los jugadores mayores podrían crear este juego para los más pequeños, asegurándose de usar un rango de números apropiado para el nivel de grado. Los jugadores más viejos, o los jugadores más viejos y más jóvenes juntos, podrían hacer algún tipo de diadema para sostener el micro: pedazo en la frente. Los jugadores mayores les enseñarán a los jugadores más jóvenes cómo funciona el juego.

- Si solo usa dos micro: bits y desea mantener la puntuación: agregue el código para mantener la puntuación en uno de los micro: bits que se están utilizando y use los botones Ay B para mantener la puntuación en ese micro: bit.

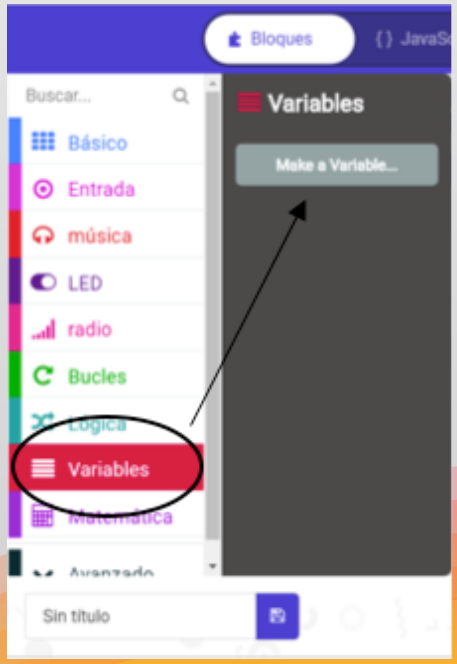

SEGUNDA
CLASE

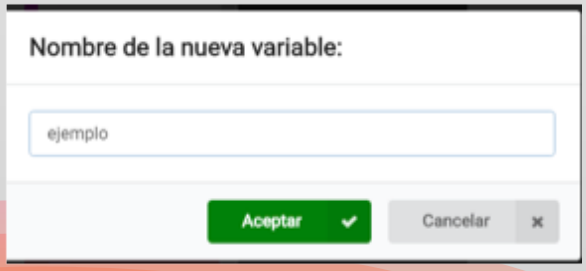
Condiciones

- Condicionales: En programación, una sentencia condicional es una instrucción o grupo de instrucciones que se pueden ejecutar o no en función del valor de una condición.



- Declaración: Una declaración es una construcción del lenguaje que asocia un nombre con una entidad.





Enlace
https://youtu.be/GGHS0vXEFdk?list=PL-N6j36Yd89u2AE6rxTZhVr6g7HXqi1R3
(declarar y condicionar).

Actividad: Juego de mesa (Tiempo de reacción)

Materiales:

- cartulina
- papel de aluminio
- marcadores permanentes
- 1 micro: bit, soporte de batería y 2 baterías AAA
- 4 pinzas de cocodrilo

Paso 1: Variables

Para que el Tiempo de reacción rastree la velocidad de la reacción de un jugador, necesitamos agregar variables para mantener algunos datos. Inicializamos (asignamos o configuramos) las variables a algunos valores iniciales. Las variables que se necesitan son: start, end, false_start, y running. Establezca los valores de las variables starty enda 0, lo que significa que no ha transcurrido ningún tiempo. A continuación, establezca el valor de las variables false_starty runningpara falsedecir que no hemos empezado todavía.

Por lo tanto, nuestras variables de seguimiento hacen esto: - el experimento de tiempo de reacción comienza y termina en momentos específicos según la reacción del jugador. - el código rastrea cuando el experimento se está ejecutando, así como cuando el jugador tiene un inicio falso.

Agregue estas variables a su código:

Paso 2: en el pin presionado

Necesitamos registrar los controladores de eventos que se ejecutarán cada vez que el usuario presione el pin GND con una mano y presione el pin P0 o P1 con la otra mano lo que completa el circuito. Nuestros manejadores de eventos son dos **On pin pressed** bloques, uno para P0 y otro para P1

.

Agrega los **On pin pressed** bloques a tu código:

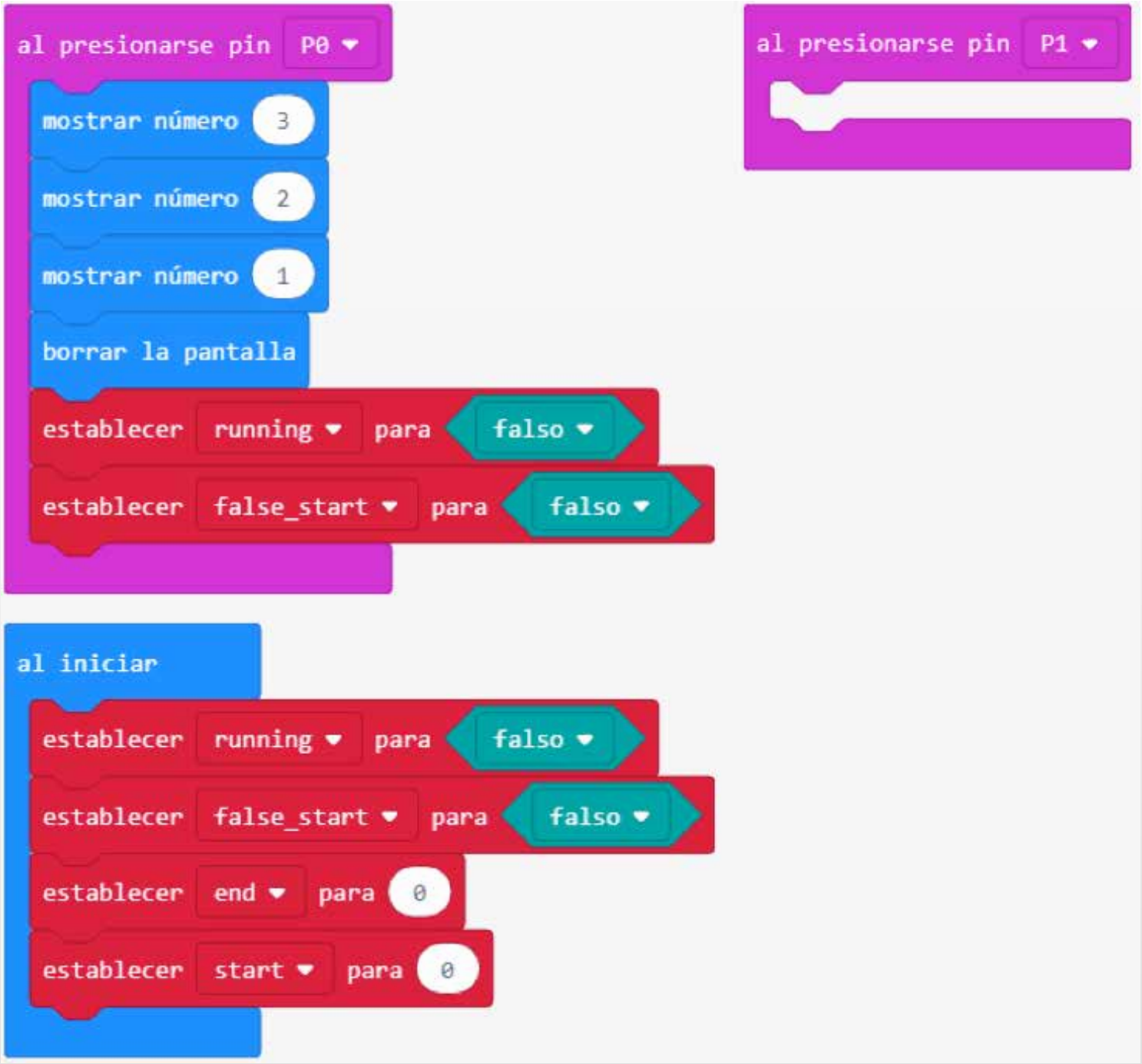
Paso 3: Temporizador de cuenta regresiva

Necesitamos un temporizador de cuenta regresiva que muestre los segundos de cuenta regresiva cuando se presiona el pin P0 . Insertemos tres **Show Number** bloques para mostrar visualmente la secuencia de cuenta regresiva: 3 ... 2 ... 1 . A continuación, agregue un **Clear Screem** bloque para borrar los números de la pantalla. Modifica tu código para que se

Paso 4: Boolean

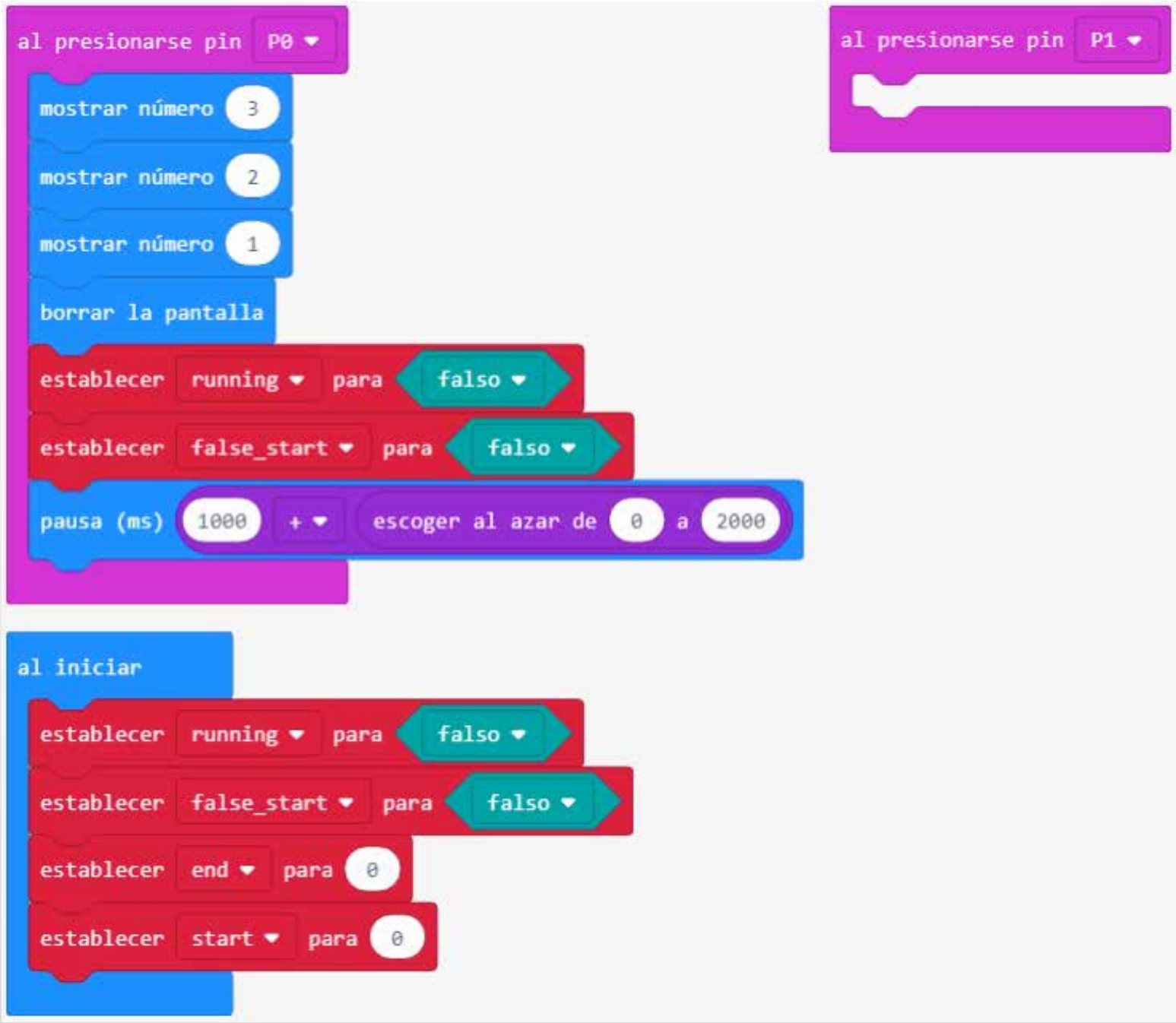
Ahora vamos a establecer las variables runningy false_startque falseen el P0 evento.

Agrega los **Set to** bloques para runningy de false_startesta manera:



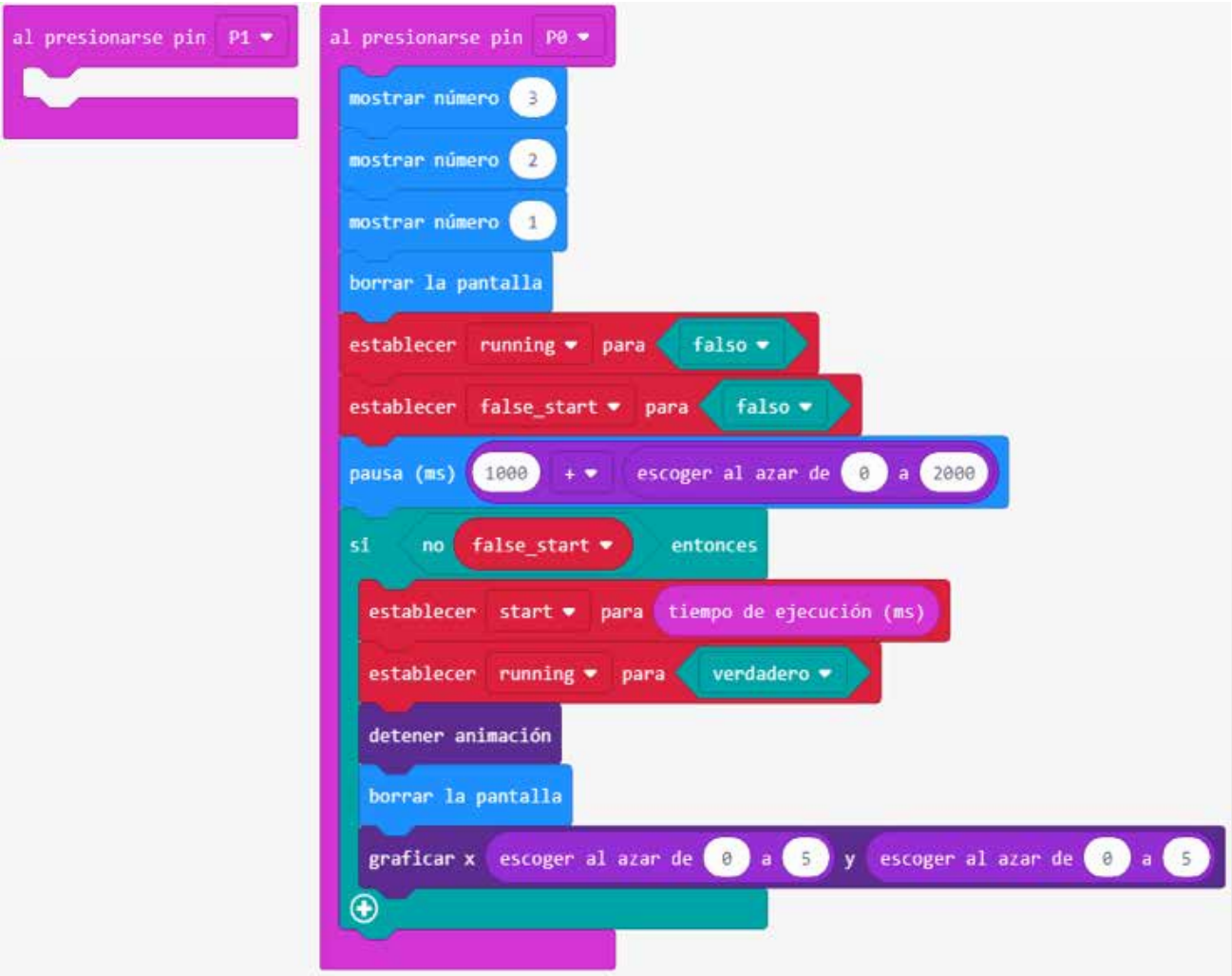
Paso 5: Comience el tiempo de reacción al azar

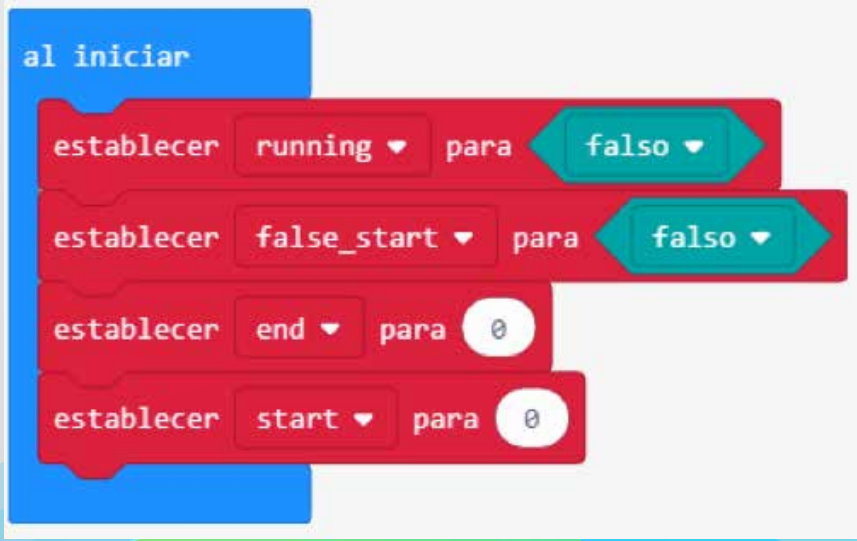
Agreguemos un tiempo de inicio aleatorio después de presionar el pin p0 . Incluya el **Random** bloque en una **Pause** en la parte inferior del bloque de eventos de esta manera:



Paso 6: Trazar el LED en X, coordenadas Y aleatoriamente

El tiempo de reacción comenzará si no se detecta un inicio falso (el pin P0 se presiona en el momento equivocado). Cuando comienza el tiempo de reacción, un LED se traza aleatoriamente en algunas de las coordenadas x e y en la pantalla. Agregue los bloques contenidos en los **If then** que muestran el tiempo de reacción:



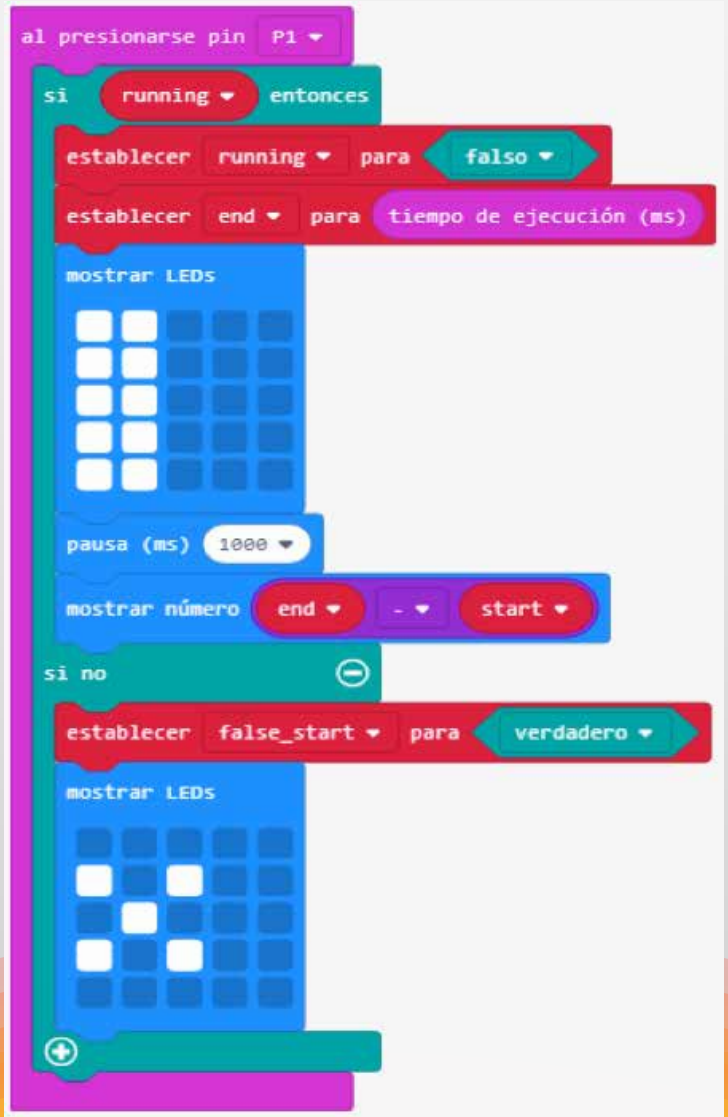


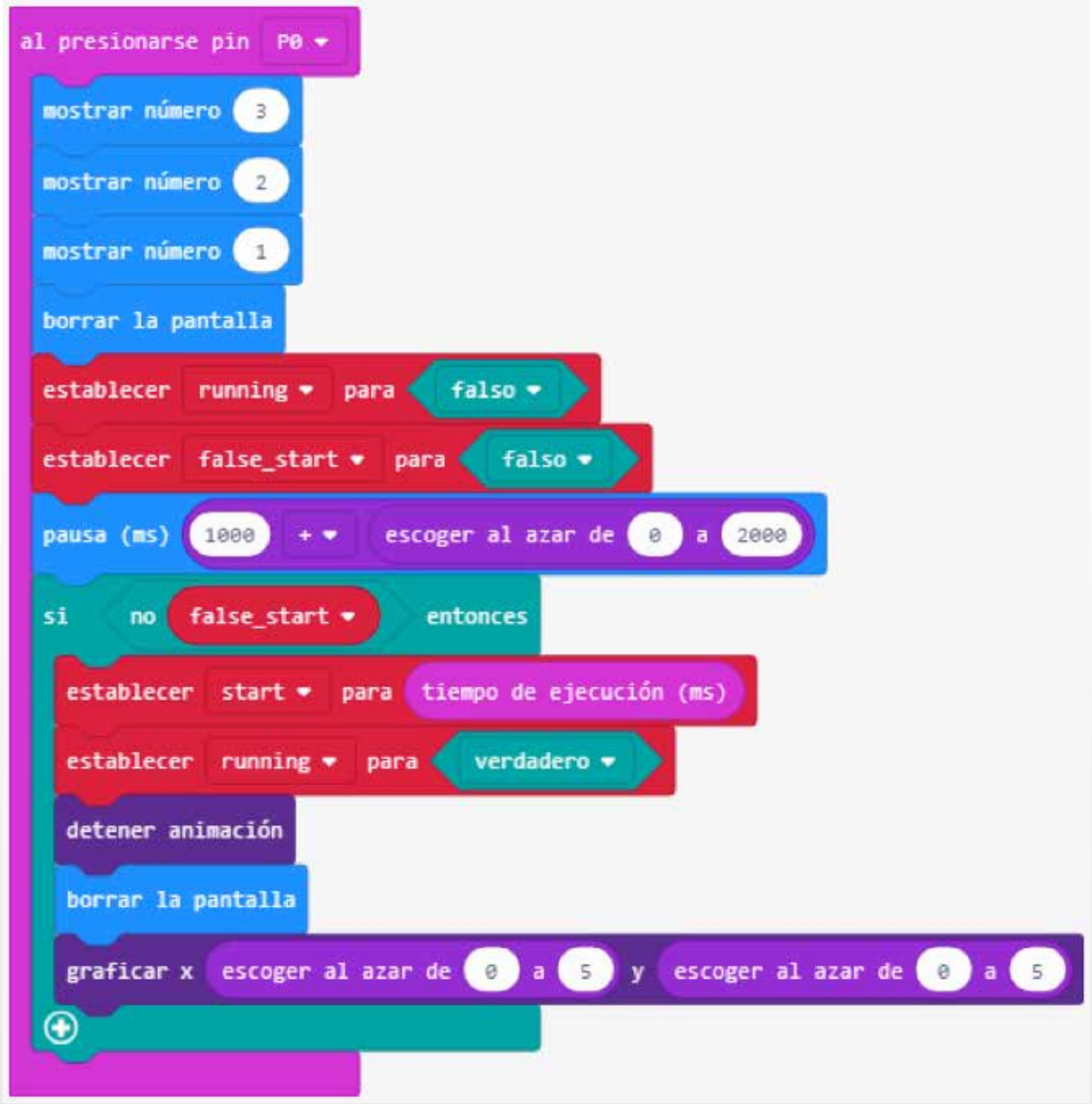
Paso 7: Mostrar retroalimentación a la reacción

Agregue un código para detectar cuándo el jugador presiona la lámina GND con una mano y el pin P1 con la otra. Este código detecta la conexión del circuito y apaga el temporizador. Además, agregue código para que el micro: bit lea el tiempo en milisegundos desde que se inicia el temporizador y se completa el circuito. Este código también detecta si hay una reacción correcta o un inicio falso si se presiona el pin P1 .

Vamos a mostrar una de las dos imágenes si se presiona el pin P1 . La primera imagen se muestra si el jugador completa correctamente el circuito entre GND y P1 . Esto significa que se produjo una reacción correcta para completar el circuito con el pin P1 presionado después de que aparece el LED generado aleatoriamente.

La segunda imagen se muestra si el jugador completa un circuito entre GND y P1 pero en un inicio falso. Se detecta un inicio falso si el jugador completa un circuito si se presiona el pin P1 antes de que el LED aparezca aleatoriamente en su coordenada aleatoria x, y. Modifique el código para incluir las acciones para el evento pin P1 :




al presionarse pin P0
mostrar número 3
mostrar número 2
mostrar número 1
borrar la pantalla
establecer running para falso
establecer false_start para falso
pausa (ms) 1000 + escoger al azar de 0 a 2000
si no false_start entonces
establecer start para tiempo de ejecución (ms)
establecer running para verdadero
detener animación
borrar la pantalla
graficar x escoger al azar de 0 a 5 y escoger al azar de 0 a 5

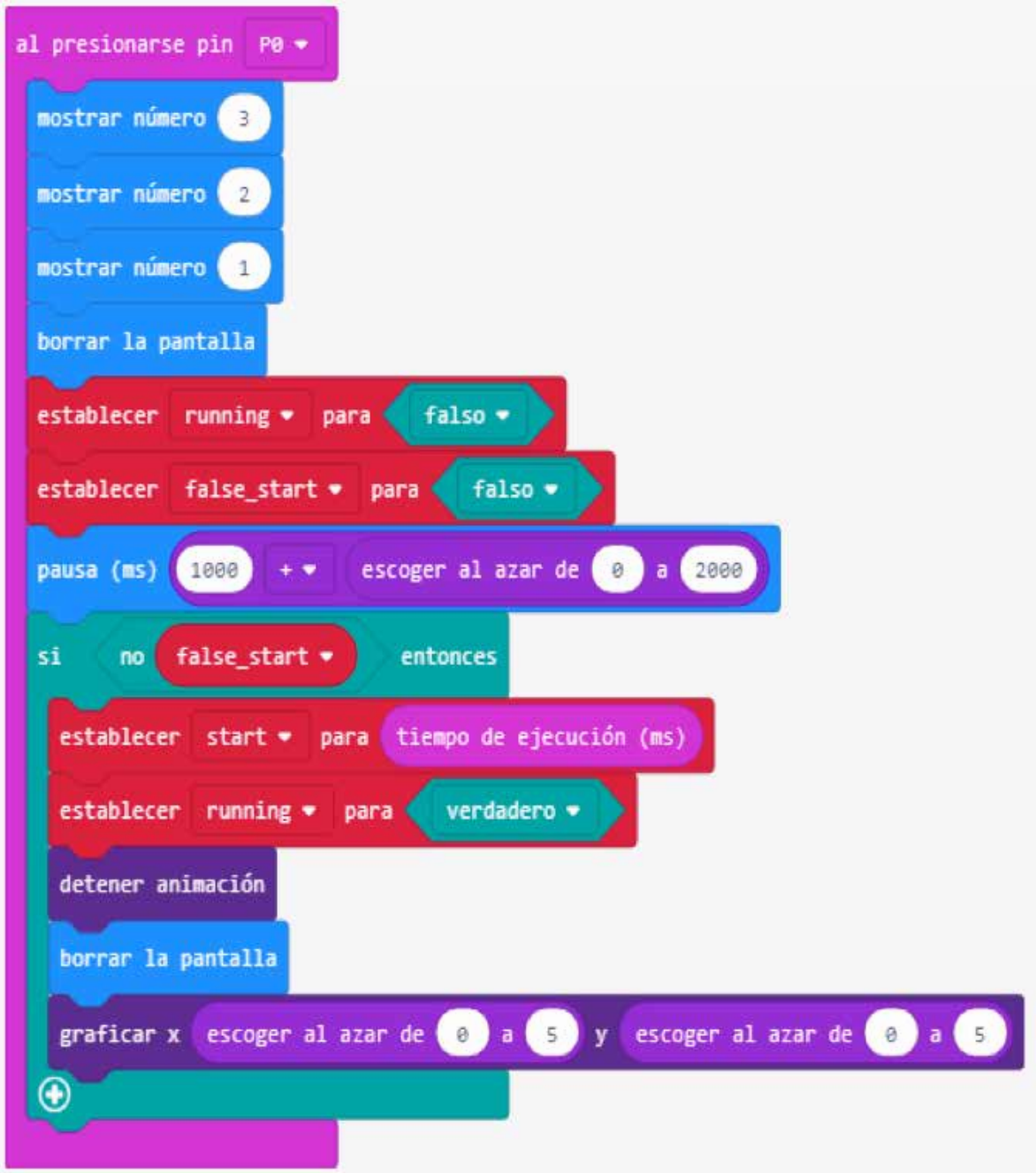

al iniciar
establecer running para falso
establecer false_start para falso
establecer end para 0
establecer start para 0

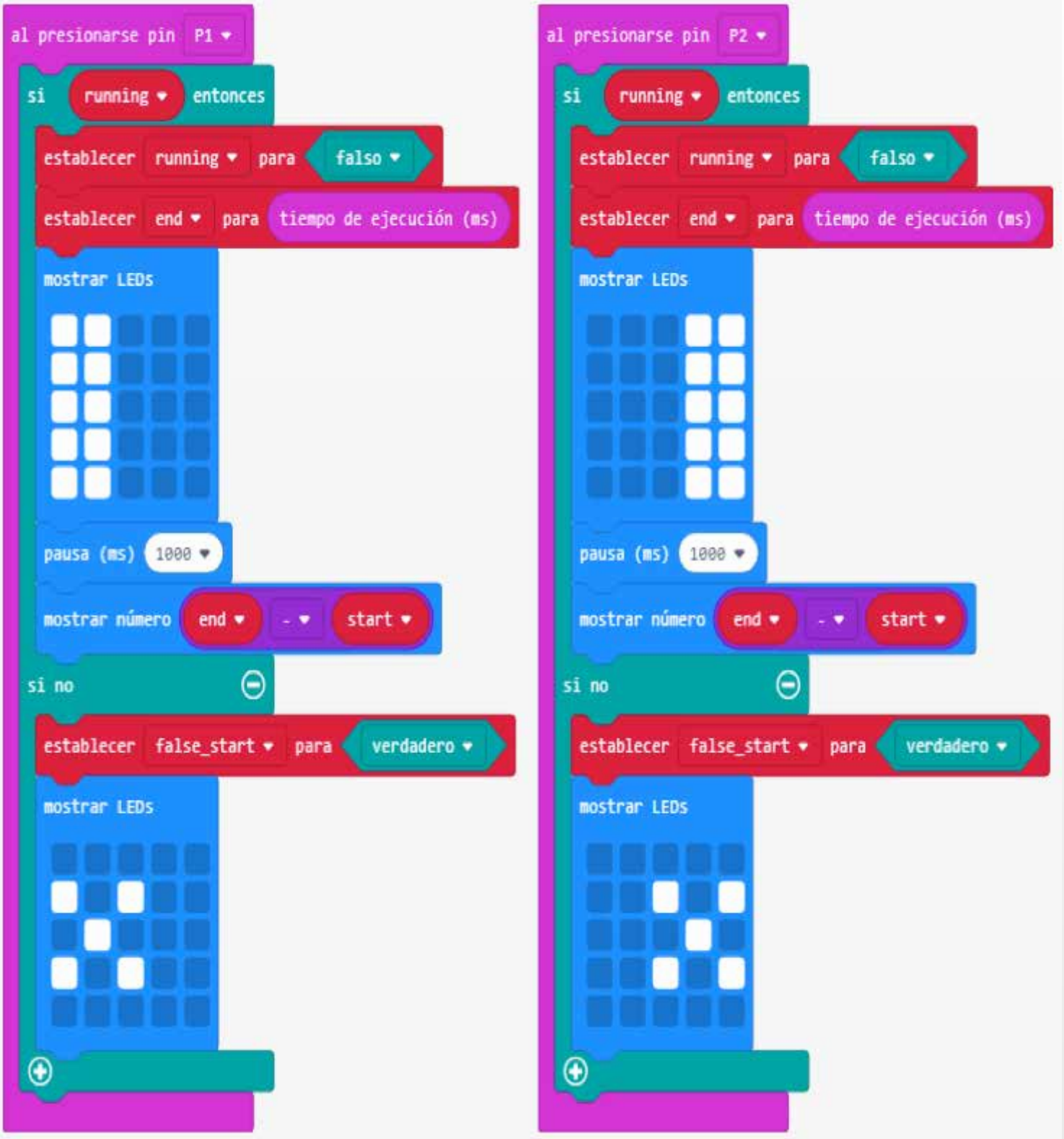
ACTIVIDAD

Extensión

Después de que los estudiantes hayan terminado sus experimentos. Pídales que jueguen con un amigo (usando el pin P2) y participe en algunos concursos para ver quién es el más rápido en el sorteo.

Puede encontrar el código para esto a continuación:

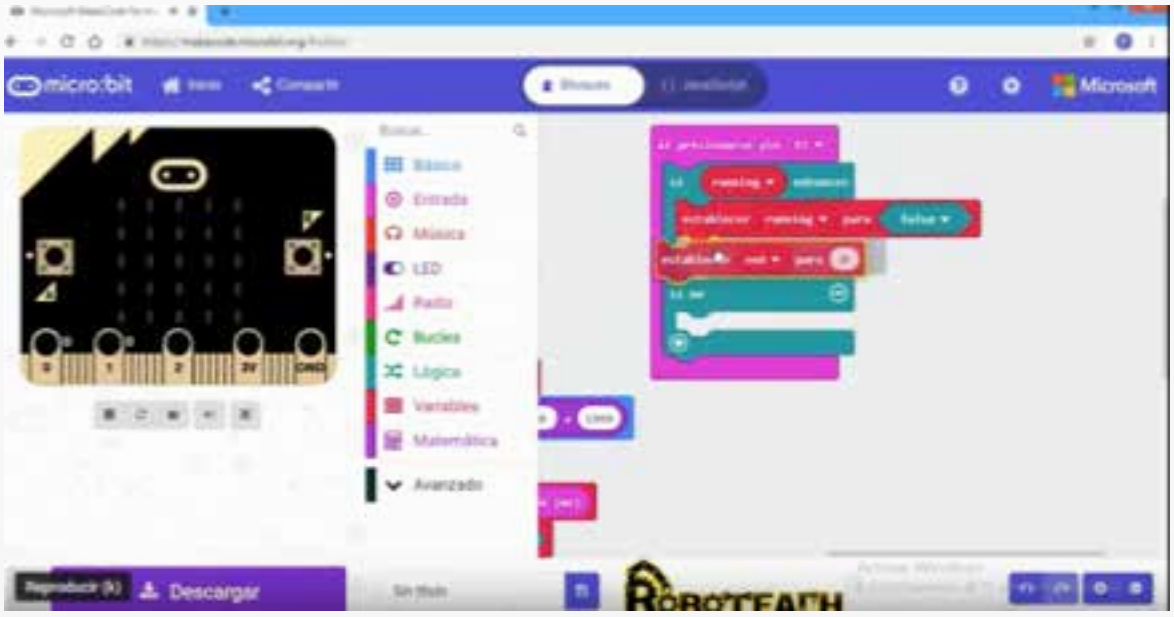



al presionarse pin P1
si running entonces
establecer running para falso
establecer end para tiempo de ejecución (ms)
mostrar LEDs
pausa (ms) 1000
mostrar número end - start
si no
establecer false_start para verdadero
mostrar LEDs
al presionarse pin P2
si running entonces
establecer running para falso
establecer end para tiempo de ejecución (ms)
mostrar LEDs
pausa (ms) 1000
mostrar número end - start
si no
establecer false_start para verdadero
mostrar LEDs

Enlaces:
Programación

1) https://youtu.be/xLehLdOhBcE?list=PL-N6j36Yd89u2AE6rxTZhVr6g7HXqi1R3

2) https://youtu.be/4mADdTM_4So?list=PL-N6j36Yd89u2AE6rxTZhVr6g7HXqi1R3




TERCERA
CLASE

Radio Control a mini buggy

Radiocontrol (RC) es la técnica que permite el control de un objeto a distancia y de manera inalámbrica mediante una emisora de control remoto.
En el radiocontrol entran en juego tres técnicas fundamentales: la electrónica que se encarga de transformar los comandos dados en ondas de radio en el transmisor y a la inversa en el receptor, la electricidad, encargada de proporcionar la energía necesaria a los dispositivos tanto el comando (o transmisor) como el receptor y la mecánica encargada de mover los accionadores (o servos) que dan las señales eléctricas de moduladas o decodificadas en movimiento mecánico.

Existen todo tipo de vehículos de modelismo dirigidos por radiocontrol, siendo los más populares los coches, los aviones, los barcos, los helicópteros y los submarinos.

Materiales:

- **Dos micro bit.**
- **Porta baterías**

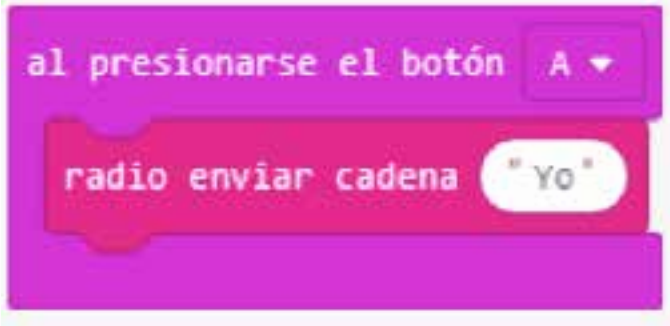
Paso 1:

Usa la radio para enviar y recibir mensajes con otro micro: bit.

Paso 2:

Enviando un mensaje

Se usa **on button pressed** pressedpara enviar un mensaje de texto por radio con **sned string**. Cada micro: bit cercano recibirá este mensaje.



Paso 3:

Recibiendo un mensaje

Agregar un **on received string** bloque para ejecutarse cuando se recibe un mensaje.

Paso 4:

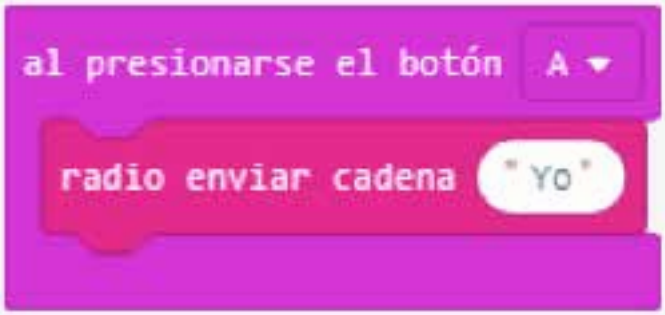
Mostrando texto

Agregue un **mostrar cadena** para mostrar la cadena en la pantalla. Encontrará la receivedStringvariable en Variables en la caja de herramientas.

Paso 5:

Pruebas en el simulador.

Presione el botón A en el simulador, notará que aparece un segundo micro: bit (si su pantalla es demasiado pequeña, el simulador podría decidir no mostrarla). Intente presionar A nuevamente y observe que el mensaje "Yo" se muestra en el otro micro: bit.



Paso 6:

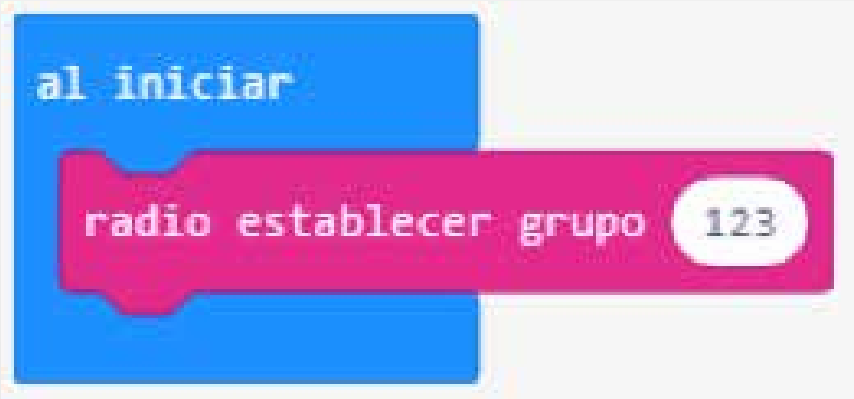
Pruébalo de verdad

Si ustedes dos micro: bits, descargue el programa a cada uno. Presione el botón A en uno y vea si el otro recibe un mensaje.

Paso 7:

Los grupos

Usa el **grupo conjunto** bloque para asignar un número de grupo a tu programa. Solo recibirás mensajes de micro: bits dentro del mismo grupo. Use esto para evitar recibir mensajes de cada micro: bit que está transmitiendo.



ACTIVIDAD

FLASH THE ZIP LED

PASO 1: Coloque las baterías en la PCB SERVO: LITE y enciéndala.

PASO 2: Conéctelo a una computadora usando un cable micro-USB.

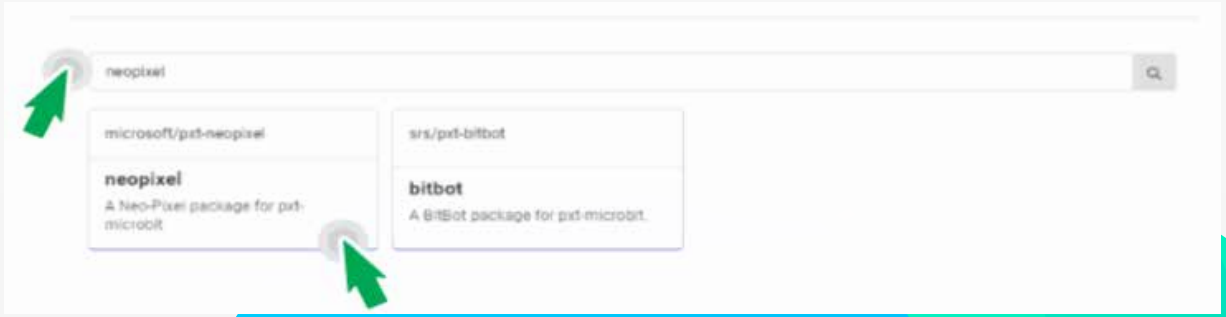
PASO 3: abrir el editor de bloques de JavaScript (makecode.microbit.org).

NOTA: los LEDs ZIP de Kitronik son compatibles con los Neopixels de Adafruit.

PASO 4: en la caja de herramientas a la izquierda de la pantalla, seleccione la sección "Avanzado". Los paquetes adicionales deben aparecer a continuación.

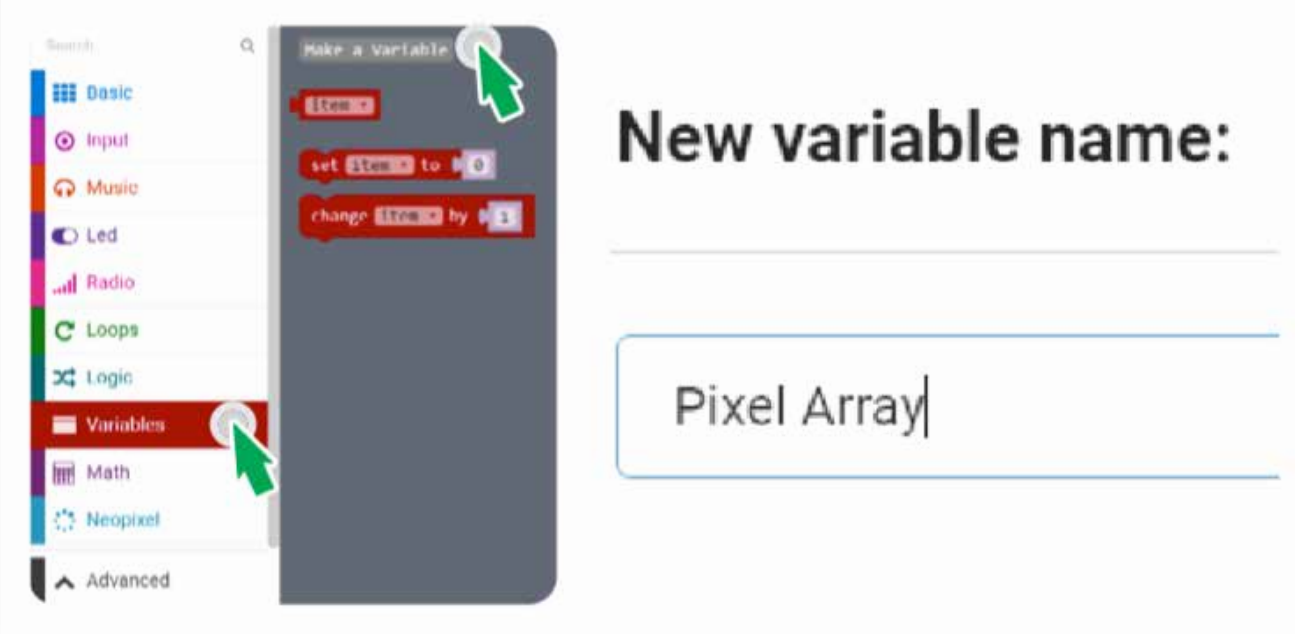
PASO 5: Seleccione 'Agregar paquete'.

PASO 6: En la barra de búsqueda, escriba "neopixel", luego seleccione la casilla "neopixel".



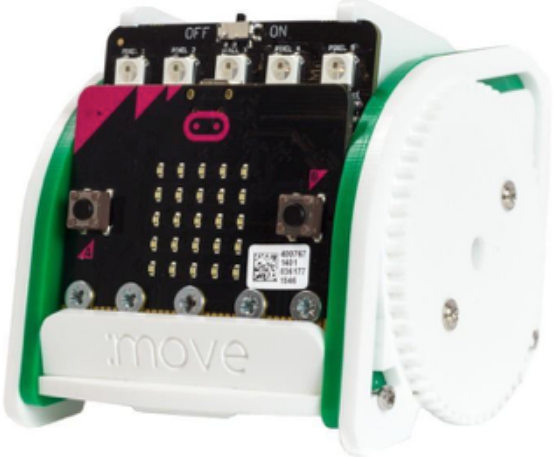
NOTA: Esto cargará un conjunto de bloques compatibles con los LEDs ZIP de Kitronik, ¡lo que los hace realmente fáciles de codificar.

PASO 7: Cree una variable y llámela 'Pixel Array'.



LO QUE ESTO SIGNIFICA

Una variable es como un contenedor que puede almacenar información. Esto podría ser un número, una palabra o una pieza de información que desea que su programa recuerde.



PASO 8: cree el código siguiente.

LO QUE ESTO SIGNIFICA

Esto indica al micro: bit que el pin 0 está conectado a 5, LEDs direccionales de color.

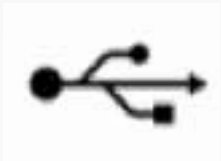
LO QUE ESTO SIGNIFICA

Cuando se presiona el botón B, desactive los LED representados por la variable ' pixel array '. Esto los desactivará

PASO 9: ¡Descarga!

El programa se ejecutará automáticamente en el simulador. Haga clic en el botón ' A ' del simulador para ver el patrón de LED. Haga clic en el botón "B" para borrar el patrón.

Descargue el programa para generar el archivo. hex.
Guarde el archivo. hex listo para subir a la BBC Micro: bit.

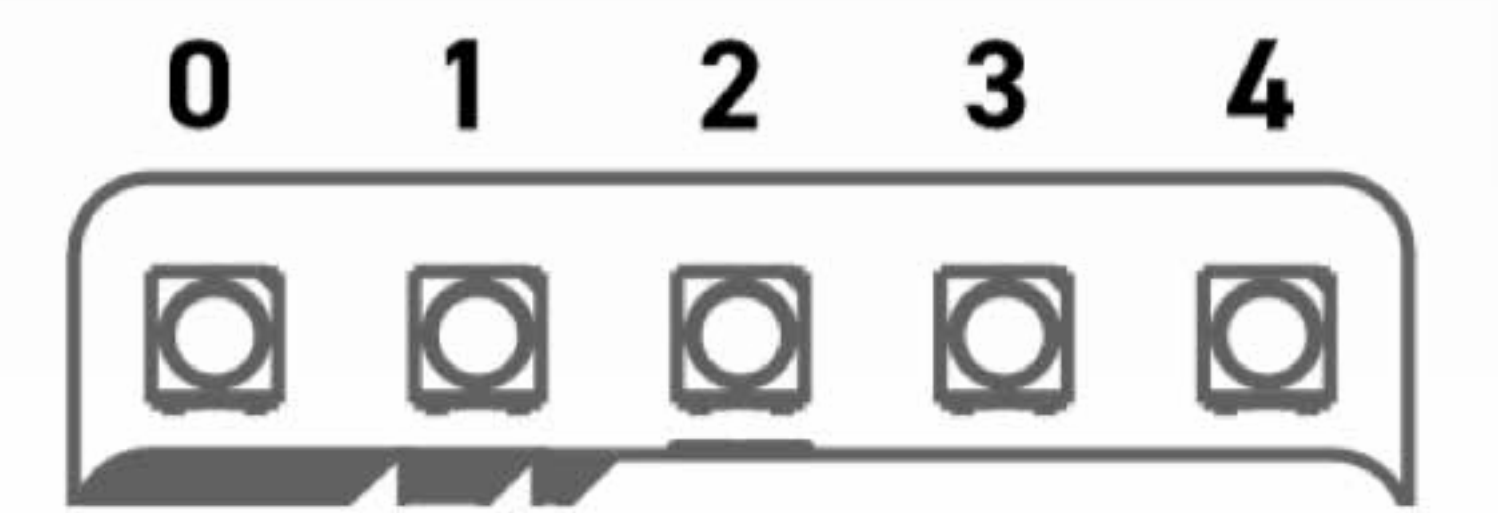


Conecte la BBC Micro: bit en un USB po luego arrastre y suelte el archivo. hex en la ventana de BBC Micro: bit para cargar el programa.

PASO 10: Presione A y prepárese para las luces!

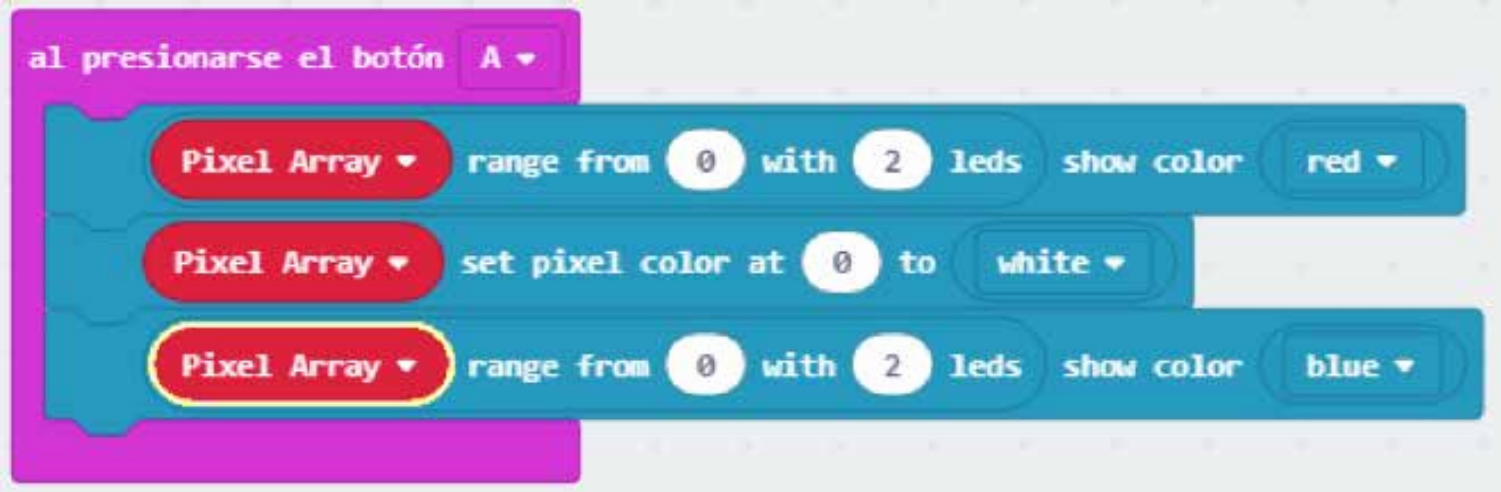
Paso 11: pruebe a cambiar el código para crear un color diferente.

Además de abordar todos los píxeles a la vez, es posible establecerlos individualmente o como grupo. El primer píxel siempre tiene una dirección de 0 (véase el siguiente diagrama).



Paso 12: cambie el código bajo el botón ' on ha presionado ' a lo siguiente.

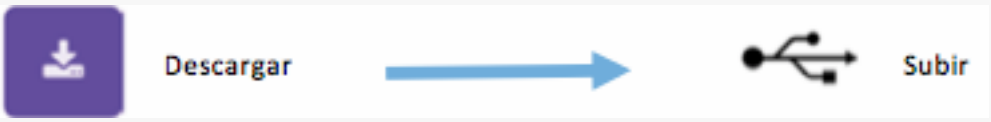
Nota: es posible que deba hacer clic en "más" en el conjunto de herramientas "neopixel" para ver los bloques adicionales.



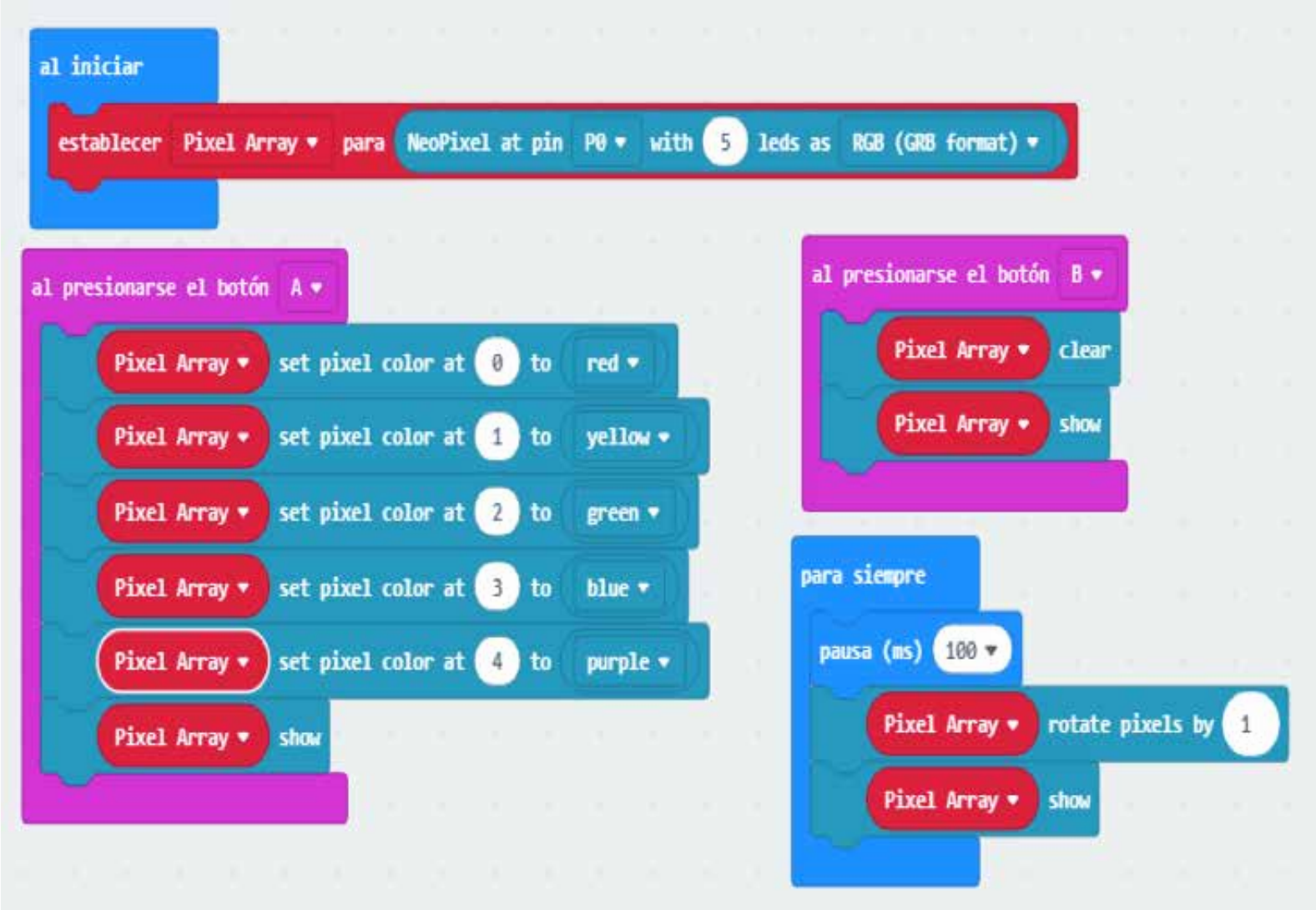
Lo que esto significa establecer los primeros dos píxeles en rojo, el píxel medio a blanco y los dos últimos a azul.

Recuerde: para mostrar un cambio debe utilizar un bloque con ' show ' en él.

PASO 13:



PASO 14: Presione A y observe las luces!



Paso 15: crea el código a continuación.

LO QUE ESTO SIGNIFICA

Este código muestra un patrón de cambio de color cuando se presiona el botón A y se detiene cuando se presiona el botón B.

Nota: la herramienta "rotar píxeles" desplaza cada color del LED hacia el siguiente LED. Cuando llega al final de la línea, vuelve al primer LED.

Paso 17: Presione A y ver el espectàculo de luz!


CUARTA
CLASE

Añadiendo Control Remoto a Bluetooth

Las tecnologías inalámbricas de infrarrojo y bluetooth son comúnmente usadas para comunicación inalámbrica de corto alcance entre aparatos electrónicos. Ambas tecnologías son efectivas, pero dadas sus capacidades y limitaciones diferentes, suelen ser mejores para aplicaciones muy diferentes. En la mayoría de los aspectos, el bluetooth tiene una ventaja distintiva, aunque el infrarrojo sigue siendo popular en algunas aplicaciones.

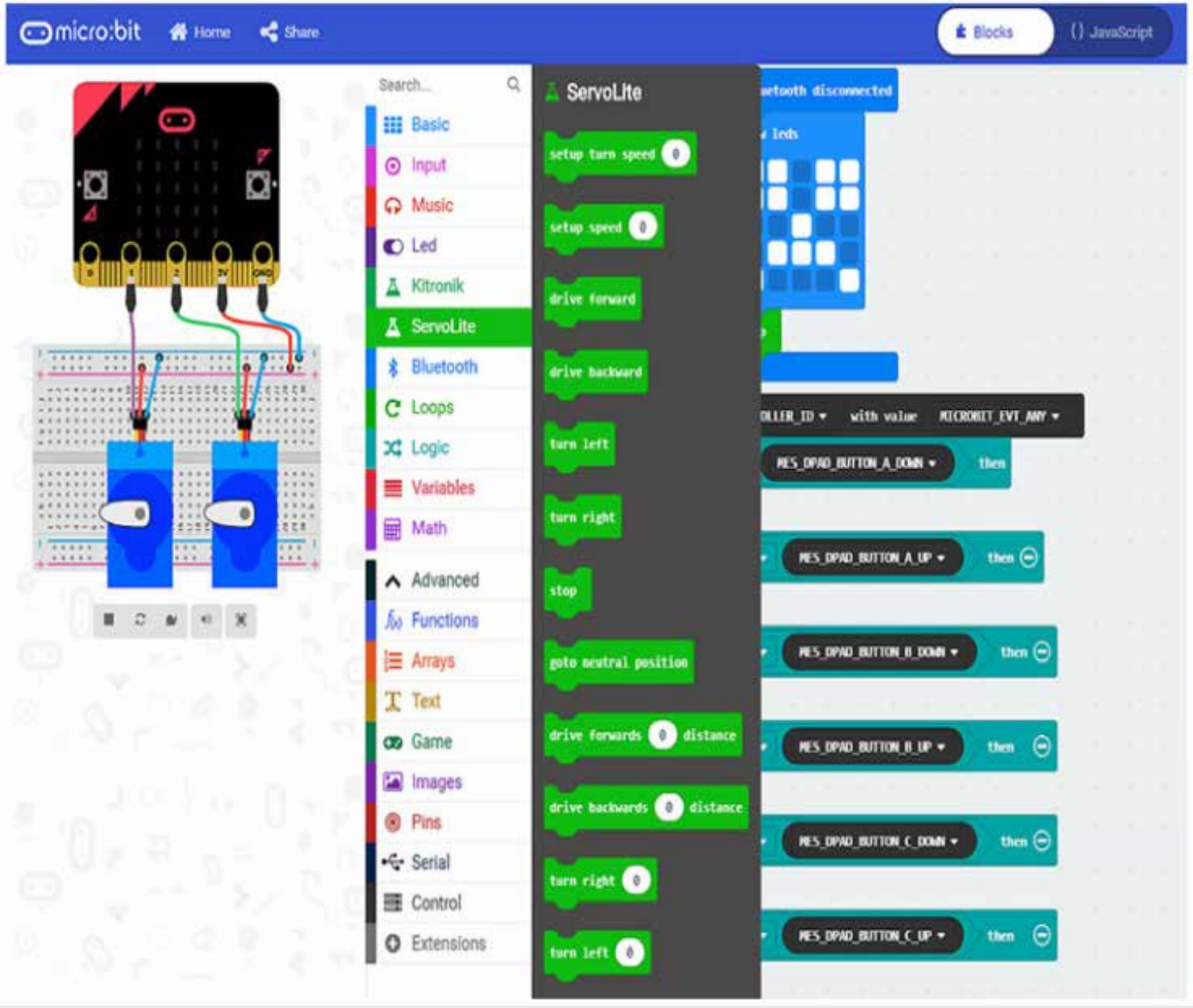
El infrarrojo inalámbrico usa pulsos de luz infrarroja para transmitir datos de un aparato a otro. Estos pulsos son invisibles para el ojo desnudo, pero pueden ser d etectados por un sensor en el aparato receptor. El bluetooth inalámbrico usa ondas de radio en una frecuencia particular (2,4 gigaHertz) para la transmisión de datos de un aparato al otro. Tanto el bluetooth como el infrarrojo consume una cantidad considerablemente menor de energía que otras tecnologías inalámbricas.

El alcance efectivo para el infrarrojo inalámbrico es muy corto, generalmente no más de cinco metros, y a menudo más cercano a un metro. El bluetooth tiene un alcance máximo de 10 metros, el cual, si bien es el doble que el del infrarrojo, es mucho menor que otras tecnologías inalámbricas de frecuencia de radio. El bluetooth tiene una ventaja clara sobre el infrarrojo en términos de alcance efectivo, pero ambas tecnologías son útiles solo para la comunicación entre aparatos relativamente próximos el uno del otro.

Agregar la funcionalidad de Bluetooth a: MOVE mini es tan fácil como poner un código en el microbit. Luego descargando la aplicación Kitronik: MOVE para Android.

Kitronik Custom Blocks para el editor de bloques de Javascript de Microbit:

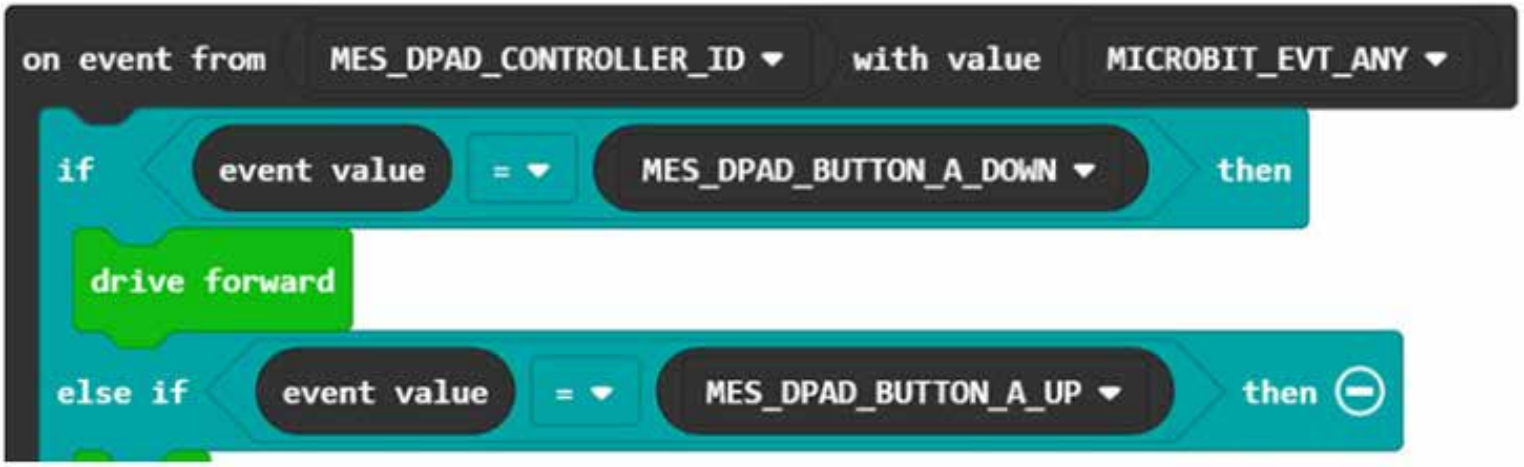
Para agregar nuestros bloques personalizados al editor. Haga clic en el engranaje en la parte superior derecha del editor y seleccione "Extensiones" en el menú. Esto abrirá una ventana de diálogo con un cuadro de búsqueda. Escriba Kitronik en el cuadro de búsqueda y todos nuestros bloques personalizados aparecerán como mosaicos. Seleccione el mosaico titulado 'servo-lite' y se agregará al menú de bloques.



Para agregar la funcionalidad Bluetooth al menú, primero debemos agregar el paquete Bluetooth. Seleccione Extensiones en el menú y luego seleccione Bluetooth. Ahora estamos listos para escribir algún código.

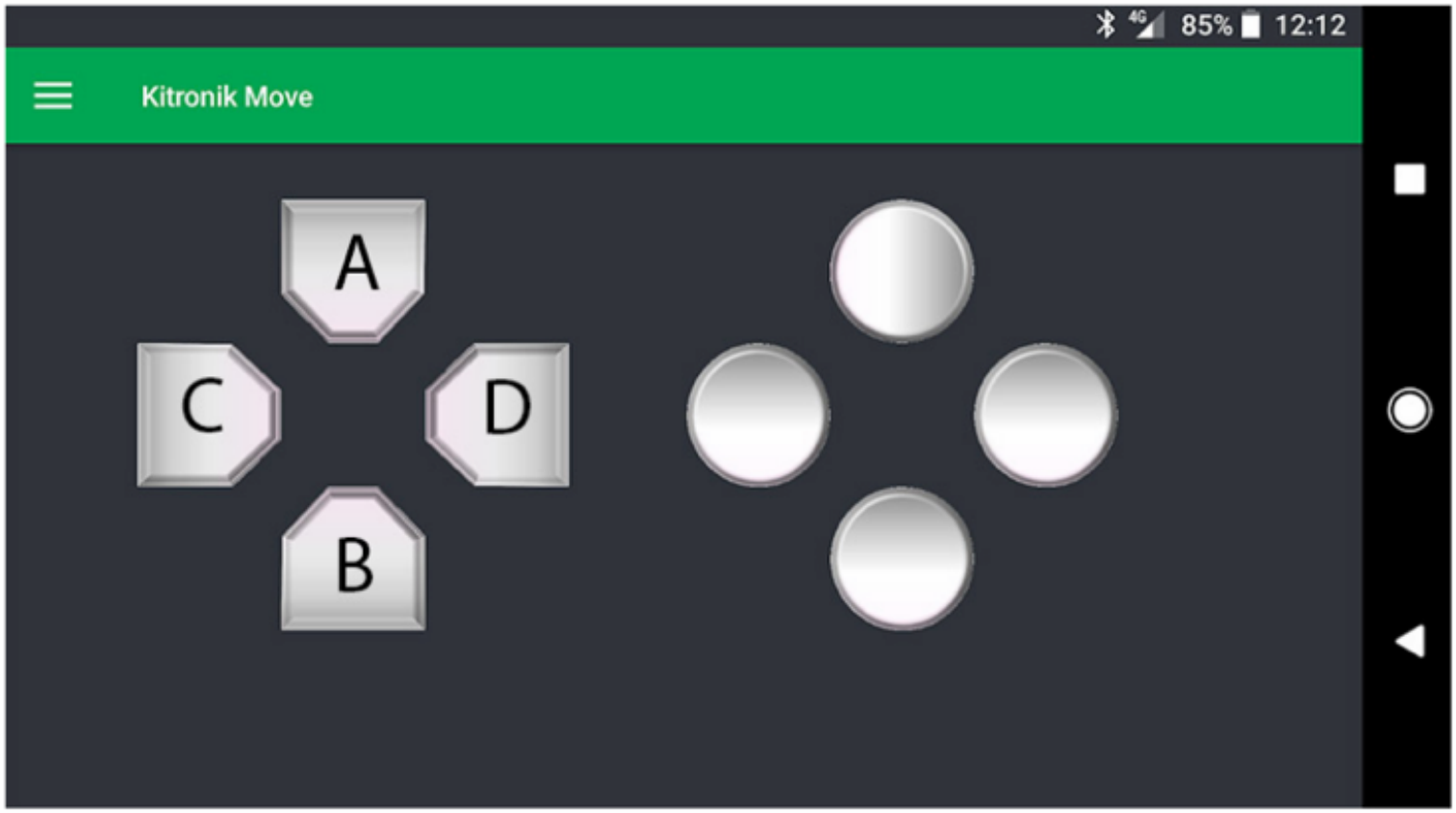
Escribiendo el código de control remoto de Bluetooth:

Como vamos a controlar: MOVE mini a través de Bluetooth necesitamos codificar el micro:bit para que responda de la manera correcta cuando recibe los comandos de Bluetooth.



Todo nuestro código de control remoto se colocará dentro de un bucle 'on event'. Como se muestra arriba. Puede encontrar el bucle 'en el evento' expandiendo la opción de menú Avanzado, luego puede encontrar el bucle en el menú de Control.

Esto asegurará que el microbit sepa esperar comandos del controlador DPad y qué recibirlos una vez que los reciba. Vamos a utilizar los botones A, B, C y D en la aplicación para conducir, puede ver cómo están orientados en los pads de la aplicación en la imagen.



Nuestros controles serán:

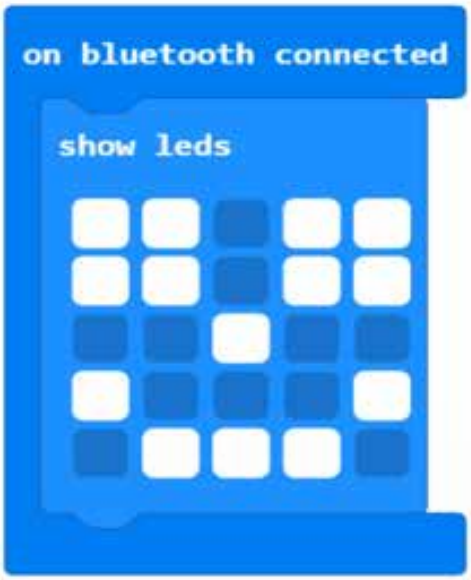
- A - Adelante.
- B - Hacia atrás.
- C - Izquierda.
- D - Derecho.

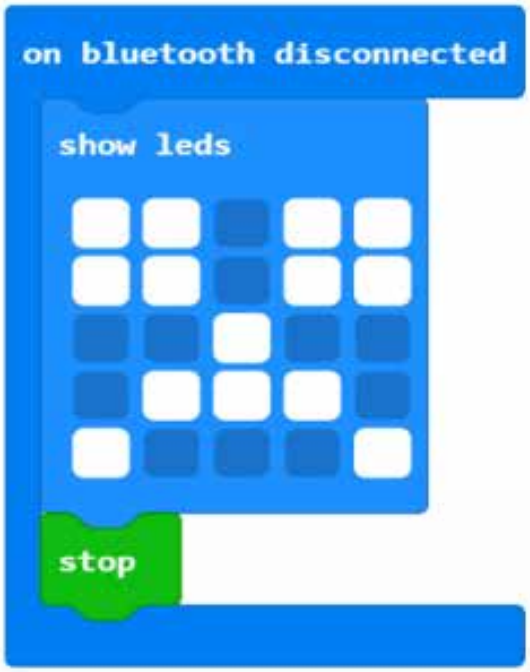
Además de escribir nuestro código para responder a una pulsación de botón, también debemos escribir código para saber qué hacer cuando se suelta el botón. Para nuestro código, la liberación del botón siempre será igual a parada.

En el código anterior puede ver cómo funciona el código para la presión del botón A. Cuando se presiona el botón A, el buggy avanzará. Cuando se suelta el botón A, el cochecito se detendrá.

Seguimos el mismo procedimiento para los otros tres botones, simplemente modificando el tipo de comando de la unidad para cada uno.

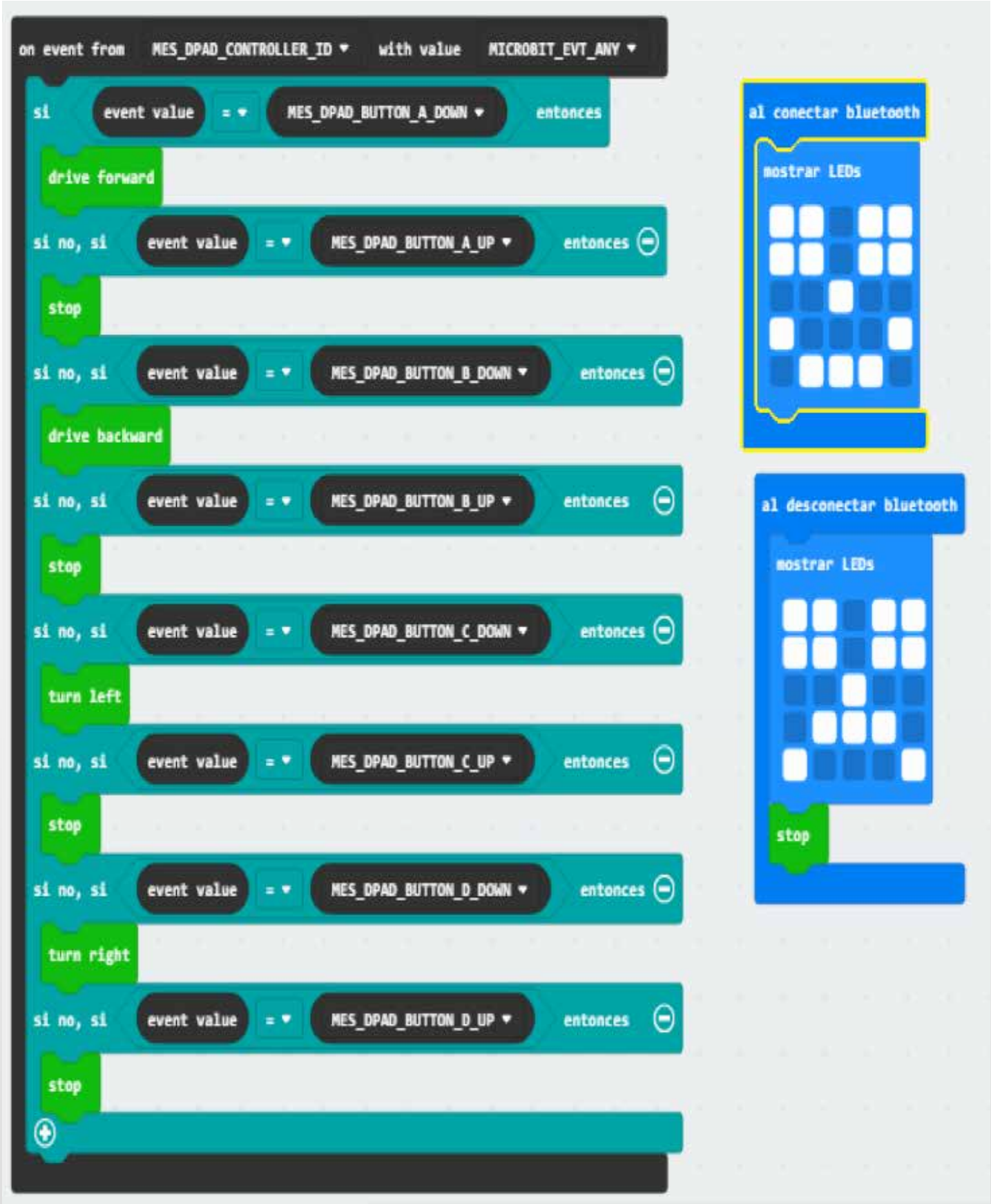
También queremos escribir un código para proporcionar información visual a través de los indicadores LED de los micro:bits para saber si Bluetooth está conectado o no. Si se interrumpe la conexión Bluetooth, lo ideal sería que el buggy se detuviera, independientemente del último comando. El comando de detención actúa como un dispositivo a prueba de fallas para proteger el sistema si falla el control.





El código anterior mostrará una cara sonriente en el micro:bit cuando Bluetooth esté conectado. Si por alguna razón se pierde la conexión, se mostrará una cara triste y los servos dejarán de girar.

En el editor incrustado a continuación puedes ver todo el programa. Una vez que comprenda cómo funciona el bloque de eventos y cómo definir un valor de evento, el código no tiene complicaciones para producir.


on event from MES_DPAD_CONTROLLER_ID with value MICROBIT_EVT_ANY
si event value = MES_DPAD_BUTTON_A_DOWN entonces
drive forward
si no, si event value = MES_DPAD_BUTTON_A_UP entonces
stop
si no, si event value = MES_DPAD_BUTTON_B_DOWN entonces
drive backward
si no, si event value = MES_DPAD_BUTTON_B_UP entonces
stop
si no, si event value = MES_DPAD_BUTTON_C_DOWN entonces
turn left
si no, si event value = MES_DPAD_BUTTON_C_UP entonces
stop
si no, si event value = MES_DPAD_BUTTON_D_DOWN entonces
turn right
si no, si event value = MES_DPAD_BUTTON_D_UP entonces
stop
al conectar bluetooth
mostrar LEDs
al desconectar bluetooth
mostrar LEDs
stop


QUINTA
CLASE

Dibujando con Mini Buggy

MOVE mini funciona con dos servomotores de rotación continua. La velocidad de estos servos se puede controlar simplemente modificando la señal PWM (Modulación de ancho de pulso) al servo, lo que es fácil de hacer usando los bloques Servo en el editor de bloques Microsoft MakeCode. También hemos producido los bloques personalizados de Kitronik para que Servo: Lite haga que la tarea de codificación sea lo más rápida y sencilla posible.
MOVE servo: lite board incluido también se puede utilizar junto con un micro: bit de la BBC para construir otros proyectos basados en el movimiento

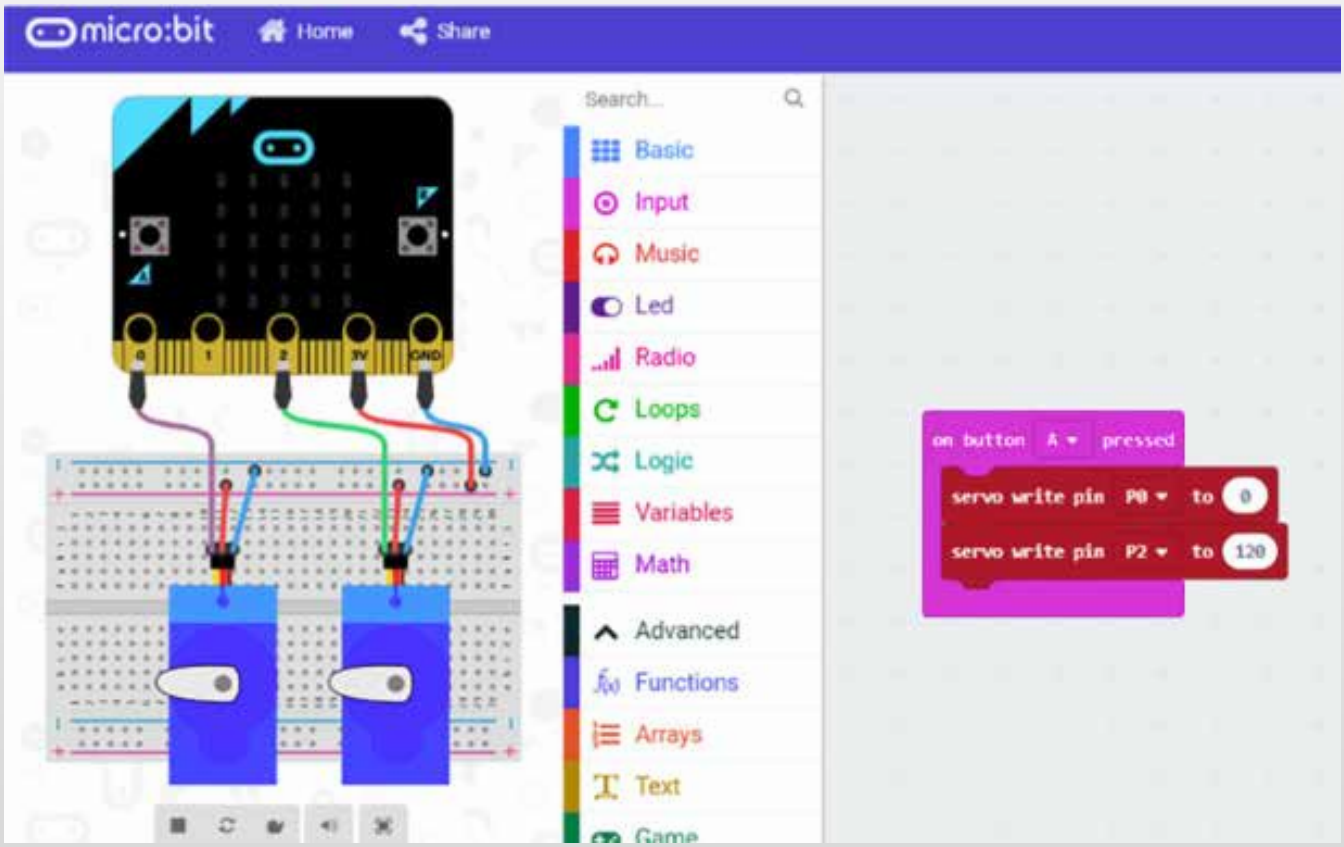
Materiales:

- **Un rotulador del tamaño y alto de un Sharpie.**
- **Micro bit.**
- **Move Buggy.**
- **Aplicación bluetooth (https://play.google.com/store/apps/details?id=com.samsung.microbit&rdid=com.samsung.microbit) o un segundo controlador micro bit para manejar el move.**

ACTIVIDAD

Dibujando con mini Buggy:

Paso 1: Para dibujar un círculo, la rueda exterior debe moverse más rápido que la interior. En a: MOVE mini tenemos velocidades de rueda controlables de forma independiente.
Cuando las ruedas van en direcciones opuestas, el: MOVE mini girará en el lugar, y luego cuando las ruedas se acerquen a la misma velocidad, formará un círculo más grande y más grande, hasta que termine en una línea recta. Este código dibuja un círculo de pequeño tamaño haciendo funcionar un motor a toda velocidad y un motor a 1/3 de velocidad.



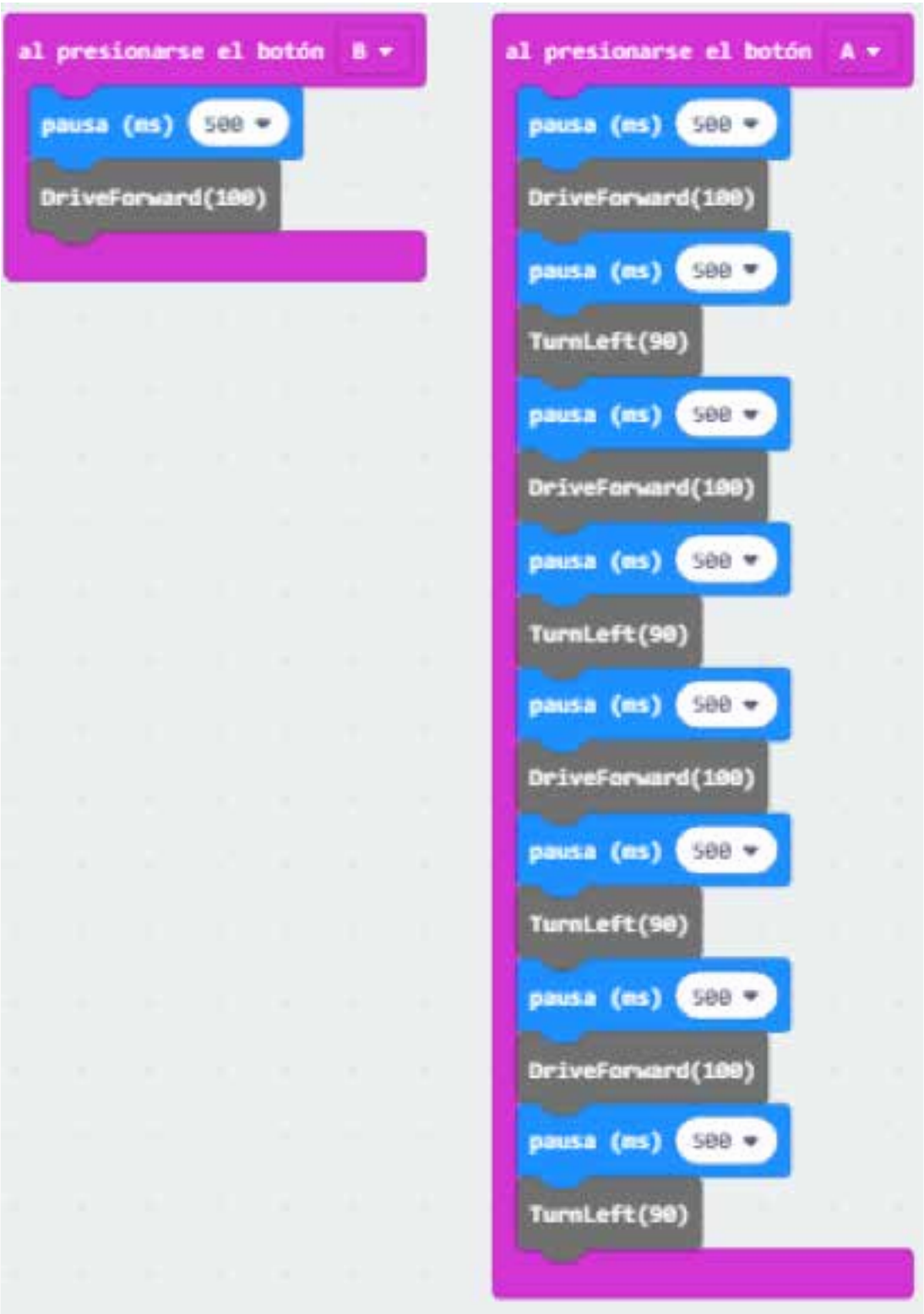
Puede editar el radio del círculo modificando la velocidad de escritura. Use una velocidad de escritura de 90 - 0 para un servo y 90 - 180 para el otro, donde 0 y 180 es una línea recta (lo que permite una variación en los servos probablemente se desvíe hacia un lado u otro).
Para otras tareas, vea si puede escribir código para hacer que: MOVE mini gire a la derecha y conduzca hacia atrás.
Combina ese código para dibujar otras formas.

Extra

Crea el código para que tu Buggy se desplace dibujando un cuadrado.

Bloque de JavaScript

JavaScript es un lenguaje muy poderoso; tan poderoso que algunos conceptos en JavaScript no pueden mostrarse usando bloques. Cuando PXT encuentra un fragmento de código que no se puede convertir en bloques, en su lugar crea un bloque gris de JavaScript para conservarlo. Estos bloques se convierten de nuevo en el JavaScript original cuando vuelves al editor de texto. El código en los bloques de JavaScript no se puede editar, pero los bloques se pueden mover, copiar, pegar y eliminar como cualquier otro bloque.


al presionarse el botón B
pausa (ms) 500
DriveForward(100)
al presionarse el botón A
pausa (ms) 500
DriveForward(100)
pausa (ms) 500
TurnLeft(90)
pausa (ms) 500
DriveForward(100)
pausa (ms) 500
TurnLeft(90)
pausa (ms) 500
DriveForward(100)
pausa (ms) 500
TurnLeft(90)
pausa (ms) 500
DriveForward(100)
pausa (ms) 500
TurnLeft(90)

Desplàzate a la pestaña JavaScript y transcribe el siguiente código como se te presenta. Al finalizar de escribir el código selecciona la pestaña de Bloques y veras como cambian cada uno de los códigos que has escrito.

```
    deje que  timeToWait = (grados * MicroSecInASecond) / Numb erOfDegreesPerSec;
    pasadores . servoWritePin (AnalogPin.P1,  45 );
    pasadores . servoWritePin (AnalogPin.P2,  45 );
    el control . waitMicros (timeToWait);
    pasadores . servoWritePin (AnalogPin.P1,  90 );
    pasadores . servoWritePin (AnalogPin.P2,  90 );
}
// Esta función impulsa los: MOVE mini hacia adelante.
// @param distancia a qué distancia conducir
Función  DriveForward (distancia:  número ):  void  {
    deje que  timeToWait2 = (distance * MicroSecInASecond) / Di stancePerSec;
    pasadores . servoWritePin (AnalogPin.P1,  0 );
    pasadores . servoWritePin (AnalogPin.P2,  180 );
    el control . waitMicros (timeToWait2);
    pasadores . servoWritePin (AnalogPin.P1,  90 );
    pasadores . servoWritePin (AnalogPin.P2,  90 );
}

```

```
deja  NumberOfDegreesPerSec =  0
deja  DistancePerSec =  0
deja  MicroSecInASecond =  0
de entrada . onButtonPressed (Button.B,  function  () {
    básica . pausa ( 500 )
    DriveForward ( 100 )
})
de entrada . onButtonPressed (Button.A,  function  () {
    básica . pausa ( 500 )
    DriveForward ( 100 )
básica . pausa ( 500 )
    Girar hacia la izquierda ( 90 )
básica . pausa ( 500 )
    DriveForward ( 100 )
básica . pausa ( 500 )
dieci    Girar hacia la izquierda ( 90 )
básica . pausa ( 500 )
    DriveForward ( 100 )
básica . pausa ( 500 )
    Girar hacia la izquierda ( 90 )
básica . pausa ( 500 )
    DriveForward ( 100 )
básica . pausa ( 500 )
    Girar hacia la izquierda ( 90 )
})
MicroSecInASecond =  1000000
DistanciaPerSec =  100
NumberOfDegreesPerSec =  200
// @param grados deseados grados para girar
función  TurnLeft (grados:  número ):  void  {
```


SEXTA
CLASE

Faros automàticos

Los faros los puedes ver si estás en la costa, o en un puerto marino.
Los faros automáticos se activan a través de un sensor fotoeléctrico que está incrustado en el panel. La sensibilidad del sensor es configurada.
El sensor se activa según las condiciones de luz al amanecer o al atardecer. La sensibilidad del sensor y la demora del faro se fijan en la fábrica y no se pueden ajustar. El control automático de luz enciende las luces cada vez que el sensor, en el panel de instrumentos.
Funciona gracias a la instalación de células fotoeléctricas que miden la luminosidad exterior para gestionar en cada caso concreto el encendido y apagado de las luces de cruce.

Guía para faros automàticos

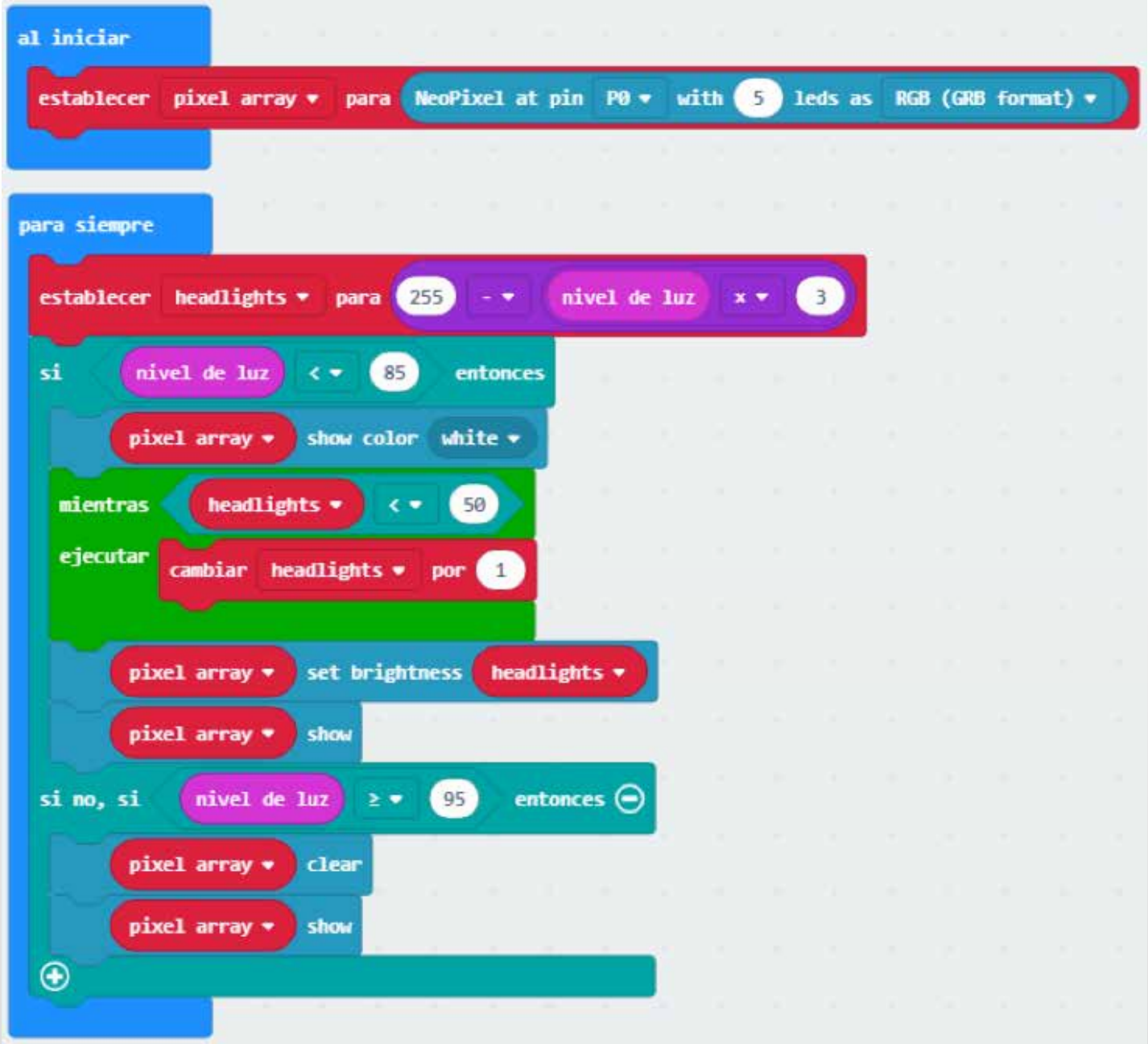
Con las noches que comienzan a oscurecerse, debemos asegurarnos de que: MOVE mini pueda continuar explorando sus alrededores mediante la codificación de algunos faros automáticos, de modo que no importa a qué hora del día o los niveles de luz ambiental, MOVE mini todavía puede encontrar su camino.

La idea era ver si podíamos usar los sensores de luz de micro: bit para detectar cuándo su entorno se vuelve más sombrío y luego usar esto para encender los LEDs de ZIP, así como para hacerlos brillar más a medida que los niveles de luz descienden.

Necesitará instalar los bloques de Neopixel en el Editor de MakeCode. Simplemente haga clic en las opciones avanzadas en el lado izquierdo del menú de codificación. En la parte inferior del nuevo panel habrá una opción para agregar Extensiones, seleccione los Neopixels en la ventana que se abre y todo está listo.

Necesitamos escribir un código que:

- Encienda y apague los LEDs ZIP y atenúelos o aclárelos según el entorno.


al iniciar
establecer pixel array para NeoPixel at pin P0 with 5 leds as RGB (GRB format)
para siempre
establecer headlights para 255 - nivel de luz x 3
si nivel de luz < 85 entonces
pixel array show color white
mientras headlights < 50
ejecutar
cambiar headlights por 1
pixel array set brightness headlights
pixel array show
si no, si nivel de luz ≥ 95 entonces
pixel array clear
pixel array show

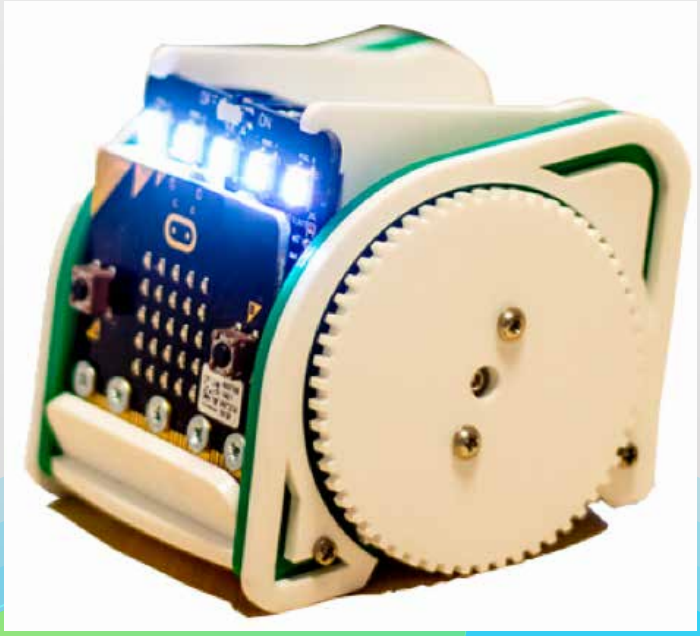
La sección de inicio está definiendo los LEDs ZIP en la placa Servo: Lite, todo agradable y sencillo aquí. También crea nuestra primera variable del código a saber "matriz de píxeles". Esta es simplemente la variable para describir los LED que estamos usando.

El bucle Forever contiene la mayor parte del código y verifica continuamente el nivel de luz y modifica la variable que hemos titulado "faros" que establecen el brillo de los LEDs de ZIP. Tanto el nivel de luz como el brillo de los LED que usamos en la matriz de píxeles se pueden configurar de 0 a 255.

Entonces, para establecer el brillo, primero tenemos el micro: bit que detecta el nivel de luz y menos esto desde 255 para alcanzar el valor de "faros". Entonces el micro: bit funciona si necesitamos tener los LED encendidos o no. La declaración de 'else if' en la parte inferior garantiza que si los niveles de luz tienen una calificación de 85 o más, los LEDs ZIP se apagarán nuevamente para toda esa bondad de ahorro de energía.

Sin embargo, si está oscuro, utilizamos el valor de los faros para ajustar el brillo de los LED. Si este brillo es tan bajo que causaría un parpadeo (alrededor de la marca 44-47), el código utiliza un bucle para garantizar que el brillo del LED esté siempre por encima de un valor establecido.

La otra cosa que podría notar es que la instrucción "else if" apaga los LED nuevamente en un umbral más alto que cuando están encendidos, esto es nuevamente para reducir el parpadeo de los LED.




SEPTIMA
CLASE

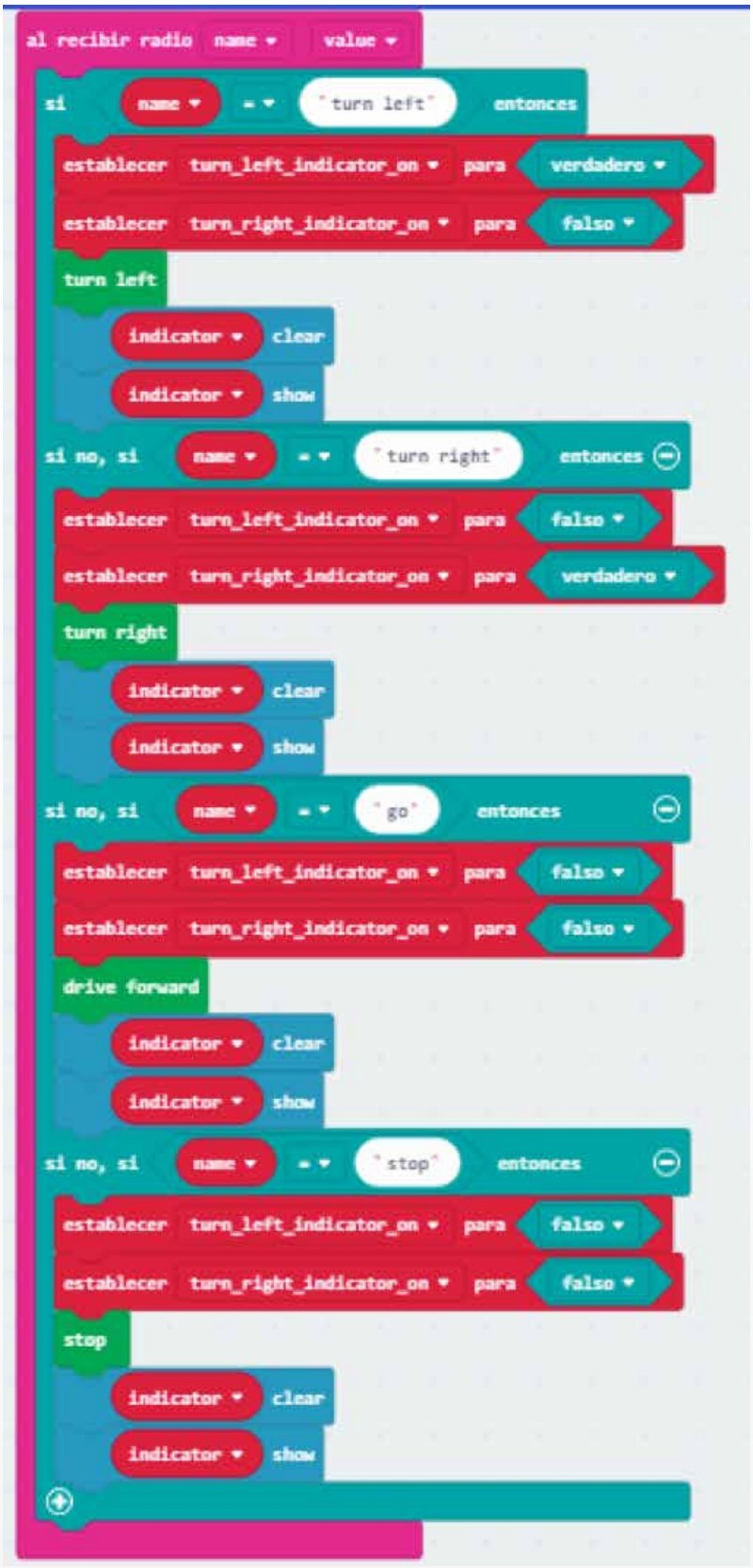
Indicadores de codificación

Un buen truco para indicar cuándo está girando. Un pequeño guiño a la seguridad vial. Una vez más estamos controlando nuestro: mini buggy con señales de radio. Esta vez estamos usando "banderas" en el código para una funcionalidad aún mejor.
Es posible que haya notado que algunos de los automóviles más recientes utilizan LED para sus luces e indicadores. Algunos, en particular, tienen indicadores "animados" que hacen girar las luces en la dirección de la próxima maniobra. Ese es el efecto que buscábamos aquí.

Que necesitaràs:

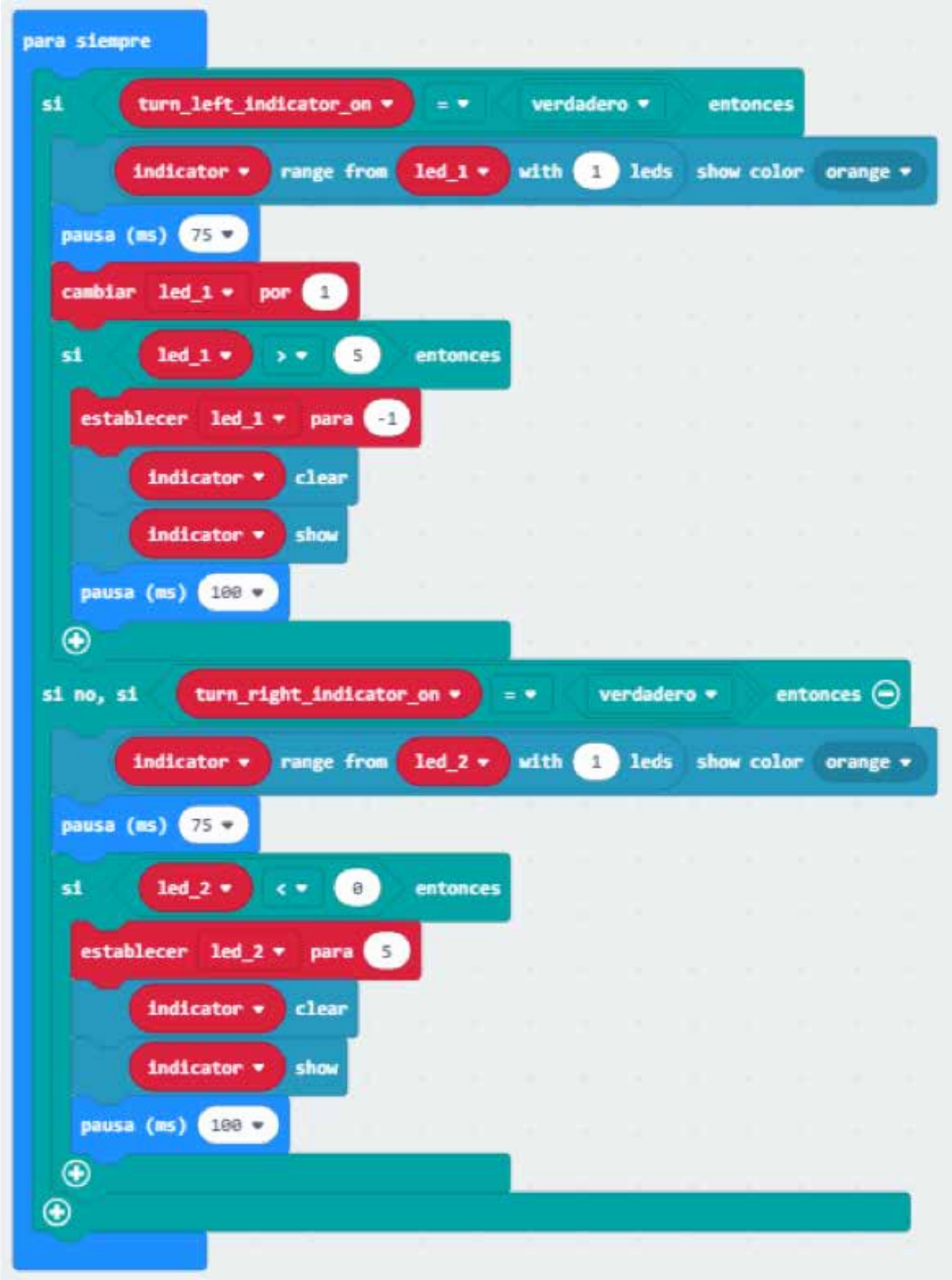
- **:mini buggy . x 1**
- **micro:bit . x 2**
- **Cable micro USB . x 1**

Programación: micro:bit – mini buggy
Este código se divide de nuevo en tres partes y se inicia. Un radio recibido y un bucle para siempre. El inicio es bastante simple simplemente configurando la ubicación de los LEDs ZIP en el: mini buggy. El brillo y el grupo de radio. Los bloques más interesantes son los otros dos ...


al recibir radio name value
si name = "turn left" entonces
establecer turn_left_indicator_on para verdadero
establecer turn_right_indicator_on para falso
turn left
indicator clear
indicator show
si no, si name = "turn right" entonces
establecer turn_left_indicator_on para falso
establecer turn_right_indicator_on para verdadero
turn right
indicator clear
indicator show
si no, si name = "go" entonces
establecer turn_left_indicator_on para falso
establecer turn_right_indicator_on para falso
drive forward
indicator clear
indicator show
si no, si name = "stop" entonces
establecer turn_left_indicator_on para falso
establecer turn_right_indicator_on para falso
stop
indicator clear
indicator show

El código de recepción de radio interpreta las instrucciones de otro micro: bit, borra todos los LEDs ZIP y crea banderas con las declaraciones de verdadero / falso. Las banderas en la codificación son como señales. Indican al resto del código que algo está en un estado particular, en este caso que: mini buggy está girando.

Cuando estamos cambiando de dirección cambiamos nuestras banderas. Por ejemplo, cuando le decimos a mini buggy que gire a la izquierda, configuramos turn_left_indicator_on en true mientras dejamos turn_right_indicator_on como falso. Cuando paramos o comenzamos a conducir hacia adelante, establecemos ambas declaraciones en falso. Esto garantiza que nuestros indicadores se apagarán y que, como máximo, solo uno de nuestros indicadores estará encendido a la vez.


para siempre
si turn_left_indicator_on = verdadero entonces
indicator range from led_1 with 1 leds show color orange
pausa (ms) 75
cambiar led_1 por 1
si led_1 > 5 entonces
establecer led_1 para -1
indicator clear
indicator show
pausa (ms) 100
si no, si turn_right_indicator_on = verdadero entonces
indicator range from led_2 with 1 leds show color orange
pausa (ms) 75
si led_2 < 0 entonces
establecer led_2 para 5
indicator clear
indicator show
pausa (ms) 100

Este conjunto final de código usa las banderas establecidas en la radio para responder si hay declaraciones. En este caso, la instrucción if pregunta si turn_left_indicator_on es verdadero? Si es así, enciende un rango definido de LED donde el LED que está encendido es igual a la variable led_1 , la otra declaración if funciona exactamente igual que led_2. El código luego usa la pausa y la función de cambio para aumentar el valor de las variables. Esto permite encender una secuencia de LEDs. Una vez que esta variable ha alcanzado un número establecido, el código restablece la variable a 0 y el indicador puede comenzar de nuevo.

La función de pausa cambia la velocidad de los indicadores y aumentar el valor en la pausa hará que el indicador sea más lento. Por el contrario, la disminución acelerará los indicadores. ¡Siéntase libre de jugar con esos valores para cambiar sus indicadores

Programación: micro:bit
Este código se divide de nuevo en cinco partes y se inicia. Un radio que envía codificación al mini buggy. El inicio es bastante simple simplemente configurando la ubicación de los giros en el micro:bit.

Estable el radio de frecuencia con el mini buggy.

Cada vez que se agite el micro:bit el mini buggy se parara y mostrará una "X" formada por las LEDs.

Cada vez que se presione el botón "A" el mini buggy girara a la izquierda y mostrara en las LEDs dos flechas indicando el lado a girar.

Cuando presiones el botón "B" de la micro:bit el mini buggy girara a la derecha y mostrara unas flechas indicando la dirección de giro.

Cuando presiones los botones "A+B" al mismo tiempo de la micro:bit el mini buggy avanzará al frente y mostrara unas flechas indicando la dirección a desplazarse.


ROBOTEACH
MASTER