Helicóptero marciano: arreglos y funciones

Grado 8° Guía 1

TIC

Estudiantes

Apoya:

Educación

BRITISH COUNCIL

Colombia Programa
{EL CÓDIGO A TU FUTURO}

Helicóptero marciano: arreglos y funciones

Grado 8° Guía 1

TIC

Estudiantes

Educación

BRITISH COUNCIL

Colombia Programa
{EL CÓDIGO A TU FUTURO}

MINISTERIO DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

Julián Molina Gómez
Ministro TIC

Luis Eduardo Aguiar Delgadillo
Viceministro (e) de Conectividad

Yeimi Carina Murcia Yela
Viceministra de Transformación Digital

Óscar Alexander Ballen Cifuentes
Director (e) de Apropiación de TIC

Alejandro Guzmán
Jefe de la Oficina Asesora de Prensa

Equipo Técnico
Lady Diana Mojica Bautista
Cristhiam Fernando Jácome Jiménez
Ricardo Cañón Moreno

Consultora experta
Heidy Esperanza Gordillo Bogota

BRITISH COUNCIL

Felipe Villar Stein
Director de país

Laura Barragán Montaña
Directora de programas de Educación, Inglés y Artes

Marianella Ortiz Montes
Jefe de Colegios

David Vallejo Acuña
Jefe de Implementación Colombia Programa

Equipo operativo
Juanita Camila Ruiz Díaz
Bárbara De Castro Nieto
Alexandra Ruiz Correa
Dayra Maritza Paz Calderón
Saúl F. Torres
Óscar Daniel Barrios Díaz
César Augusto Herrera Lozano
Paula Álvarez Peña

Equipo técnico
Alejandro Espinal Duque
Ana Lorena Molina Castro
Vanesa Abad Rendón
Raisa Marcela Ortiz Cardona
Juan Camilo Londoño Estrada

Edición y coautoría versiones finales
Alejandro Espinal Duque
Ana Lorena Molina Castro
Vanesa Abad Rendón
Raisa Marcela Ortiz Cardona

Edición
Juanita Camila Ruiz Díaz
Alexandra Ruiz Correa

British Computer Society – Consultoría internacional

Niel McLean
Jefe de Educación

Julia Adamson
Directora Ejecutiva de Educación

Claire Williams
Coordinadora de Alianzas

Asociación de facultades de ingeniería - ACOFI

Edición general
Mauricio Duque Escobar

Coordinación pedagógica
Margarita Gómez Sarmiento
Mariana Arboleda Flórez
Rafael Amador Rodríguez

Coordinación de producción
Harry Luque Camargo

Asesoría estrategia equidad
Paola González Valcárcel

Asesoría primera infancia
Juana Carrizosa Umaña

Autoría
Arlet Orozco Marbello
Harry Luque Camargo
Isabella Estrada Reyes
Lucio Chávez Mariño
Margarita Gómez Sarmiento
Mariana Arboleda Flórez
Mauricio Duque Escobar
Paola González Valcárcel
Rafael Amador Rodríguez
Rocío Cardona Gómez
Saray Piñerez Zambrano
Yimzay Molina Ramos

PUNTOAPARTE EDITORES

Diseño, diagramación, ilustración, y revisión de estilo

Impreso por Panamericana Formas e Impresos S.A., Colombia

Material producido para Colombia Programa, en el marco del convenio 1247 de 2023 entre el Ministerio de Tecnologías de la Información y las Comunicaciones y el British Council

"Esta guía corresponde a una versión preliminar en proceso de revisión y ajuste. La versión final actualizada estará disponible en formato digital y puede incluir modificaciones respecto a esta edición"

Prólogo

Estimados educadores, estudiantes y comunidad educativa:

En el Ministerio de Tecnologías de la Información y las Comunicaciones, creemos que la tecnología es una herramienta poderosa para incluir y transformar, mejorando la vida de todos los colombianos. Nos guia una visión de tecnología al servicio de la humanidad, ubicando siempre a las personas en el centro de la educación técnica.

Sabemos que no habrá progreso real si no garantizamos que los avances tecnológicos beneficien a todos, sin dejar a nadie atrás. Por eso, nos hemos propuesto una meta ambiciosa: formar a un millón de personas en habilidades que les permitan no solo adaptarse al futuro, sino construirlo con sus propias manos. Hoy damos un paso fundamental hacia este objetivo con la presentación de las guías de pensamiento computacional, un recurso diseñado para llevar a las aulas herramientas que fomenten la creatividad, el pensamiento crítico y la resolución de problemas.

Estas guías no son solo materiales educativos; son una invitación a imaginar, cuestionar y crear. En un mundo cada vez más impulsado por la inteligencia artificial, desarrollar habilidades como el pensamiento computacional se convierte en la base, en el primer acercamiento para que las y los ciudadanos aprendan a programar y solucionar problemas de forma lógica y estructurada.

Estas guías han sido diseñadas pensando en cada región del país, con actividades accesibles que se adaptan a diferentes contextos, incluyendo aquellos con limitaciones tecnológicas. Esta es una apuesta por la equidad, por cerrar las brechas y asegurar que nadie se quede atrás en la revolución digital. Quiero destacar, además, que son el resultado de un esfuerzo colectivo: más de 2.000 docentes colaboraron en su elaboración, compartiendo sus ideas y experiencias para que este material realmente se ajuste a las necesidades de nuestras aulas. Además, con el apoyo del British Council y su red de expertos internacionales, hemos integrado prácticas globales de excelencia adaptadas a nuestra realidad nacional.

Hoy presentamos un recurso innovador y de alta calidad, diseñado en línea con las orientaciones curriculares del Ministerio de Educación Nacional. Cada página de estas guias invita a transformar las aulas en espacios participativos, creativos y, sobre todo, en ambientes donde las y los estudiantes puedan desafiar estereotipos y explorar nuevas formas de pensar.

Trabajemos juntos para garantizar que cada estudiante, sin ·importar dónde se encuentre, tenga acceso a las herramientas necesarias para imaginar y construir un futuro en el que todos seamos protagonistas del cambio. Porque la tecnología debe ser un instrumento de justicia social, y estamos comprometidos a que las herramientas digitales ayuden a cerrar brechas sociales y económicas, garantizando oportunidades para todos.

Con estas guias, reafirmamos nuestro compromiso con la democratización de las tecnologías y el desarrollo rural, porque creemos en el potencial de cada región y en la capacidad de nuestras comunidades para liderar el cambio.

Julián Molina Gómez
Ministro de Tecnologías de la Información y las Comunicaciones
Gobierno de Colombia

Grado 8° Guía 1*

Guía de íconos

- Algoritmos, patrones, abstracción y descomposición
- Lógica, programación y depuración
- Prácticas de datos

Aprendizajes de la guía

Con las actividades de esta guía se espera que puedas avanzar en:

- Dividir programas complejos en funciones que dan solución a las tareas más pequeñas y luego unirlas, usarlas o reutilizarlas.
- Hacer pruebas y ajustar sus programas y algoritmos usando casos de prueba.
- Controlar la ejecución de un programa o algoritmo usando condicionales, variables numéricas y booleanas y operadores de comparación (mayor, menor, igual).

Resumen de la guía

Esta guía propone 5 sesiones de trabajo orientadas a programar la *micro:bit* mediante un lenguaje de bloques, utilizando arreglos, condicionales y operaciones entre variables con el fin de simular y validar los desplazamientos de un helicóptero usado en la exploración espacial.

Se alterna entre actividades desconectadas y conectadas usando el editor denominado *MakeCode*.

Resumen de las sesiones

Sesión 1	En esta sesión se revisa el concepto de variable en el marco de la computación. Se exploran diferentes tipos de variables.
Sesión 2	En esta sesión se explora el uso de arreglos de variables como las matrices.
Sesión 3	En esta sesión se aborda el concepto de funciones en computación y su uso en la simplificación en la programación.

* Esta guía es una adaptación de la ficha “Misión a Marte” desarrollada en 2022 por ACOFI para el programa Coding for Kids, en el marco del convenio 8/002 de 2022 suscrito entre el MEN, el Ministerio TIC y el British Council.

Aprendizajes de la guía

- Diseñar algoritmos para la solución de problemas incluyendo condicionales, variables, entradas, salidas y bucles.
- Justificar la selección de estructuras de control (diferentes tipos de bucles, condicionales) cuando se deben tomar decisiones respecto a la implementación, desempeño, calidad, tiempo y legibilidad del código.
- Recolectar y manejar datos usando funciones y arreglos de datos sencillas.
- Guardar datos en tablas y variables y hacer operaciones simples sobre estas.

Sesión 4	En esta sesión se aborda el diseño de un programa para controlar un helicóptero-robot.
Sesión 5	En esta sesión se prueba y mejora el programa desarrollado en la sesión anterior.

Conexión con otras áreas

Esta guía tiene una fuerte conexión con las ciencias espaciales y la astrobiología, aunque también retoma aprendizajes de matemáticas, ciencias naturales, y lenguaje. A continuación, se presentan algunos puntos específicos en los que se puede apreciar esta articulación temática.

Matemáticas

- Descripción y localización de la posición y trayectoria de un objeto según coordenadas cartesianas, y cálculo de promedios, valores máximos y mínimos.

Lenguaje

- Creación de organizadores gráficos que integran signos verbales y no verbales para dar cuenta de conocimientos adquiridos alrededor de un tema, y comunicación visual de datos.

Ciencias naturales

- Estudio del clima y las condiciones atmosféricas de un ecosistema.

Si se requiere

- En la guía 2 de grado 7 se comienza el trabajo con arreglos. En la guía 3 de grado 7 se trabajan funciones.

Grado 8° Guía 1

TIC

Sesión 1

Aprendizajes esperados

Al final de esta sesión verifica que puedas

- Definir variables, almacenar datos y utilizar estos datos en cálculos.
- Reconocer tipos de variables.

Duración sugerida

40% 40% 20%

Material para la clase

- ○ Dispositivo con acceso a *MakeCode*
- ○ Copia del Anexo 1.1

Anexo

Anexo 1.1

El Mars Helicopter o helicóptero marciano es un proyecto experimental de la NASA adelantado entre 2022 y 2024. El proyecto consiste en desplegar un pequeño helicóptero autónomo en Marte, de ahí su nombre. La finalidad de este helicóptero-robot es explorar diferentes áreas de Marte. Tal como lo señala la NASA, este pequeño helicóptero funciona como explorador robótico de prueba para obtener información relevante, con el fin de evaluar y planear misiones futuras de otros robots en Marte. Las baterías y el helicóptero mismo deben ser trasladados en una nave espacial interplanetaria hasta ese planeta. Dada la gran distancia desde la Tierra, todo lo que se transporte debe ser del tamaño y peso más pequeño posible. Por ello, este helicóptero es muy pequeño y tiene varias restricciones:

- Fue diseñado inicialmente para realizar máximo 5 vuelos en Marte.
- Cada vuelo no puede durar más de 90 segundos, esto es, minuto y medio.
- Se puede desplazar en un área definida por una cuadrícula de 5 por 5 donde la posición en la parte de arriba a la izquierda es la posición 0,0 y la posición abajo a la derecha es la 4,4.
- Cada movimiento (despegar, aterrizar, realizar un movimiento en X o un movimiento en Y) le toma 10 segundos. En consecuencia, cada vuelo sólo puede tener máximo 10 de estos movimientos, y regresar al punto de partida.

En la micro:bit crea un programa que permita validar un plan de vuelo que cumpla con los siguientes requerimientos:

- El helicóptero inicia en las coordenadas (0,0)
- El plan de vuelo se introducirá usando los botones:
  - Al presionar el Botón A incrementa o disminuye la coordenada X.
  - Al presionar el Botón B incrementa o disminuye la coordenada Y.
  - Al presionar los botones A y B a la vez (Botón A+B) las funciones de los botones A y B cambian entre incrementar y disminuir ambas coordenadas X y Y. Aunque estas funciones siempre deben comenzar incrementando.
- La posición del helicóptero debe mostrarse en la pantalla.
- Cuando el helicóptero se salga del espacio de trabajo, la pantalla debe parpadear para indicar el error.
- Cuando se hayan hecha más de 10 movimientos sin regresar al origen, la pantalla debe indicar una X.
- En cada posición, el helicóptero toma un dato de nivel de luz.
- Al regresar al punto de partida (coordenadas 0,0) se presenta en pantalla el promedio de las mediciones de nivel de luz.
- Se reinicia todo agitando la micro:bit.

Lo que sabemos, lo que debemos saber

Esta sección corresponde al 40% de avance de la sesión

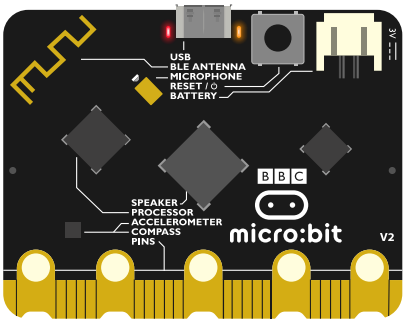
Como probablemente ya sabes, programar es describir en detalle una serie de pasos lógicos en un lenguaje apropiado que el computador interpreta y ejecuta para realizar una tarea.

Tendrás en esta guía la posibilidad de trabajar sobre algunos aspectos de lo que implica la programación de un robot como el helicóptero marciano, *Mars Helicopter*. Este helicóptero-robot fue un proyecto experimental de la NASA. El proyecto consistía en desplegar un pequeño helicóptero autónomo en Marte, de ahí su nombre. La finalidad del helicóptero-robot era explorar diferentes áreas de Marte. Tal como lo señaló la NASA, este helicóptero funcionaría como explorador robótico de prueba para obtener información relevante, con el fin de evaluar y planear misiones futuras de otros robots en Marte.

Las baterías y el helicóptero mismo deben ser trasladados en otra nave espacial interplanetaria hasta Marte. Recuerda que, dada la gran distancia, todo lo que se transporte debe ser del tamaño y peso más pequeño posible. Por ello, este helicóptero-robot es tan pequeño que solo puede realizar máximo 5 vuelos en Marte con una duración máxima de 90 segundos cada uno, con despegues y aterrizajes verticales controlados. Dale una lectura al *Anexo 1.1* que resume el reto que te proponemos.

Teniendo en cuenta las características del helicóptero-robot y las actividades que vas a realizar, deberás comprender algunos aspectos que se irán presentando a lo largo de esta guía. En particular debes saber qué es una **variable**, qué es una función y qué son casos de prueba en programación.

Figura 2. Tarjeta *micro:bit*, vista posterior



La *micro:bit*, así como otras tarjetas como Arduino o Raspberry poseen una memoria donde se guardan diferentes tipos de datos.

Examina la *Figura 1*:

Figura 1. Casillero con espacios de almacenamiento

La memoria de un computador (por ejemplo, la *micro: bit* es un pequeño computador) se puede imaginar como un cajonero como el que muestra la *Figura 1.* Cada cajón tiene un nombre o etiqueta que nos permite identificarlo sin equivocarnos, y dentro de cada cajón puedo guardar, retirar, o consultar un objeto; en el caso de las variables, un dato.

Cuando se programa la *micro:bit* con *MakeCode*, por ejemplo, una variable puede guardar un valor, como se muestra en la *Figura 3*.

Este valor puede ser de diferente tipo. Estos son los 3 tipos de variable que se usarán en esta guía:

- Texto
- Booleano
- Numérico

Sin embargo, como observas en la siguiente imagen, además de estos, *MakeCode* permite trabajar con otros dos tipos de variables que no abordarás aún:

Todo tipo de dato en un computador se representa siempre en números, y no de cualquier tipo, sino específicamente en números binarios o en base 2. Por ejemplo, el número 122 se escribe en binarios como:

1111010

MakeCode cuenta con herramientas de accesibilidad como alto contraste y funciona con lectores en pantalla. Consulta a tu docente si necesitas apoyo para hacer uso de estas opciones.

Cada dígito es denominado ***bit***. Es la interpretación del número la que define el tipo de variable.

Por ejemplo, el número binario anterior podría representar un número entero, el 122, o simplemente una letra, la "z", usando una codificación de las letras en números conocido como **Código ASCCI**.

En la *Figura 3* puedes ver una representación de cuatro variables de tipos diferentes, con ejemplos de valores.

Figura 3. Variables con valores

Variable	Valor
T1	21
T2	1
Llueve	Falso
Nombre	"Cali"

- La variable T1 tiene asignado el valor numérico 21.
- La variable T2 tiene asignado el valor numérico 1.
- La variable Llueve tiene asignado el valor Booleano Falso, significa que no está lloviendo.
- La variable Nombre tiene asignado el valor alfabético "Cali".

Se pueden realizar operaciones con los valores almacenados en las variables y actualizarlos. Por ejemplo, es posible sumar al valor de T1 con el valor de T2 y guardar el resultado de la suma en T2 , como se ve en la *Figura 4*.

Figura 4. Operaciones con variables

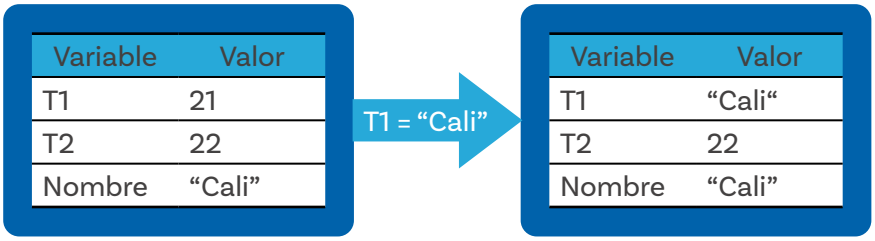
Variable	Valor
T1	21
T2	1
Nombre	"Cali"

T2 = T1 + T2

Variable	Valor
T1	21
T2	22
Nombre	"Cali"

Se puede cambiar el valor y, en *MakeCode*, además, el tipo de valor almacenado en una variable. Por ejemplo, es posible cambiar el valor asignado a T1 por el valor "Cali", tal como se observa en la *Figura 5*.

Figura 5. Cambios de valores en variables



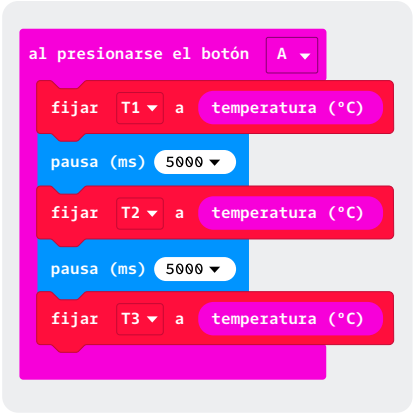
Glosario

- **Variables:** En programación, una variable es un espacio en la memoria donde se almacena un valor que podemos crear, cambiar, leer y usar durante la ejecución de un programa.
- **Memoria:** Lugar donde un computador guarda la información.
- **Bit:** Unidad mínima de memoria en un computador, que puede almacenar un "0" o un "1". Corresponde a cada dígito binario.
- **Variable Booleana:** Asume dos posibles valores, "1" para cierto, "0" para falso.
- **Variable de texto:** Cada letra se guarda como un número usando el código ASCCI.
- **Código ASCII:** Define un conjunto de 128 caracteres, incluyendo las letras mayúsculas y minúsculas del alfabeto inglés, números, símbolos de puntuación y algunos caracteres de control.

  Cada caracter ASCII se representa con un código de 7 bits, lo que significa que se utilizan 7 dígitos binarios (0 o 1) para representar cada caracter.

Figura 6. Bloques de variables en *MakeCode*

Figura 7. Código para tomar 3 medidas de temperatura



Código para medir 3 veces los valores de temperatura y almacenarlos en variables

Manos a la obra
Conectadas

Esta sección corresponde al 80% de avance de la sesión

Es el momento de practicar con las variables. Organízate en parejas según las instrucciones de tu docente.

Ingresen a *MakeCode*. Para crear variables este editor utiliza el menú de bloques de variables, así que diríjanse allí. Luego, elijan la opción crear una variable. Creen tres variables llamadas: T1, T2 y T3, como se muestra en la *Figura 6*. Utilizarán estas tres variables para guardar la temperatura medida por la *micro:bit* en tres momentos diferentes.

El helicóptero marciano tiene varios sensores que miden diferentes variables físicas de las rutas como la temperatura (termómetro), la presión (barómetro), la orientación (brújula), etc.

Programarán el helicóptero marciano para que recoja la temperatura promedio de cada punto en su vuelo. Esta información será útil para conocer más sobre la atmósfera de este planeta.

Los sensores, en particular los termómetros, pueden arrojar imprecisiones en las medidas. Es común tomar varias mediciones y promediarlas para reducir este error.

Recuerden que el cálculo del promedio se debe realizar con la formula presentada en la *Figura 8*.

Figura 8. Formula de promedio

$$\frac{(T1+T2+T3)}{3}$$

Examinen ahora el programa en bloques que aparece en la *Figura 7*. Traten de anticipar qué hace y luego pruébenlo para verificar su predicción.

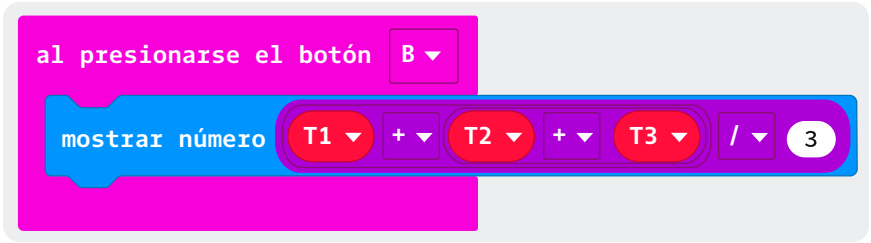
Figura 9. Bloques de Entrada en *MakeCode*. Medir nivel de luz

¿Cómo podrían verificar los valores que quedaron en las variables?

Igualmente, tengan cuidado al ensamblar la parte del cálculo del promedio en su código.

Examinen este fragmento de programa presentado en la *Figura 10* para complementar el anterior:

Figura 10. Formula promedio en *MakeCode*



Analicen con cuidado el orden en que se hacen las operaciones en este bloque que calcula el promedio de las tres variables.

Figura 11. Formula promedio para tres variables

$$\text{promedio} = \frac{((T1+(T2+T3))}{3}$$

Si usan otro orden, ¿obtendrán un resultado incorrecto? Inténtenlo.

Ahora, programen la *micro:bit* en *MakeCode* para que simule la toma de datos que haría el helicóptero marciano. Asegúrense de que su programa:

- ○ Cuente con las variables necesarias.
- ○ Comience a tomar datos de nivel de iluminación cada minuto cuando se oprima el botón A (para esto pueden usar el comando que se muestra en la *Figura 9*).
- ○ Tome 5 datos.

- ○ Calcule el promedio de niveles de iluminación registrados.
- ○ Al final muestre este promedio.

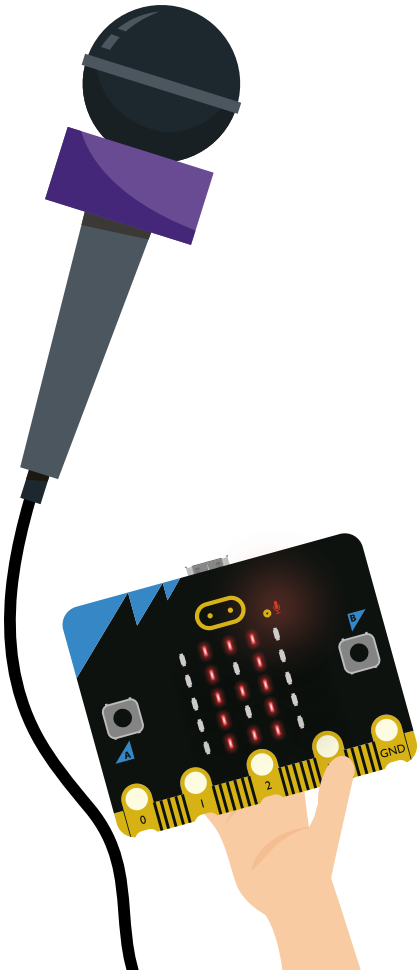
Los dos fragmentos de programa les serán de utilidad.

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión. ¿Crees que lograste alcanzarlos?

1. ¿Puedes definir variables, almacenar datos y utilizar estos datos en cálculos?
   - ○ Sí
   - ○ Parcialmente
   - ○ Aún no
2. ¿Puedes reconocer tipos de variables?
   - ○ Sí
   - ○ Parcialmente
   - ○ Aún no



Si tus respuestas a las preguntas anteriores fueron "Parcialmente" o "Aún no", revisa nuevamente los contenidos y consulta las inquietudes que todavía tienes con tu docente.

Una vez aclaradas esas dudas, trata de resolver por ti misma(o) el siguiente reto:

Se está realizando una obra cerca a la casa de unos familiares tuyos, el ruido del constante martilleo y la carga y descarga de elementos de construcción se suma a los gritos de niñas y niños que juegan en un parque, y al megáfono de personas vendendoras ambulantes, entre otros. Tú has aprendido que es importante cuidar los oídos, por tanto, has decidido verificar si los niveles de ruido a los que tus

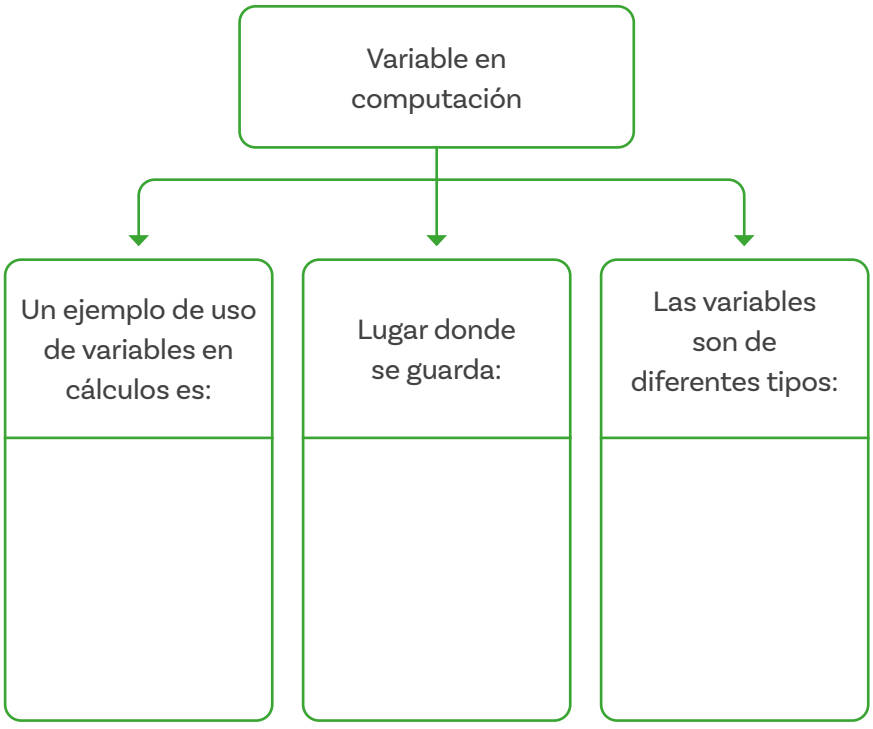
familiares se exponen cada día podrían ser de riesgo. Con tal fin, has decidido crear un programa que, con ayuda de la micro:bit, capture unas cuatro veces, de forma automática, el sonido del entorno si se presiona el botón A. Las mediciones deben realizarse a intervalos de 15 segundos. Al finalizar, tu programa deberá indicarte el valor promedio obtenido durante la toma de datos.

¡Este ejercicio te ayudará a consolidar los aprendizajes!

Ahora piensa en lo siguiente. En el ejercicio realizado se tomaron datos de temperatura.

¿Se podría hacer lo mismo sin variables?
¿Qué ventajas tiene el uso de variables?
¿Qué parte del reto podrías resolver con lo que has aprendido hasta ahora?

Aprovecha este espacio para que resumas qué entendiste completando este esquema:



Sesión 2

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

- Crear arreglos en un lenguaje de bloques.
- Guardar valores en arreglos.
- Utilizar variables booleanas.
- Realizar operaciones con los valores en arreglos y variables.

Material para la clase

- ○ Acceso a *MakeCode*

Duración sugerida

40% 40% 20%

Figura 1.

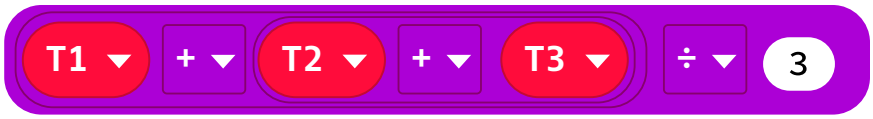
Código para hacer 4 mediciones de temperatura y almacenar los registros en variables

Lo que sabemos, lo que debemos saber

Esta sección corresponde al 40% de avance de la sesión

Variables que solo pueden guardar un valor son un gran limitante dado que, si es necesario realizar operaciones con ellas, el programa puede resultar complejo y muy largo.

Retoma este fragmento de código de la sesión anterior:



En este caso teníamos sólo 3 variables que al definirlas las llamamos T1, T2 y T3.

Imaginemos que el helicóptero debe tomar un número mayor de datos. Por ejemplo, una medida cada 5 minutos durante un día completo, para luego encontrar el valor mínimo, el valor máximo y el promedio. Serían en total 12 medidas por hora y dado que son aproximadamente 24 horas por día (realmente algo más en Marte), el número total de mediciones que tendremos que almacenar es de 288.

Difícil, aunque no imposible, pensar en:

- Crear 288 variables que van desde T1 hasta T288.
- Escribir 288 instrucciones guardando en forma secuencial los valores en variables cada 5 minutos.
- Luego realizar el cálculo de todos los valores (T1+ ...+T28) lo cual da una instrucción que ocuparía muchas pantallas.

Un pedazo muy pequeño del código parecería al código de la *Figura 1* pero muchísimo más largo; no cabría en esta página.

En computación este problema se resuelve usando una sola variable en la que se puedan guardar muchos datos, a la que se conoce como ***arreglo***, en la que se puede acceder a cada espacio de almacenamiento con un número o **índice**.

Nota

Nota: Cuando se describe un algoritmo, para asignar un valor a una variable se usan diferentes notaciones.

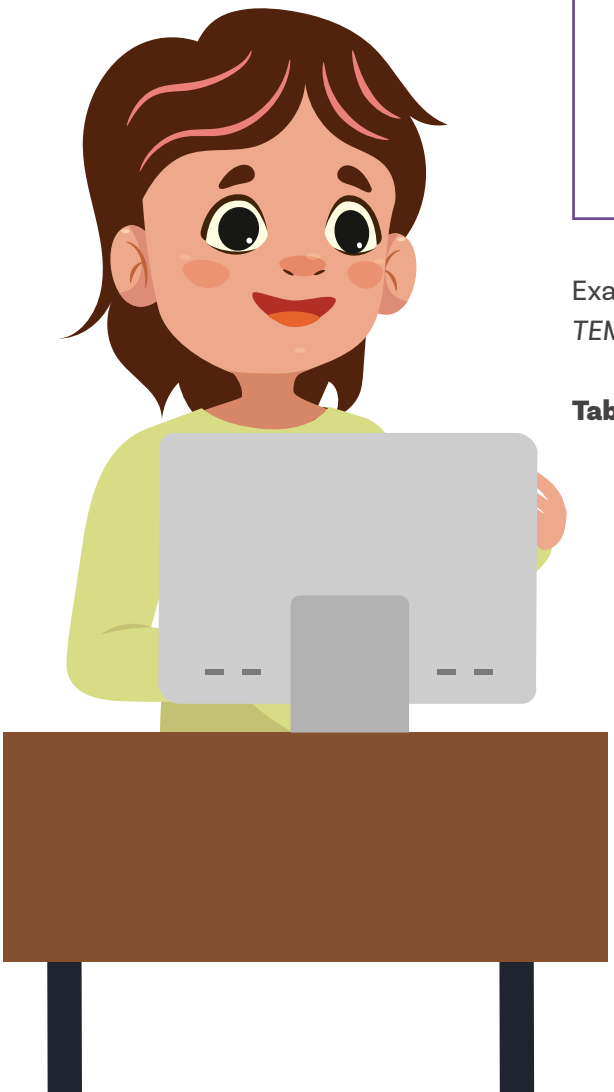
- SUMA = SUMA+2
- A SUMA asignar SUMA+2
- SUMA <- SUMA+2
- SUMA ← SUMA+2
- Incrementar SUMA en 2

Las 5 significan: asignar a la variable SUMA el resultado de lo que había en SUMA más 2.

Es de anotar que una expresión como:

SUMA = SUMA+2

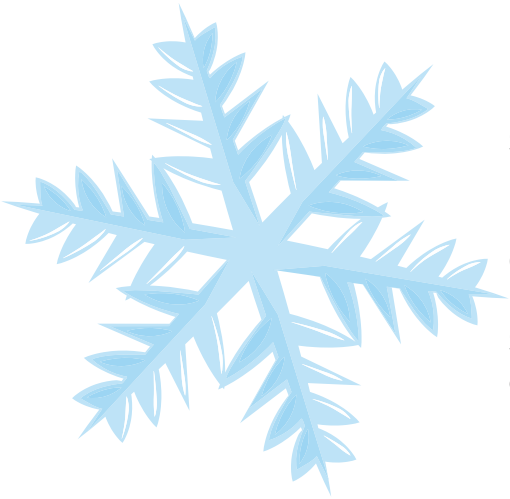
Es incorrecta en matemáticas porque SUMA no puede ser SUMA+2, por ello se prefieren las otras formas de representar la asignación de un valor a una variable. Se usará la flecha ← en lo que se denomina pseudocódigo o programa en palabras.



Examina la tabla 1 que busca representar un arreglo llamado *TEMPERATURA*, que puede guardar 10 valores:

Tabla 1. Arreglo / Temperatura

	T
T(0)	28,7
T(1)	32,5
T(2)	33,4
T(3)	31,2
T(4)	30,0
T(5)	28,7
T(6)	25,2
T(7)	22,8
T(8)	23,5
T(9)	25,1



Esta variable *(T)* es un arreglo que tiene 10 campos, rotulados de 0 a 9, para guardar o recuperar la información de temperatura de 10 valores.

Si quisiéramos escribir un valor en la casilla 3 diríamos:

T(3)=31,2

Que es justo el valor que se encuentra en este ejemplo.

Si quisiéramos sumar la temperatura de la casilla 3 con la de la 5 escribiríamos:

SUMA ← T(3)+T(5)

Podría pensarse que si hubiésemos hecho la suma de los 288 valores tendríamos que hacer algo como:

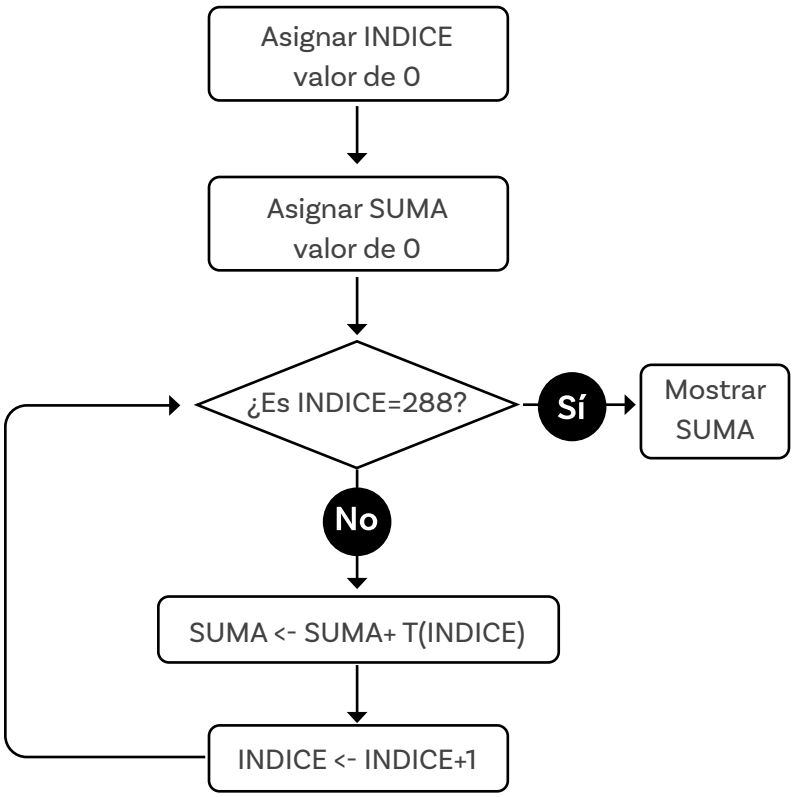
SUMA ← T(1)+T(2)+...T(288)

Lo que no resulta más fácil que escribir:

SUMA ← T1+T2+...+T288

Sin embargo, poder referirte a una posición específica en el arreglo de variables con un número permite crear un algoritmo, tanto escrito en palabras como en diagrama de flujo, como se ilustra en la *Figura 2*, para sumar en pocas instrucciones, incluso un millón de datos:

Figura 2. Algoritmo para desplazarse por un arreglo e ir sumando los valores almacenados



INDICE ← 0
SUMA ← 0

Para INDICE de 0 a 287 veces
SUMA ← SUMA+T(i)
Verificar veces
Mostrar SUMA/288

Nota

Para INDICE de 0 a 287 veces es una instrucción que indica que se vaya variando la variable **INDICE** en 1 desde 0 hasta 287 y que se vaya ejecutando las instrucciones que encierra la instrucción **Para**. Aquí la variable **INDICE** se usa en este ejemplo para acceder a cada posición de la variable T.

El acceso a arreglos está en el menú Avanzado:

Se debe crear primero la variable *T* en el menú de variables para luego usarla como un arreglo al usar el comando.

Luego para asignar a una posición específica del arreglo *T* un valor en la posición *index* se utiliza:

Para recuperar el valor guardado en la posición *index* se usa el comando:

Y existe la ventaja de que se puede modificar el valor en la variable *index*.

Deberás tener antes el arreglo *T* con las 288 mediciones que luego serán utilizadas para realizar la suma de todos los elementos y dividir por el número de datos para encontrar el promedio.

Examina el código presentado en la *Figura 3*. Verás que se usa el bloque *al iniciar* que se ejecuta al comienzo, una sola vez, creando el arreglo *T*.

> *¿Qué sucede cuando se oprime A?*
> *¿Y cuando se oprime B?*

Figura 3. Creación de arreglo con *MakeCode*

Nota

Para definir una variable booleana se usa el bloque:

La variable *Datos_listos* quedó definida como booleana con el valor de falso.

Esta variable podrá cambiarse a verdadero con la misma instrucción anterior.

Pruébalo luego.

Utiliza estos 3 casos de prueba:

- ◯ Oprimir el botón B sin haber oprimido el botón A.
- ◯ Oprimir el botón A e inmediatamente el botón B.
- ◯ Oprimir el botón A, esperar el ok y oprimir el botón B.

Notarás que desafortunadamente los dos primeros casos de prueba no funcionan bien, dado que al oprimir B muy rápido después de A o sin oprimir A, los datos no están listos.

Una solución que nos propone una persona es la siguiente:

Usar una variable lógica o booleana que mientras se toman datos esté en falso y cuando ya se haya terminado de tomar los datos pase a verdadero. A su vez, el bloque de oprimir B solo debería funcionar cuando la variable de toma de datos esté en verdadero.

A menudo el funcionamiento de los programas se evalúa con diferentes datos que se denominan casos de prueba.

Un ejemplo de la estructura del bloque de oprimir B podría ser:

Como verás, si la variable *Datos_listos* no está en verdadero, no se realizará el cálculo y se mostrará una X.

Glosario

Arreglo: Conjunto de espacios de memoria en el que se pueden guardar diferentes variables a las que se puede acceder indicando el número del campo o posición que ocupan dentro de esta estructura. Por ejemplo, Temperatura[5] hace referencia al campo o espacio de almacenamiento 5 de un arreglo llamado Temperatura.

Índice: Identificador del campo o espacio de almacenamiento en que se ubica un valor dentro de un arreglo.

Manos a la obra Conectadas

Esta sección corresponde al 80% de avance de la sesión

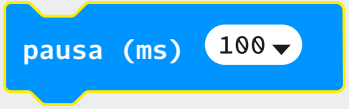
Organízate en grupos siguiendo las indicaciones de tu docente.

Es momento de resolver el siguiente reto:

Programen la micro:bit *para que simule la captura de datos de temperatura realizada por el helicóptero marciano a lo largo de un día, realizando una medición cada 5 minutos. Los datos recogidos se utilizarán posteriormente para calcular la temperatura máxima y la mínima o el promedio.*

Utilicen el código anterior y modifíquenlo. No realicen las mediciones cada 5 minutos, pues deberían esperar un día completo para contar con todos los datos. Tomen datos cada 100 ms usando el comando.

Figura 4. Bloque de pausa



De esta forma, los 288 datos se tomarán en 288 veces 0.1 segundos lo cual es menos de un minuto.

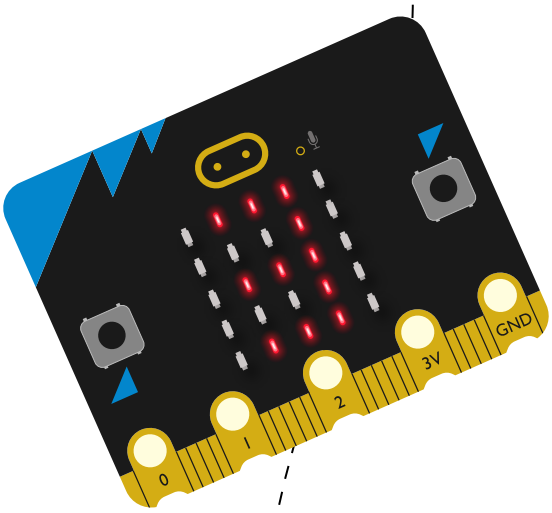
¡Manos a la obra!

Verifiquen que su código funciona bien y que evita generar cálculos si los datos no están completos:

¿Y cómo harían un programa que haga lo mismo con el nivel de iluminación usando el comando `nivel de luz` *tomando datos cada media hora durante 24 horas?*

Se sugiere que 20 minutos se escalen a 100 ms.

¿Cuánto durará la toma de datos de un día si se asume que 30 minutos son 100ms?



Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión. ¿Crees que lograste alcanzarlos?

1. ¿Puedes crear arreglos en un lenguaje de bloques?
   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no
2. ¿Puedes guardar valores en arreglos?
   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

3 ¿Puedes utilizar variables booleanas?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

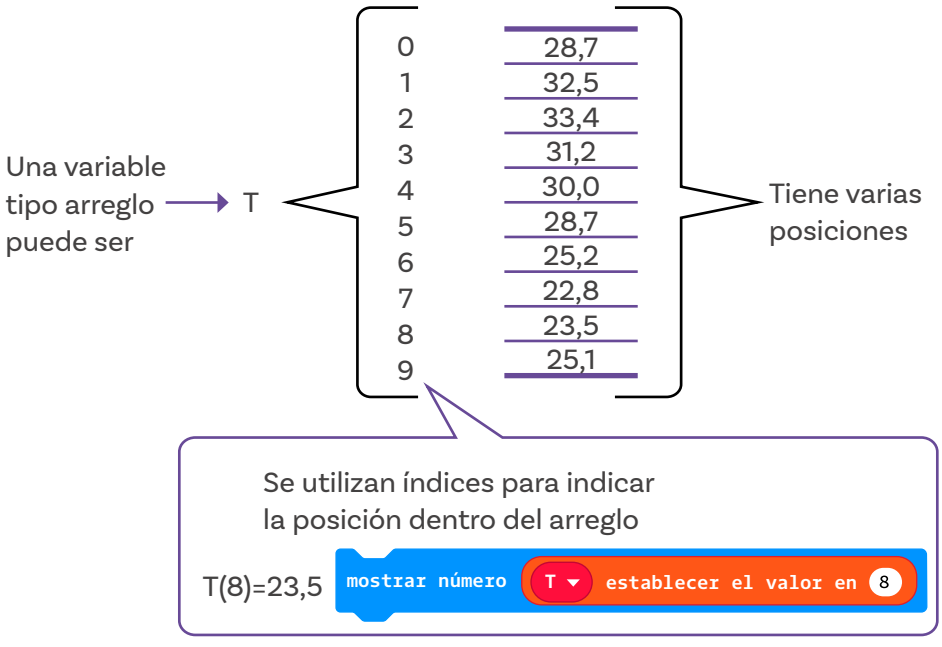
4 ¿Puedes realizar operaciones con los valores en arreglos y variables?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

Si tus respuestas a las preguntas anteriores fueron "Parcialmente" o "Aún no", revisa nuevamente los contenidos de esta sesión y consulta con tu docente las inquietudes que todavía tienes sobre los temas vistos.

Piensa, igualmente, en qué parte del reto puedes resolver con lo que has aprendido. Poder trabajar con arreglos simplifica la programación y permite manejar grandes cantidades de datos usando muy pocas instrucciones.

Completa este esquema con los aprendizajes logrados durante esta sesión:



Grado 8° Guía 1

TIC

Sesión 3

Aprendizajes esperados

Al final de esta sesión verifica que puedas

- Definir y utilizar funciones para simplificar la escritura de programas.
- Utilizar arreglos para guardar datos y hacer cálculos.
- Usar casos de prueba para verificar el funcionamiento correcto de un algoritmo.

Duración sugerida

40% 40% 20%

Material para la clase

- ◯ Acceso a *MakeCode*

Lo que sabemos, lo que debemos saber

Esta sección corresponde al 40% de avance de la sesión

En esta sesión conocerás o profundizarás en el uso de funciones, una herramienta para programar de forma clara, más sencilla y estructurada.

Cuando hablamos de funciones nos referimos a un conjunto de instrucciones o líneas de código que realizan una tarea específica y de forma frecuente. Al crear una función a este conjunto de instrucciones se le da un nombre que lo identifica y lo consolida como una nueva instrucción que puede llamarse y ejecutarse cuando se requiera dentro de otro programa.

En la sociedad actual, existen expertos en múltiples actividades, lo que facilita adelantar proyectos complejos. Por ejemplo, si una persona está remodelando una casa probablemente no haga todo el trabajo ella misma, sino que llame a personas expertas en pintar, en colocar pisos, o en hacer instalaciones eléctricas. Cada una de estas personas sabe qué hacer y cómo hacerlo.

En programación se aplica la misma idea de especialización: si un conjunto de instrucciones realiza una tarea especializada se convierte en una función. A las funciones se les asigna un nombre por medio del cual se pueden realizar todas las instrucciones, cada vez que se requiera, mediante una llamada.

Así, una función permite hacer una tarea varias veces sin repetir las líneas de código necesarias para realizarla.

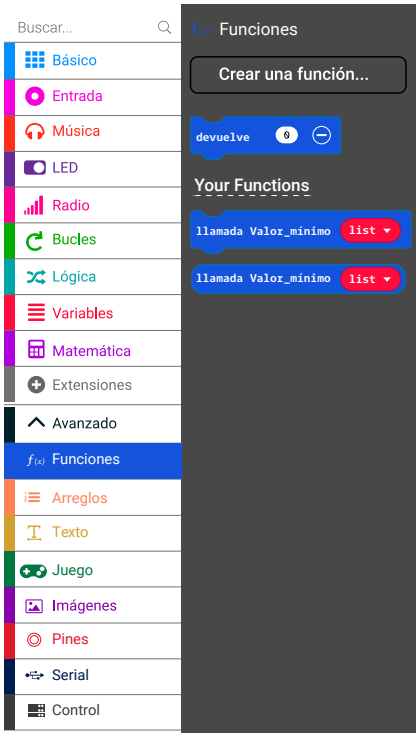
En *MakeCode* cada bloque que usas para programar es, de hecho, una función y los puedes combinar para crear nuevas funciones.

Examina el siguiente ejemplo.

Dado un arreglo que contiene datos, se quiere encontrar el valor más grande del conjunto de datos almacenados en él.

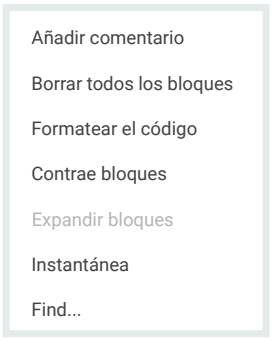
Figura 1. Bloques de funciones en *MakeCode*

En Avanzado, encontrarás el menú para crear funciones y las instrucciones que se requieren.



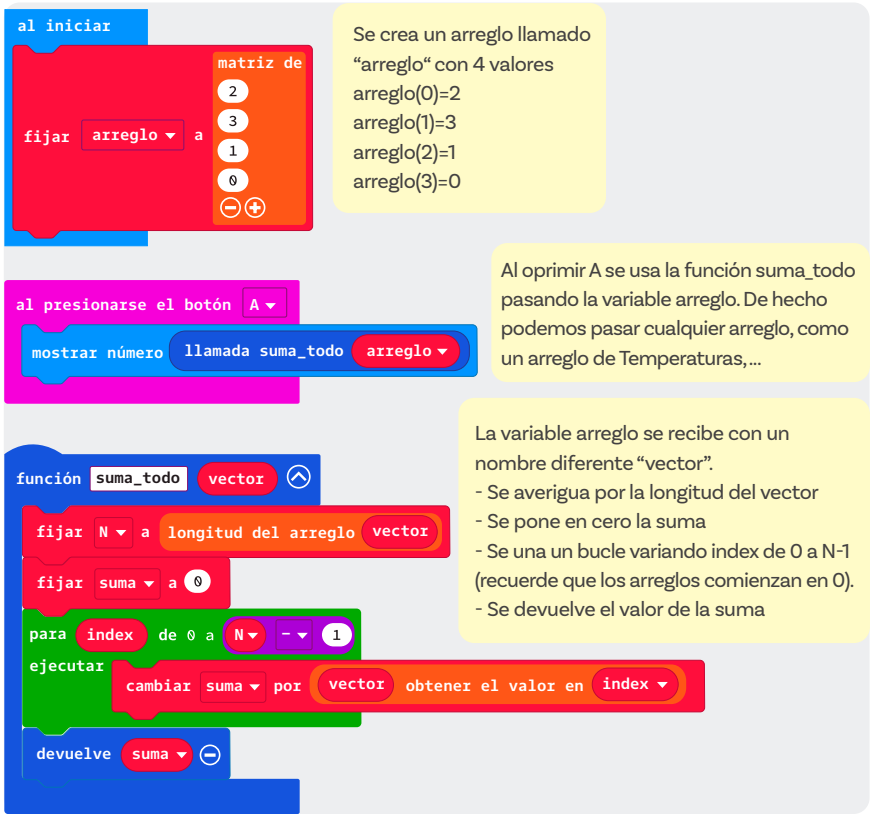
Nota

Para documentar un código oprime el botón derecho del ratón y aparecerá la opción de *Añadir comentario*.



Para ello examina el código siguiente:

Figura 2. Programación de arreglo en *MakeCode*



Este código tiene notas que facilitan entenderlo. A esto se le llama **documentar el código**.

Examina el código de abajo que es la función *suma_todo*. Para crear una función examina la *Figura 2*. Esta función recibe un arreglo al que se le coloca el nombre de *vector* con el que se procede a:

- Averiguar su longitud y guardarla en una variable llamada N. Colocar una variable "suma" en cero.
- Repetir mientras una variable "index" va de cero a N-1.
  Suma ← suma + vector(index)
- Al terminar devuelve el valor que se ha calculado en la variable *suma*.

Prueba este programa. No dudes en cambiar los valores del arreglo, agregar más *incluso* y verificar que la suma es siempre correcta.

Nota

En una función se pueden recibir parámetros, como en este caso que se le ha denominado vector y que aparece en la definición de la función:

Al usar la función, se podrá pasar cualquier arreglo, que será copiado en este parámetro vector para uso interno de la función. En este caso, se pasa a la función MAX el arreglo *Temperaturas*:

Manos a la obra

Conectadas

Esta sección corresponde al 80% de avance de la sesión

Vas a trabajar en parejas, siguiendo las instrucciones de tu docente. Ingresen a *MakeCode*, teniendo abierta una copia del programa para sumar los valores almacenados en un arreglo.

Es el momento de hacer algo más interesante:

Modifiquen el programa anterior para que encuentre el valor máximo almacenado en el arreglo. Utilicen el siguiente algoritmo en la función:

- N ← longitud(vector)
- Max ← 0
- Repetir de *index* a N-1.
- Si Max < vector(index) entonces Max=vector(index)
- Devolver "suma".

Primero prueben este algoritmo usando lápiz y papel. Escriban las variables y sus valores, vayan cambiando los valores en la medida en que siguen el algoritmo anterior.

Después de que hayan verificado el funcionamiento de este algoritmo, examinen el código que lo programa en una función:

Figura 3. Programación de función en *MakeCode*

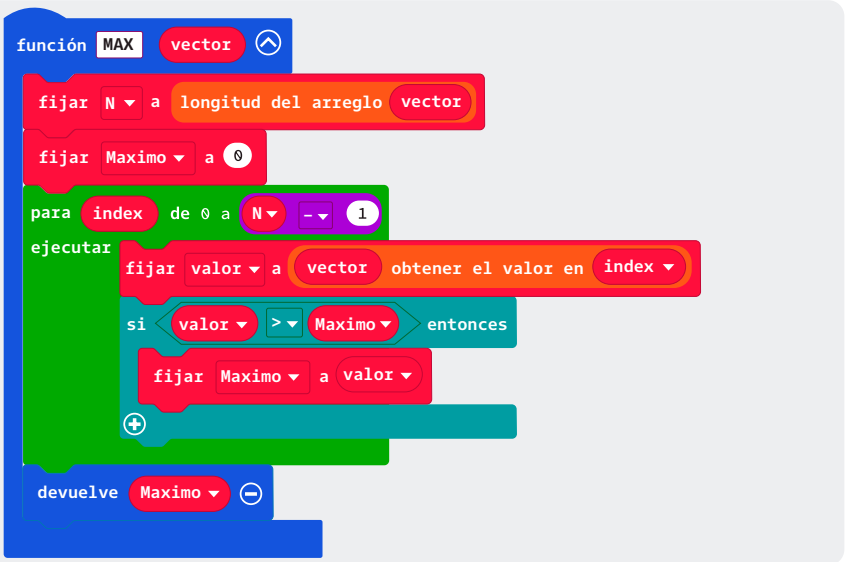


Figura 4. Llamado a la función MAX y su parámetro el arreglo Temperatura

Nota

Si los datos usados para probar fueran:

-2
-10
-5
-20

¿Funciona el algoritmo?

Hagan el mismo ejercicio de seguir paso a paso, con papel y lápiz, el funcionamiento de este código. Luego completen el programa con el código de la *Figura 4*.

Notarán que en este caso se hace uso de un arreglo que contiene temperaturas medidas, a las que, por ahora, se le asignaron de forma arbitraria los valores 20, 35, 19 y 18. A estos valores se les denominan **casos de prueba**.

Si todo funciona bien, al oprimir el botón B deberían ver en la pantalla el valor 35 que corresponde a la temperatura más alta almacenada en el arreglo.

Es importante que se den cuenta de que esta función llamada "max" podrá encontrar el valor más grande en cualquier arreglo, incluso entre millones de datos almacenados.

El reto que tienen ahora es reemplazar los valores asignados arbitrariamente al arreglo Temperatura por temperaturas reales tomadas con la *micro:bit*, usando el comando respectivo de entrada.

Si necesitan recordar cómo hacer esta captura de datos, pueden revisar lo visto en la sesión anterior donde se capturaron y sumaron todos los valores obtenidos.

Ahora se quiere encontrar la temperatura máxima en un período de tiempo dado, a partir de mediciones tomadas de forma regular. El reto es, por tanto, el siguiente:

Crear con la micro:bit *un programa que simule la captura de datos de temperatura realizada por el helicóptero marciano. La toma de datos de temperatura debe hacerse cada 10 milisegundos y los valores obtenidos deben almacenarse en un arreglo. Se deben tomar 1.000 mediciones. El programa debe encontrar la máxima temperatura en el arreglo. Esto significa que se requiere de menos de un minuto para la toma de todos los datos.*

Glosario

Documentar el código: Es la práctica de agregar comentarios a un programa para explicar cómo funciona. Esto ayuda a cualquier persona, incluso si no participó en la creación del código, a comprender mejor qué información se guarda en las variables y qué hacen las funciones.

Caso de prueba: Son un conjunto de condiciones, acciones, y entradas específicas (datos) que se utilizan para verificar si un programa funciona como se espera, es decir,
si cumple.

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión. ¿Crees que lograste alcanzarlos?.

1. ¿Puedes definir y utilizar funciones para simplificar la escritura de programas?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

2. ¿Puedes utilizar arreglos para guardar datos y hacer cálculos?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

3. ¿Puedes usar casos de prueba para verificar el funcionamiento correcto de un algoritmo?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

Si tus respuestas a las preguntas anteriores fueron "Parcialmente" o "Aún no", puedes revisar nuevamente los contenidos de esta sesión, tratando de seguir paso a paso las instrucciones que se ofrecen. Mientras lo haces, toma nota de las preguntas o dificultades que te surjan, y consulta con tu docente para aclararlas o para realizar las actividades adicionales que te proponga.

Ahora, completa las siguientes frases, usando lo aprendido.

- Definir funciones en los programas es útil porque

  _____

  _____

  _____

  cuando se requiere el uso de una misma secuencia de instrucciones varias veces.

- Con lo que he aprendido hasta ahora podría

  _____

  _____

  _____

  para resolver el reto propuesto.

Utiliza el siguiente espacio para crear un gráfico o esquema que destaque lo que aprendiste en esta sesión. Luego, compara tu trabajo con el de tus compañeras y compañeros y agrégale lo que necesite, si sientes que debes complementarlo después de haber visto otras propuestas.

Grado 8° Guía 1

TIC

Sesión 4

Aprendizajes esperados

Al final de esta sesión verifica que puedas

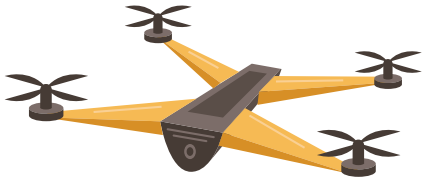
- Definir y utilizar funciones para simplificar la escritura de programas.
- Utilizar arreglos para guardar datos y hacer cálculos.
- Usar casos de prueba para verificar el funcionamiento correcto de un algoritmo.

Duración sugerida

40% 40% 20%

Material para la clase

- ○ Acceso a *MakeCode*



Lo que sabemos, lo que debemos saber

Esta sección corresponde al 40% de avance de la sesión

Es hora de que comiences a desarrollar el programa de computador que se pide en el reto. Para esto, necesitarás utilizar lo aprendido en cuanto a:

- Variables
- Arreglos
- Operaciones usando variables y arreglos
- Funciones

Recuerda que el helicóptero marciano (*Mars Helicopter*) es un helicóptero robótico que sirve de demostración tecnológica para explorar objetivos interesantes en el planeta Marte, y como base para planificar la mejor ruta para próximas misiones.

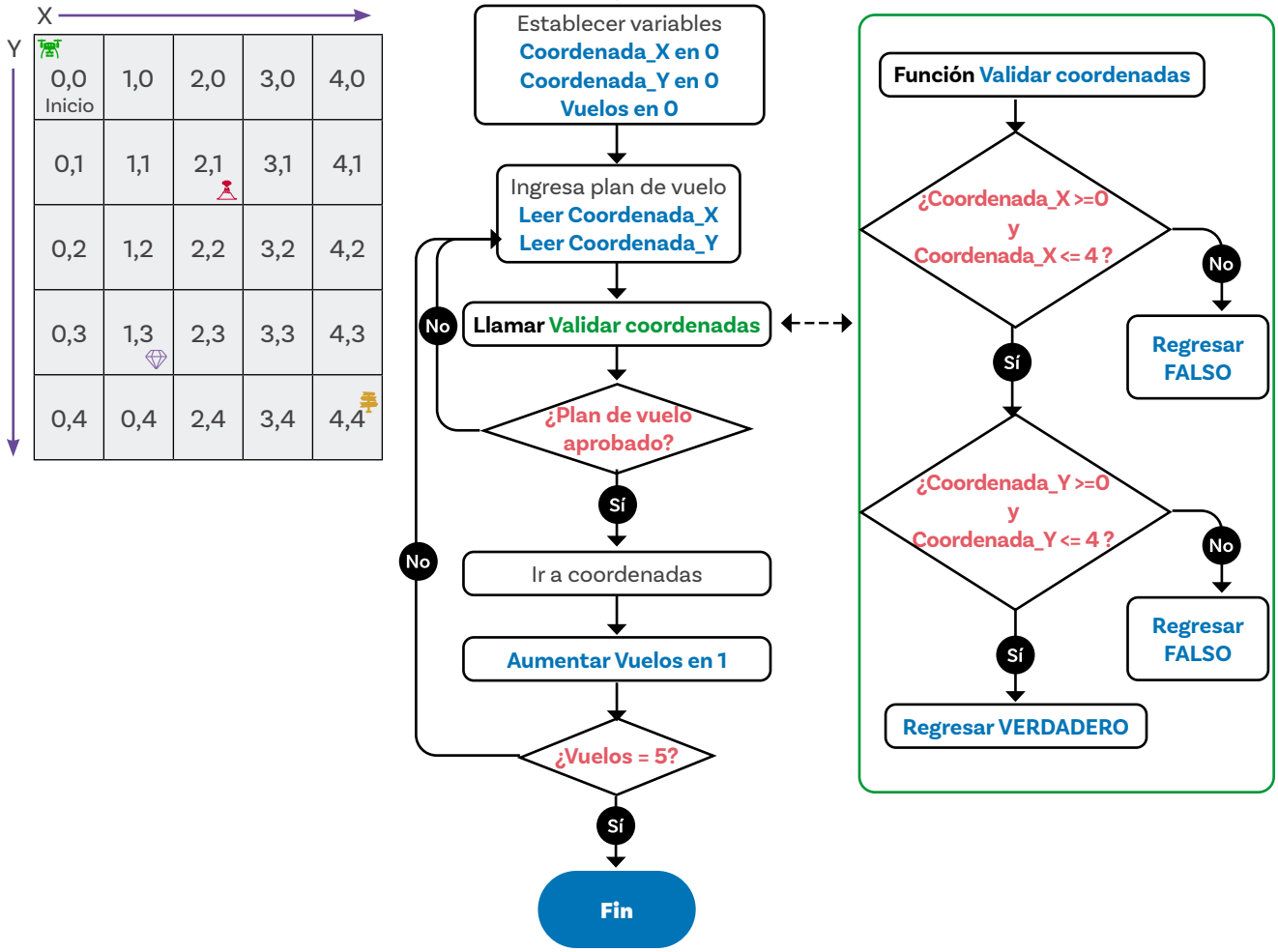
Primero que todo, la misión será programar el helicóptero-robot para realizar 5 vuelos de exploración en Marte. Este helicóptero-robot es muy pequeño, sólo puede realizar 5 vuelos en Marte, con una duración máxima de 90 segundos cada uno, y hacer despegues y aterrizajes verticales controlados. Por tanto, este helicóptero marciano no puede ir lejos.

Trabaja en grupos de 3 personas, según las indicaciones de tu docente.

Sigan paso a paso las instrucciones del diagrama de flujo para determinar las coordenadas x,y del punto de partida del helicóptero-robot y planificar los 5 vuelos de prueba.

Seguir un programa paso a paso, usando lápiz y papel, es la mejor opción para comprenderlo y depurar su código.

Figura 1. Vuelos exploratorios del helicóptero marciano.



- Realicen los vuelos exploratorios en el tablero de navegación de la *Figura 1*. Los planes de vuelo deben establecerse dentro de esos rangos de coordenadas.
- Lleven el conteo de la variable "Vuelos" en una hoja.
- Una de las personas del equipo tendrá el rol de función y se especializará en ejecutar la **función *Validar coordenadas***.

Como recordarán, las funciones ayudan a hacer los códigos más legibles. Así que cada vez que se necesite hacer una verificación de coordenadas, solo se verá el llamado a la función y no todas las líneas de código necesarias para tal verificación.

Recuerden que ya se ha hablado de los casos de prueba. Necesitan probar la función con varios casos de prueba.

Con la función *Validación de coordenadas* se está haciendo una validación de los datos ingresados en las coordenadas. Esto es clave pues si los datos quedan mal, el helicóptero-robot quedaría fuera del área en la que se puede desplazar, dada su autonomía limitada.

Para verificar que los casos de prueba estén funcionando de forma adecuada, usen la *Tabla 1.*.

Si lograron verificar 5 vuelos que cumplen las condiciones para realizar cada viaje, ya pueden avanzar a la siguiente misión con la *micro:bit*.

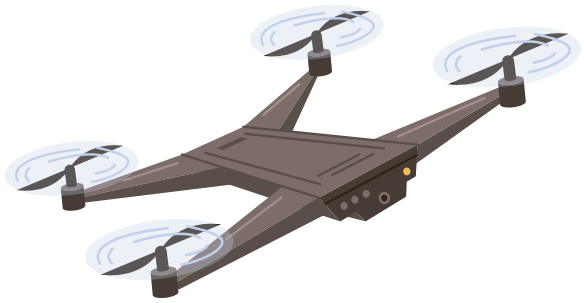


Tabla 1. Prueba de funcionamiento

Pregunta	¿Funciona?	
	Sí	No
Valor menor a 0 en una coordenada		
Valor mayor a 4 en una coordenada		
Valor menor a 0 en una coordenada y mayor a 4 en la otra		
Ambos valores desde 0 y hasta 4		

Glosario

Depurar: Significa verificar el funcionamiento de un programa y resolver lo que no funciona. Se puede realizar siguiendo con lápiz y papel la evolución del algoritmo o sus variables o utilizar, por ejemplo, en *MakeCode*, la opción de depuración ( ) e ir ejecutando instrucción por instrucción y examinar qué va pasando.

Matriz: es un tipo de arreglo o tabla para almacenar información en filas y columnas.

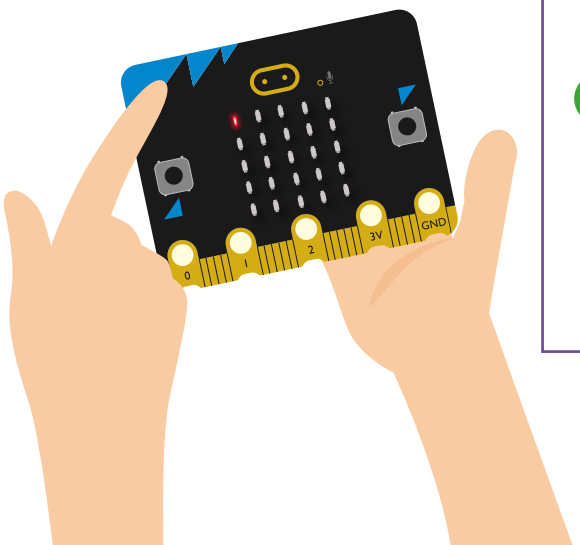
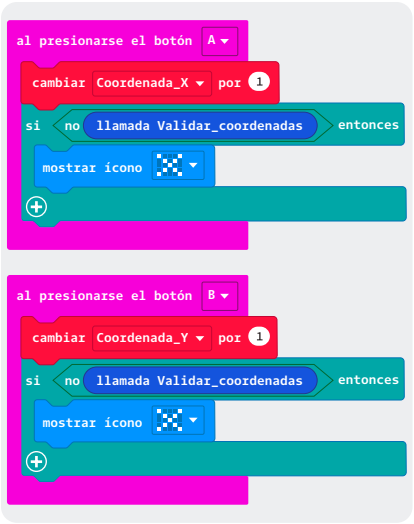


Figura 2. Código para asignar coordenadas iniciales de vuelo

Este código asigna las dos coordenadas iniciales, indicando el punto de partida (0,0) y lo ubica en la pantalla.

Figura 3. Código para cambiar y validar coordenadas x,y



Con este código, al oprimir el botón A aumenta en 1 el valor de la coordenada X. Al oprimir el botón B aumenta en 1 el valor de la coordenada Y. En ambos casos se utiliza la función para verificar el valor registrado en las coordenadas.

Manos a la obra

Conectadas

Esta sección corresponde al 80% de avance de la sesión

Después de trabajar en la actividad desconectada, tu equipo y tú ya tienen mayor conocimiento sobre la misión a Marte. Ahora es momento de trabajar en *MakeCode* haciendo un programa que les permita simular algunas de las instrucciones necesarias para lograr que el helicóptero vuele dentro de un rango de coordenadas limitado.

En esta primera tarea van a simular en el editor *MakeCode* de la *micro:bit* solamente el incremento de las coordenadas X y Y utilizando los botones A para X y B para Y. Luego, es necesario probar cada movimiento del vuelo, con una nueva coordenada, para verificar que se encuentre dentro del área de trabajo.

La *micro:bit* cuenta con una *matriz* de 25 LED programables dispuestos en una cuadrícula de 5 por 5. Este será el espacio (simulado) para los vuelos de exploración.

Para comenzar, deben configurar las variables que utilizarán en el programa , como se muestra en la *Figura 2*. El bloque “al iniciar” se utiliza para realizar acciones que solo se adelantan cuando el programa comienza a funcionar.

Una vez configuradas estas variables, pueden configurar los botones A y B para incrementar los valores de las coordenadas e ir describiendo una ruta de vuelo, tal como se ve en la *Figura 3*.

Falta que agreguen la función de verificación. Pueden probar con la siguiente:

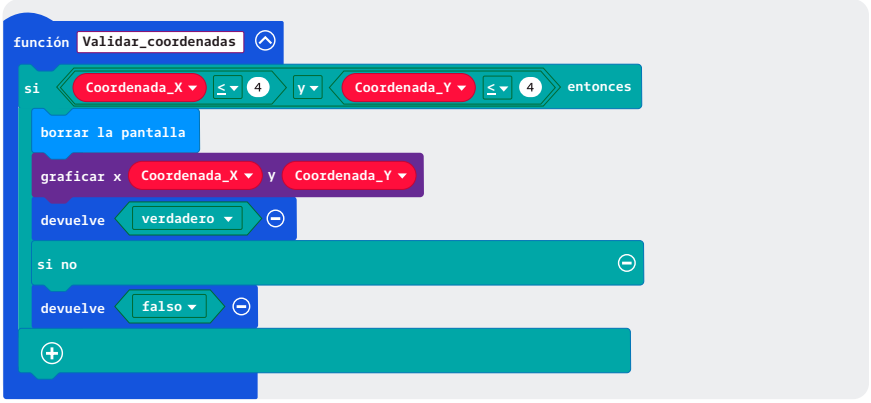


Figura 4. Bloque devolver para funciones

Analicen estos bloques, traten de anticipar qué hacen y luego pruébenlos en el *MakeCode* para verificar su predicción.

> *¿Se muestra la posición del helicóptero?*
> *¿Se detecta un error de plan de vuelo cuando el helicóptero se sale del área prevista?*

En esta función (*Validar_coordenadas*) se usa un bloque *graficar x,y* que pueden encontrar en los bloques de LED. Este bloque permite encender el LED que se encuentra en la coordenada (x, y) de la pantalla de la *micro:bit*. Recuerden que estas coordenadas empiezan en (0,0) en la parte superior izquierda de la matriz y se comportan como un arreglo de dos dimensiones, que guarda un cero para LED apagado y un 1 para LED encendido.

Tengan en cuenta que cuando la función utiliza el bloque *devuelve* aparece un tercer bloque en la sección de bloques de Funciones, que servirá para hacer uso del resultado que regresa tu función, como se ve en la *Figura 4*.

¿Cómo les fue con esta segunda misión?

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión. ¿Crees que lograste alcanzarlos?

1. ¿Puedes definir y utilizar funciones para simplificar la escritura de programas?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

2. ¿Puedes utilizar arreglos para guardar datos y hacer cálculos?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

3. ¿Puedes usar casos de prueba para verificar el funcionamiento correcto de un algoritmo?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

Si tus respuestas a las preguntas anteriores fueron "Parcialmente" o "Aún no", puedes revisar nuevamente los contenidos de esta sesión, tratando de seguir paso a paso las instrucciones que se ofrecen. Mientras lo haces, toma nota de las preguntas o dificultades que te surjan, y consulta con tu docente para aclararlas o para realizar las actividades adicionales que te proponga.

Es el momento de responder algunas preguntas:

Si examinas lo logrado hasta ahora, ¿te permite disminuir las coordenadas el programa que hiciste en esta sesión?

¿Qué más te haría falta para resolver el reto?

Para finalizar, te proponemos hacer un crucigrama con algunos de los términos que has aprendido en esta guía. Cuando lo tengas listo intercambia crucigramas con alguna compañera o compañero de clase y rétense mutuamente a descubrir las palabras a partir de las pistas propuestas.

Grado 8° **Guía 1**

TIC

Sesión 5

Aprendizajes esperados

Al final de esta sesión se verifica que puedas:

- Modificar un programa para incluir nuevas especificaciones.
- Mejorar la visualización de los resultados.
- Depurar un programa en *MakeCode*.

Duración sugerida

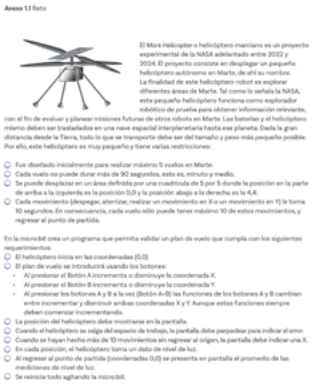
15% 70% 15%

Material para la clase

- ○ Acceso a *MakeCode*
- ○ Anexo 1.1

Anexo

Anexo 1.1



Lo que sabemos, lo que debemos saber

Esta sección corresponde al 15% de avance de la sesión

Hasta este momento ya has logrado resolver buena parte del reto planteado en el *Anexo 1.1.* Vuelve y lee lo que se solicita ahí para que identifiques lo que falta por ajustar.

> *¿Ya lo encontraste?*

Sí, como el programa que has hecho sólo permite aumentar las coordenadas, el helicóptero no puede regresar.

Antes de abordar la parte final de este reto, te falta aprender a depurar un programa usando las herramientas de *MakeCode*.

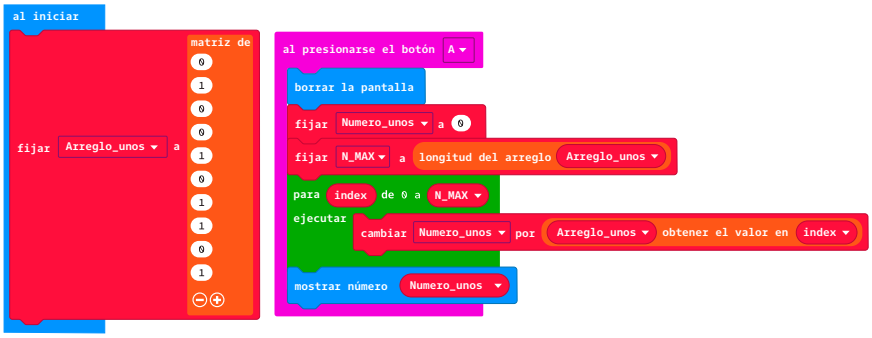
A menudo no se tiene certeza de lo que está pasando en un programa, salvo que el resultado esperado no se produce.

> *¿Cómo seguir paso a paso un programa, examinando lo que va sucediendo?*

Examinar lo que va pasando en un programa para encontrar un error se denomina depurar.

Para aprender cómo hacerlo, observa el siguiente programa presentado en la *Figura 1*:

Figura 1. Programa de arreglos en *MakeCode*



Este programa debería contar el total de unos almacenados en el arreglo. Escríbelo y pruébalo. ¿Lo hace?

Figura 2. Menú de depuración

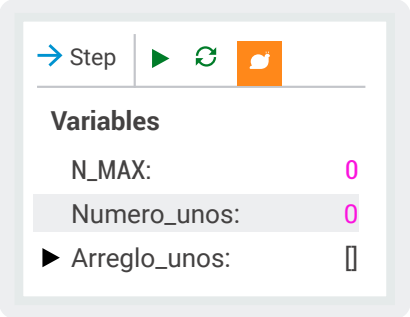
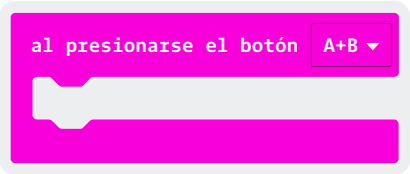


Figura 3. Bloque presionar A+B



Recomendación

Para el cambio de dirección, una posibilidad es utilizar una variable que por ejemplo guarde 1 si debe crecer el valor o -1 si se quiere que decremente o disminuya el valor:

De esta forma se sumará este valor que hará que avance o retroceda.

Probablemente habrás detectado que hay problemas. Pero ¿cómo encontrar el origen del problema?

Para probarlo paso a paso, sigue las siguientes instrucciones:

1. Activa la función de depuración:
2. Aparecerá a la derecha un menú como el que observas en la *Figura 2*. Verás las tres variables que se crearon, *N_MAX* que tiene el valor de cero, *Arreglo_unos* que aparece vacía aún [], y *Numero_unos* que también tiene el valor de cero.
3. Activa el caracol para que puedas hacer avanzar el programa paso a paso, presionando la flecha que se encuentra a la izquierda de este botón.
4. Recuerda oprimir el botón A para que el programa entre a correr.
5. Encuentra el error al examinar si las variables cambian como se esperaba o no e identifica dónde se produce el error.

Deberás ahora probar el programa para diferentes trayectorias y examinar si el resultado es el esperado. Examina estos dos casos de prueba:

- ○ Si llego a un valor superior a 4 en la coordenada X, ¿es rechazado este último movimiento?
- ○ Si llego a un valor superior a 4 en la coordenada Y, ¿es rechazado este último movimiento?

Ahora piensa en lo siguiente. El programa todavía tiene varias dificultades que deben solucionarse. Una de ellas consiste en que si quisieras introducir una nueva ruta de vuelo no tienes forma de iniciar de nuevo en la posición (0,0).

Incluye una acción que inicie las coordenadas en cero cuando se opriman los botones A y B simultáneamente, como se muestra en la *Figura 3*. Esto le indicará a la *micro:bit* que vas a introducir una nueva trayectoria. Seguro ya lo puedes hacer.

Manos a la obra

Conectadas

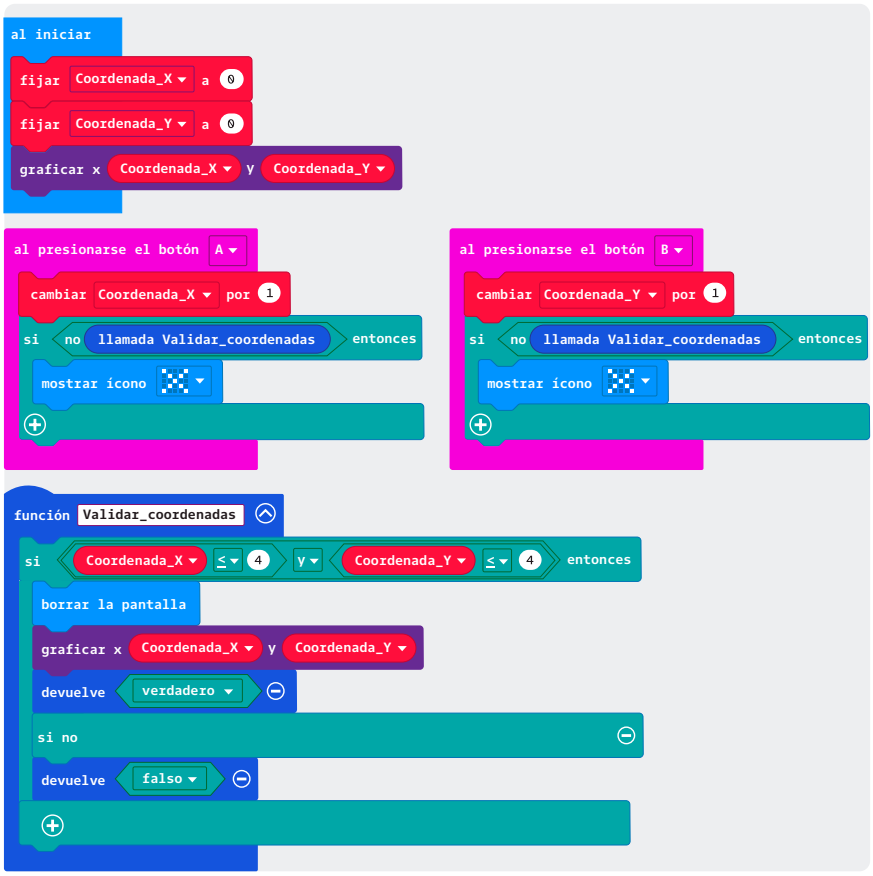
Esta sección corresponde al 85% de avance de la sesión

Si en algún momento sienten que no saben cómo continuar su programa, pregunten a su docente si les permite desplazarse, silenciosamente, por el aula para observar el trabajo que están haciendo otros grupos.

Ahora sí, es el momento de que revises el reto y termines de desarrollar la solución completa.

El programa hasta ahora desarrollado es el siguiente:

Figura 4. Avance del programa en *MakeCode*



Queda faltando:

- ◯ Cambiar de sentido al presionar los botones A+B.
- ◯ Tomar datos de nivel de luz al realizar cada vuelo.
- ◯ Mostrar el promedio de niveles de luz al regresar a la posición (0,0).

- [ ] Contar los movimientos para asegurarse que se llegará de nuevo a (0,0) en máximo 10 movimientos.

Trabaja en grupo, según indique tu docente, y completen el programa que resuelva el reto.

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión. ¿Crees que lograste alcanzarlos?

1. ¿Puedes modificar un programa para incluir nuevas especificaciones?
   - [ ] Sí
   - [ ] Parcialmente
   - [ ] Aún no
2. ¿Puedes mejorar la visualización de los resultados?
   - [ ] Sí
   - [ ] Parcialmente
   - [ ] Aún no
3. ¿Puedes depurar un programa usando *MakeCode*?
   - [ ] Sí
   - [ ] Parcialmente
   - [ ] Aún no

> **Si tus respuestas fueron "Parcialmente" o "Aún no", no olvides revisar nuevamente los contenidos de esta sesión, seguir paso a paso las instrucciones para el desarrollo del código, y consultar con tu docente cualquier inquietud que te surja en el proceso.**

Ahora que ya has trabajado en un pequeño simulador y validador de vuelo de un helicóptero:

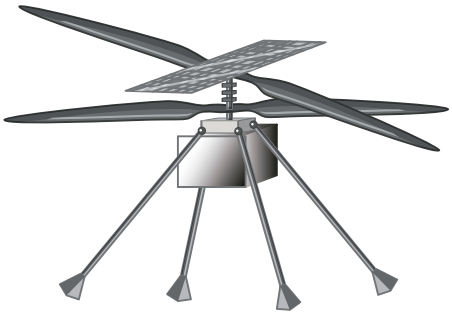
¿Qué sentiste que fue lo más complejo?

¿Por qué?

¿Cómo lo resolviste?

Finaliza haciendo un diagrama que te recuerde cómo se depuran los programas hechos en *MakeCode*. Esto te servirá más adelante.

Anexo 1.1 Reto



El *Mars Helicopter* o helicóptero marciano es un proyecto experimental de la NASA adelantado entre 2022 y 2024. El proyecto consiste en desplegar un pequeño helicóptero autónomo en Marte, de ahí su nombre. La finalidad de este helicóptero-robot es explorar diferentes áreas de Marte. Tal como lo señala la NASA, este pequeño helicóptero funciona como explorador robótico de prueba para obtener información relevante, con el fin de evaluar y planear misiones futuras de otros robots en Marte. Las baterías y el helicóptero mismo deben ser trasladados en una nave espacial interplanetaria hasta ese planeta. Dada la gran distancia desde la Tierra, todo lo que se transporte debe ser del tamaño y peso más pequeño posible. Por ello, este helicóptero es muy pequeño y tiene varias restricciones:

- Fue diseñado inicialmente para realizar máximo 5 vuelos en Marte.
- Cada vuelo no puede durar más de 90 segundos, esto es, minuto y medio.
- Se puede desplazar en un área definida por una cuadrícula de 5 por 5 donde la posición en la parte de arriba a la izquierda es la posición 0,0 y la posición abajo a la derecha es la 4,4.
- Cada movimiento (despegar, aterrizar, realizar un movimiento en X o un movimiento en Y) le toma 10 segundos. En consecuencia, cada vuelo sólo puede tener máximo 10 de estos movimientos, y regresar al punto de partida.

En la *micro:bit* crea un programa que permita validar un plan de vuelo que cumpla con los siguientes requerimientos:

- El helicóptero inicia en las coordenadas (0,0)
- El plan de vuelo se introducirá usando los botones:
  - Al presionar el Botón A incrementa o disminuye la coordenada X.
  - Al presionar el Botón B incrementa o disminuye la coordenada Y.
  - Al presionar los botones A y B a la vez (Botón A+B) las funciones de los botones A y B cambian entre incrementar y disminuir ambas coordenadas X y Y. Aunque estas funciones siempre deben comenzar incrementando.
- La posición del helicóptero debe mostrarse en la pantalla.
- Cuando el helicóptero se salga del espacio de trabajo, la pantalla debe parpadear para indicar el error.
- Cuando se hayan hecho más de 10 movimientos sin regresar al origen, la pantalla debe indicar una X.
- En cada posición, el helicóptero toma un dato de nivel de luz.
- Al regresar al punto de partida (coordenadas 0,0) se presenta en pantalla el promedio de las mediciones de nivel de luz.
- Se reinicia todo agitando la *micro:bit*.

TIC

Apoya:

Educación

BRITISH COUNCIL

Colombia Programa
{EL CÓDIGO A TU FUTURO}