Parqueaderos

Grado 11° Guía 3

TIC

Estudiantes

Educación

BRITISH COUNCIL

Colombia Programa
{EL CÓDIGO A TU FUTURO}

Parqueaderos

Grado 11° Guía 3

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 11º Guía 3

TIC

Guía de íconos

- Lógica, programación y depuración
- Computación física
- Prácticas de datos

Aprendizajes de la guía

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

- Identificar, programar y controlar componentes de entrada y salida en un sistema automatizado, desarrollando algoritmos que gestionen su comportamiento.
- Integrar y programar diversos componentes electrónicos para ampliar las capacidades de un microcontrolador, creando un sistema interactivo funcional.

Resumen de la guía

Esta guía está diseñada para profundizar en el mundo de la computación física y la programación aplicada. A lo largo de cinco sesiones, se explorará cómo crear un sistema automatizado de gestión de parqueadero utilizando la *micro:bit* como unidad de control central. Se aprenderá a simular conexiones y la programación de este microcontrolador para manejar un contador, controlar servomotores que simulan barreras de entrada y salida y utilizar la pantalla led integrada para mostrar información del estado de un sistema.

El proyecto culminará con la implementación de un prototipo de parqueadero inteligente que puede contar espacios, controlar el acceso mediante barreras automatizadas y mostrar el estado actual del parqueadero.

Resumen de las sesiones

Sesión 1

En esta sesión se presenta el entorno de programación y simulación *Tinkercad*. Se propone un repaso de la creación de variables, la visualización de números en la pantalla led y la configuración de los pines para conectar servomotores. Se trabajará en un código básico para inicializar el sistema.

Aprendizajes de la guía

Concebir e implementar un prototipo tecnológico que aborde un problema específico, considerando las necesidades del usuario final y las limitaciones técnicas existentes.

Aplicar conceptos de programación y análisis de datos en la creación de soluciones tecnológicas para problemas del mundo real, evaluando críticamente su eficacia y potencial de mejora.

Sesión 2

Esta sesión se centrará en la implementación de las instrucciones que manejan la entrada de vehículos. Para ello se programará la *micro:bit* para simular la entrada de los vehículos y se utilizarán variables para controlar la capacidad del sistema.

Sesión 3

En esta sesión se programarán nuevas funciones, simulando la salida de un vehículo. Se deberá ajustar las variables de capacidad del sistema. También se reforzará el uso de condicionales para prevenir operaciones inválidas.

Sesión 4

Se añaden mejoras al código para complejizar el sistema y proveer más información a los usuarios.

Sesión 5

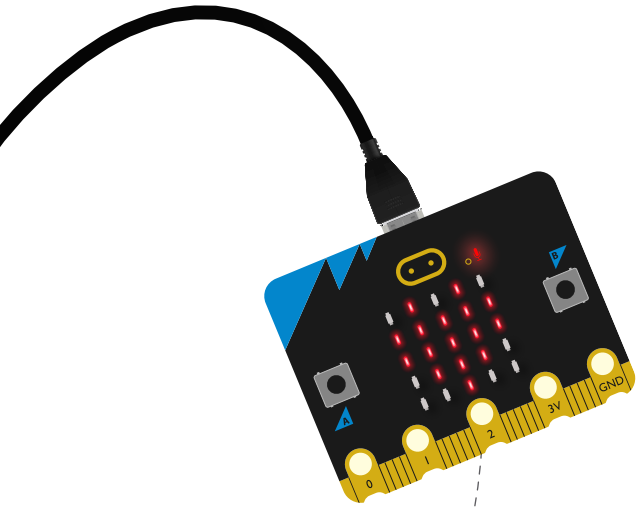
Esta sesión se enfocará en refinar el código existente y realizar pruebas para mejorar la lógica de manejo de casos límite, como intentar ingresar cuando el parqueadero está lleno o salir cuando está vacío. Se proponen mejoras en la interfaz de usuario, como mostrar mensajes más informativos en la pantalla led. También incluye la realización de pruebas para asegurar que el sistema funcione correctamente en todos los escenarios posibles.

Nota

Se recomienda que tu docente cree una clase en *Tinkercad* siguiendo el *Anexo 1.2.*

Si se requiere

- ○ Guía 1 de Grado 7 para aprendizajes iniciales sobre variables y arreglos.
- ○ Guía 1 Grado 5 para aprendizajes iniciales sobre el uso de *MakeCode*.



Conexión con otras áreas

Esta guía permite poner en práctica habilidades de diferentes áreas. A continuación se presentan algunos puntos de conexión:

Matemáticas

- Se refuerzan conceptos algebraicos al utilizar variables para controlar la capacidad del parqueadero y realizar operaciones aritméticas. Además, la implementación de condicionales para validar las acciones fomenta el uso de lógica matemática. Además, durante el montaje y uso de diferentes actuadores, se fortalecen habilidades relacionadas a la Ingeniería y el desarrollo de proyectos.

Ciencias Sociales

- El desarrollo de las actividades propuestas en estas guías permiten reflexionar sobre el uso de tecnología en la organización de ciudades inteligentes y su impacto en la calidad de vida de ciudadanas y ciudadanos.

TIC

Sesión 1

Aprendizajes esperados

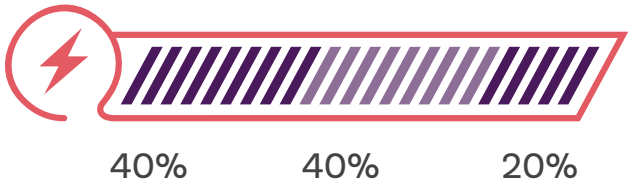
Al final de esta sesión verifica que puedas:

- Simular un sistema en *Tinkercad*.
- Crear y manipular variables en *Python* dentro del entorno *Tinkercad*.
- Configurar pines virtuales para controlar servomotores en *Tinkercad*.

Material para la clase

- ○ Anexo 1.1
- ○ Computador con acceso a internet

Duración sugerida



Lo que sabemos, lo que debemos saber

Esta sección corresponde al 40% de avance de la sesión

Observa las *Figuras 1, 2 y 3* y responde:

¿?

¿Has visto alguno de estos sistemas?
¿Cuál es su función?

Figura 1. Fila virtual

Figura 2. Control de turnos en centro de atención

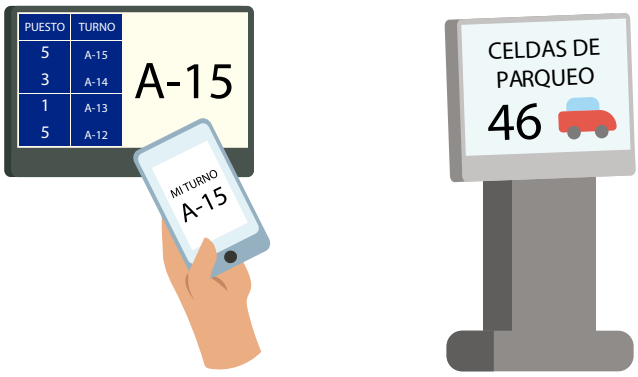


Figura 3. Disponibilidad celdas de parqueo

Anexo

Anexo 1.1

Tu reto es crear un sistema de parqueadero inteligente utilizando una micro:bit como unidad de control central. El sistema debe gestionar un parqueadero con 9 espacios disponibles inicialmente. Utilizarás dos servomotores conectados a los pines P15 y P16 de la micro:bit para simular las barreras de entrada y salida respectivamente. La pantalla led integrada de la micro:bit se usará para mostrar constantemente el número de espacios disponibles. Los botones A y B de la micro:bit simularán la llegada y salida de vehículos: el botón A representará un vehículo entrando, mientras que el botón B representará un vehículo saliendo.

El sistema debe ser capaz de:

1. Disminuir el contador de espacios y abrir la barrera de entrada (mover el servomotor en P15 a 0 grados) cuando se presiona el botón A, siempre y cuando haya espacios disponibles.
2. Aumentar el contador de espacios y abrir la barrera de salida (mover el servomotor en P16 a 0 grados) cuando se presiona el botón B, siempre que el parqueadero no esté vacío.
3. Mostrar un ícono de No en la pantalla led cuando se intente ingresar a un parqueadero lleno o salir de un parqueadero vacío.
4. Mantener las barreras abiertas por 5 segundos antes de cerrarlas (volver a 90 grados).
5. Actualizar y mostrar el número de espacios disponibles después de cada operación. Tu reto incluye implementar esta lógica, asegurando que el sistema funcione de manera fluida y maneje correctamente todos los casos posibles, incluidos los casos límite cuando el parqueadero está lleno o vacío.

Para ir más lejos: adicionalmente, el sistema debe incorporar un conjunto de led para mejorar la señalización visual. Se utilizarán cuatro led en total: un par de led (rojo y verde) para cada barrera (entrada y salida). El led rojo permanecerá encendido mientras la barrera correspondiente esté abajo y se apagará cuando la barrera esté arriba. El led verde, por el contrario, se encenderá cuando la barrera esté arriba y se apagará cuando esté abajo. Esta característica proporcionará una indicación visual clara del estado de cada barrera, mejorando la experiencia del usuario y la seguridad del sistema.

Figura 1: fila virtual para comprar boletas o productos cuando hay muchos(as) clientes intentando entrar a una página de internet y así no sobrepasar su capacidad.

Figura 2: diferentes formas de llevar los turnos en un centro de atención, por ejemplo, la sala de espera de un consultorio o en un centro de servicio al cliente. Así, las y los clientes conocen el tiempo de espera y se respeta la capacidad de atención de las personas que trabajan.

Figura 3: un anuncio de la cantidad de celdas de parqueo disponibles por piso. Así se controla la capacidad del parqueadero y se evita que muchas personas esperen a que se libere un cupo.

¿?

¿Notas algo en común en todos los ejemplos?
¿Cómo crees que funciona?

Durante esta guía vas a simular el funcionamiento de un parqueadero. A través de este proyecto vas a fortalecer tus habilidades en programación, a adquirir conocimientos nuevos y comprender mucho mejor cómo funcionan este tipo de sistemas tan utilizados en el día a día. Antes de continuar, lee el reto que se encuentra en el *Anexo 1.1.*

Como sabes, la *micro:bit* es una pequeña computadora programable que puede controlar diversos componentes y mostrar información. En nuestro parqueadero inteligente, la usaremos para llevar la cuenta de los espacios disponibles en un parqueadero y controlar las barreras de entrada y salida.

En nuestro proyecto, esto implica que debemos programar la *micro:bit* para mostrar números en su pantalla led y configurar los pines para controlar los servomotores de las barreras, entre otras cosas. Sin embargo, no siempre contamos con suficientes elementos físicos como servomotores, cables o baterías. Así que toda la solución del proyecto la realizarás en un entorno simulado llamado *Tinkercad*.

Anexo

Anexo 1.2

Nota

Se recomienda que tu docente cree una clase en *Tinkercad* siguiendo el *Anexo 1.2*

Enlace

Tinkercad

Tinkercad es un simulador de circuitos electrónicos basado en navegador que admite microcontroladores Arduino Uno, placas *micro:bit* o chips ATtiny. El código se puede crear utilizando CodeBlocks gráficos, fragmentos de código que se pueden organizar fácilmente con el ratón o código basado en texto.

En años anteriores aprendiste que las simulaciones son útiles para ahorrar tiempo, dinero y ejecutar experimentos que no podrías hacer desde tu salón. Durante esta guía aprenderás que simular sistemas también resulta muy útil ya que puedes hacer muchas pruebas sin provocar daños permanentes, además de que cuentas con un catálogo muy amplio de elementos, sin necesidad de invertir dinero.

Para ingresar a *Tinkercad* sigue estas instrucciones:

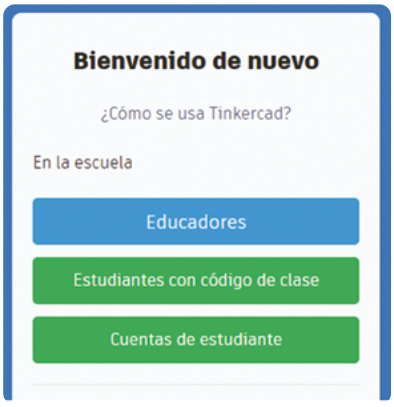
Paso 1: Entrar a *Tinkercad*

- ○ Accede a *Tinkercad* siguiendo el enlace o el QR.
- ○ Haz clic en Iniciar sesión como se presenta en la *Figura 4*.

Figura 4. Iniciar sesión en *Tinkercad*

- ○ Elige Estudiantes con código de clase como se presenta en la *Figura 5*.

Figura 5. Selección de Estudiantes con código de clase



- Ingresa el código que te comparta tu docente.

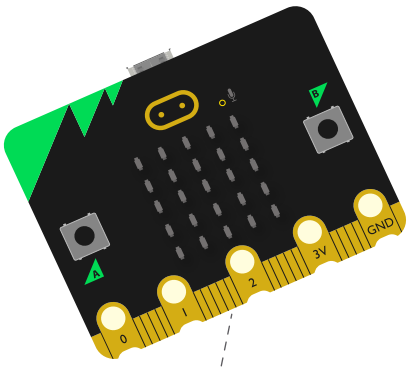
Figura 6. Unirse a clase con código

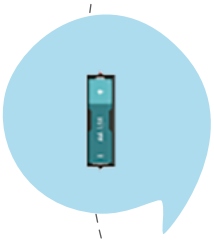
- Elige Unirse con apodo, a menos que tu docente indique algo diferente.

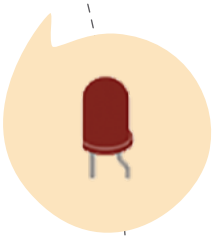
Figura 7. Ingreso de nombre de identificación en la clase

- Escribe el alias según las indicaciones de tu docente y haz clic en el botón ¡Soy yo!

Figura 8. Ingreso al curso con botón: ¡Soy yo!

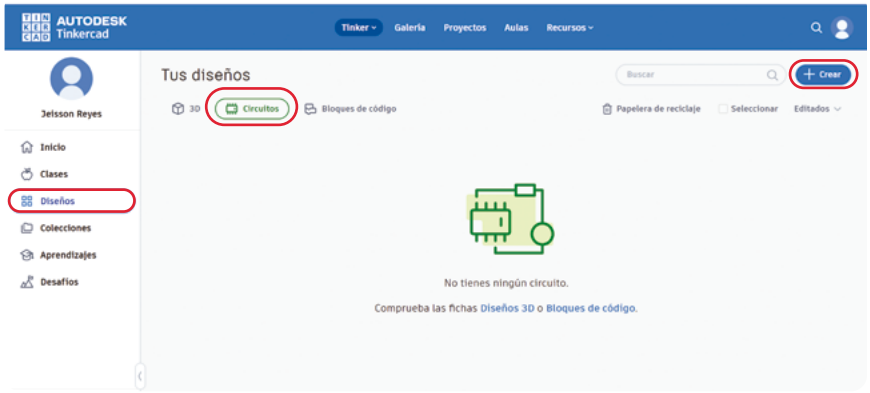






- Crea una nueva simulación de circuito haciendo clic en Crear y luego en circuito.

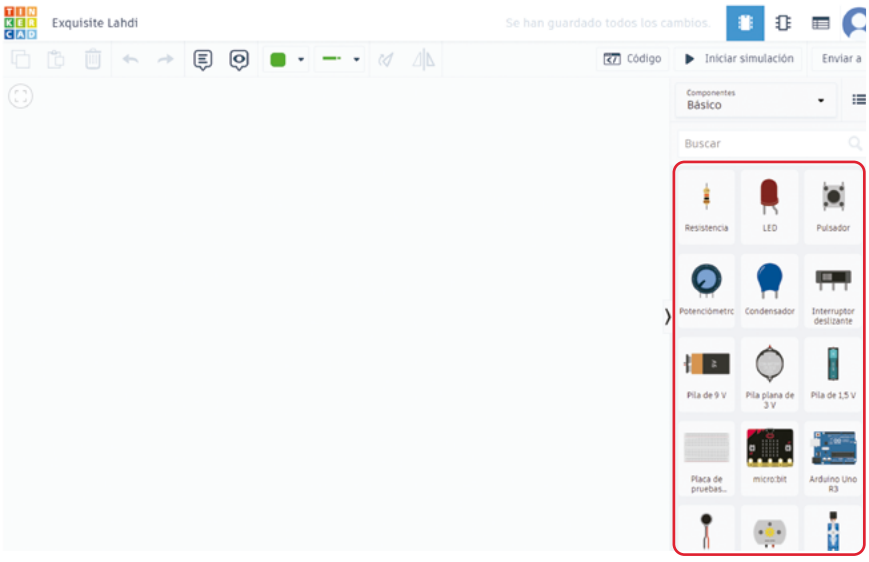
Figura 9. Creación de nueva simulación



- Tómate unos minutos para explorar la interfaz como se presenta en la *Figura 10*: componentes, editor de código, simulador. Los componentes de la derecha pueden ser arrastrados y colocados en el área de trabajo.

¿Cuáles reconoces?
¿Cuáles crees que serán útiles en la resolución del reto?

Figura 10. Exploración de interfaz gráfica de *Tinkercad*



Manos a la obra
Conectadas

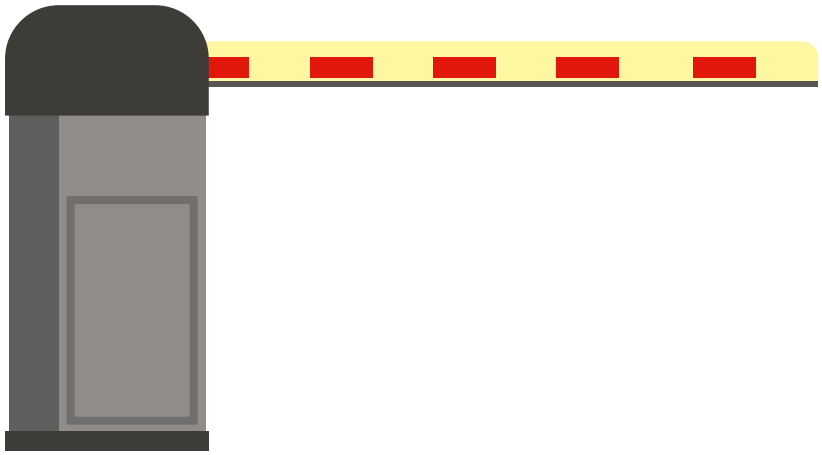
Esta sección corresponde al 80% de avance de la sesión

Ya habiendo explorado el entorno de *Tinkercad*. Vas a completar los dos primeros pasos hacia la simulación del parqueadero: 1. Realizar las conexiones para configurar los servomotores y 2. Crear variables y mostrarlas en la pantalla de led.

Paso 1. Configuración de servomotores

Es posible que en grados anteriores hayas realizado algunas conexiones de sensores y actuadores. En este caso, vas a realizar una conexión de dos servomotores y sus baterías a la *micro:bit*. Los servomotores son los elementos que controlan el movimiento de la barra de entrada al parqueadero.

Figura 11. Control ingreso parqueadero



Tal como en una conexión real, necesitas varios materiales:

- ○ *Micro:bit* con salidas
- ○ 2 servomotores
- ○ Placa de pruebas
- ○ Baterías o pilas de 1.5V

- Añade una *Micro:bit* al espacio de trabajo. Para esto haz clic en la opción buscar y escribe *micro:bit*.

Figura 12. Búsqueda de dispositivos y elementos en *Tinkercad*

- Selecciona la opción *micro:bit* con salida

Figura 13. Selección de componente micro:bit

- Añade dos servomotores al circuito en *Tinkercad*. Búscalos en la sección Componentes:

Figura 14. Selección de servomotores

Figura 16. Conexión de pin de señal

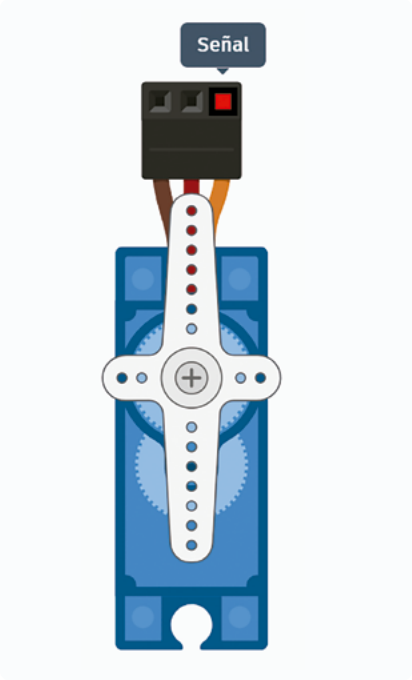
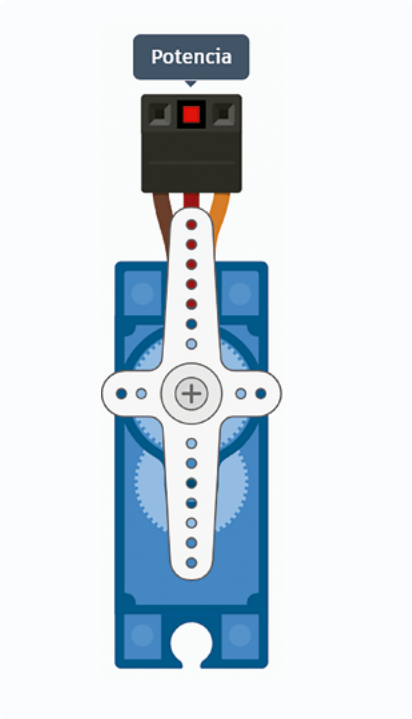


Figura 17. Conexión de pin de potencia



- Busca y añade el resto de los materiales al espacio de trabajo. Al añadir las baterías, selecciona Recuento = 2 baterías, como se ve en la *Figura 15.*

Figura 15. Selección de baterías

- Al pasar el ratón por los cables de los servomotores, puedes identificar cuál cable es la señal, potencia, y tierra.

- Conéctalos a los pines P15 y P16 de la *micro:bit* simulada. Para esto, lleva los cables de Señal al P15 o P16, el cable Potencia al polo positivo y el cable Tierra al polo negativo, siguiendo la imagen. Puedes cambiar el color de los cables para tener un entorno organizado.

Figura 19. Selección cableado

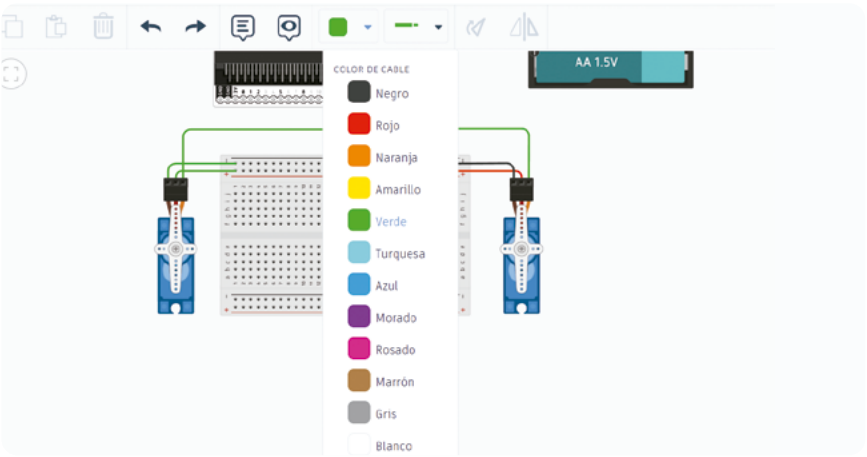


Figura 18. Conexión de pin de tierra

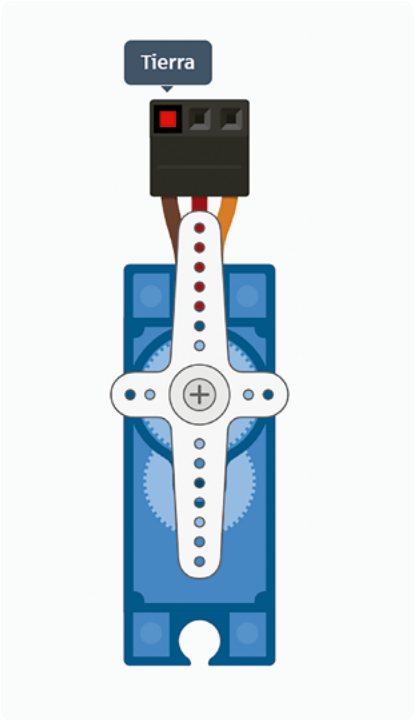


Figura 20. Conexión de pines

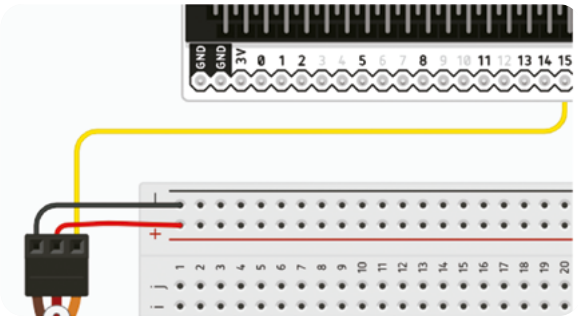
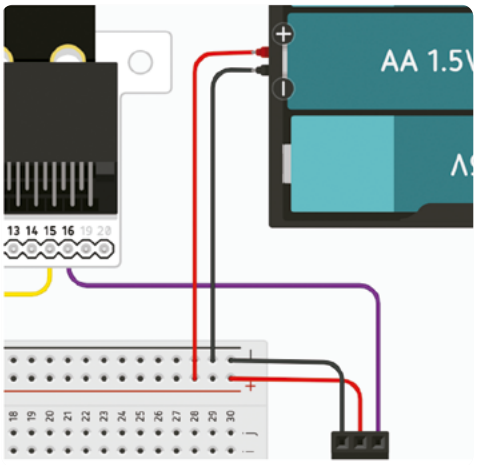
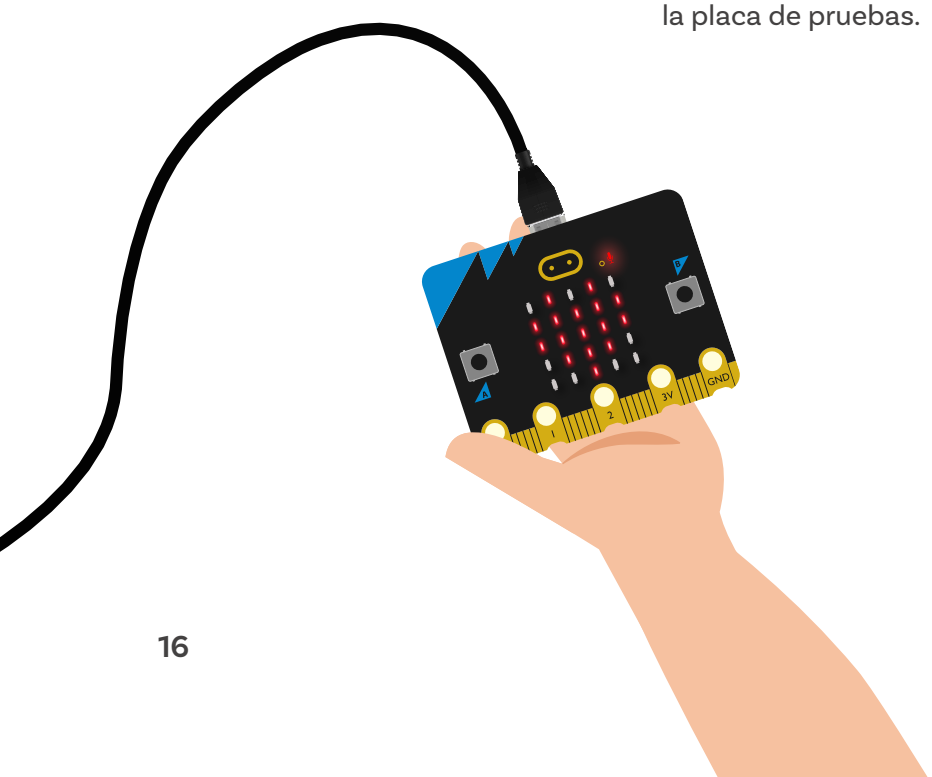


Figura 21. Conexión de baterías

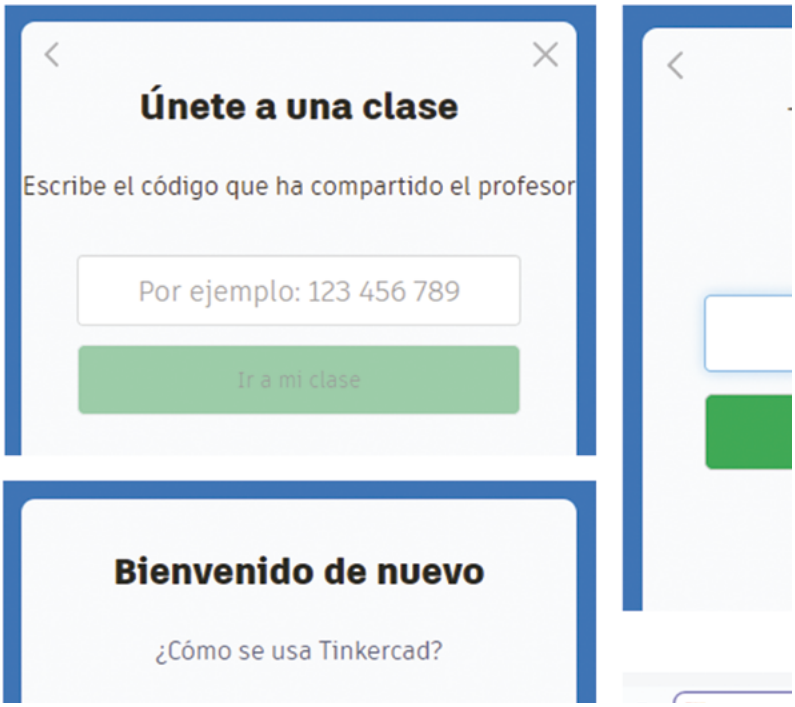


- Por último, conecta la *micro:bit*, llevando el pin de alimentación (3V) al polo positivo y el pin de Tierra (GND) al polo negativo de la placa de pruebas.



En este punto el espacio de trabajo debe verse como en la *Figura 22*.

Figura 22. Conexión completa del circuito



Paso 2: Programar la *micro:bit*

- ◯ Haz clic en el botón Código para ingresar al editor de código.

Figura 23. Ingreso a editor de código en *Tinkercad*

¿Qué ves? ¿Cómo podrías hacer que se vea una flecha en la pantalla al iniciar la simulación?

Es probable que reconozcas el lenguaje de programación por bloques con el que se puede programar la *micro:bit* desde *Tinkercad*. Haz un par de pruebas para explorar sus funciones.

○ Ahora observa estos los códigos de la *Figura 24*.

Figura 24. Visualización de programación en *Tinkercad*


```
puestosDisponibles = 9
basic.show_number(puestosDisponibles)
pins.servo_write_pin(AnalogPin.P15, 90)
pins.servo_write_pin(AnalogPin.P16, 90)
```

¿Qué hacen? Escribe tu predicción.

¿En qué se diferencian?

¿Cuántos puestos disponibles tiene el parqueadero?

Tinkercad ofrece la opción de programar las simulaciones en bloques y el texto. En este caso, el lenguaje de programación en texto es *Python*.

Figura 25. Opciones de visualización en bloques o en código

Recrea el código en tu simulador y pruébalo.

¿Qué sucede? ¿Era lo que esperabas?

Este código inicia nuestro simulador:

- Crea una variable llamada *puestosDisponibles* y la inicia en 9.
- Usa basic.show_number() para mostrar el valor en la pantalla led simulada.
- Usa pins.servo_write_pin() para controlar los servomotores.
- Establece la posición inicial de los servomotores a 90 grados.

Realiza las pruebas y verifica el funcionamiento:

- Experimenta cambiando el valor de *puestosDisponibles* y ejecuta la simulación para ver cómo se actualiza la pantalla.
- Cambiando los ángulos de giro del servomotor.

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión de forma individual respondiendo las preguntas de forma que mejor reflejen tu progreso:

1. ¿Puedes simular un sistema en *Tinkercad*?
   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no
2. ¿Puedes crear y manipular variables en *Python* dentro del entorno *Tinkercad*?
   - ◯ Sí
   - ◯ Parcialmente
   - ◯ No
3. ¿Puedes configurar pines virtuales para controlar servomotores en *Tinkercad*?
   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

Al inicio de la sesión leíste este párrafo:

> *En años anteriores aprendiste que las simulaciones son útiles para ahorrar tiempo, dinero y ejecutar experimentos que no podrías hacer desde tu salón.*

¿Cómo evidencias estas afirmaciones durante el desarrollo de la actividad?

Para cerrar la actividad te proponemos que completes la siguiente tabla y escribas una breve reflexión sobre las consecuencias que tendrían estos experimentos si utilizaras conexiones reales.

Descripción	Figura	Efecto
Conectar la tierra y el pin de alimentación de la *micro:bit* al polo negativo de la placa de pruebas	Figura 25	
Agregar 4 baterías	Figura 26	
Conectar los servomotores directamente sobre la *micro:bit*	Figura 27	

Tu reflexión:

Grado 11° | Guía 3

TIC

Sesión 2

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

- Implementar condicionales en la simulación de un sistema en *Tinkercad*.
- Actualizar y mostrar el contador de espacios disponibles en la pantalla led simulada.
- Reconocer y programar múltiples estados del sistema.

Material para la clase

- ○ Anexo 2.1

Duración sugerida

20% 40% 40%

Enlace

¿Qué es un servomotor y qué hace?

Lo que sabemos, lo que debemos saber

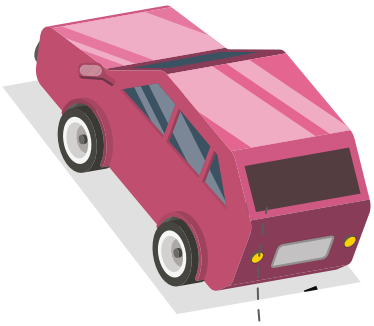
Esta sección corresponde al 20% de avance de la sesión

En la sesión anterior configuramos nuestro entorno en *Tinkercad* y verificamos cómo mostrar información básica en la *micro:bit* simulada. Además, programamos el estado inicial de dos servomotores.

¿Puedes recordar su configuración inicial? Nombra las diferentes conexiones con las que cuentan los servomotores.

Antes de continuar, observa el video sugerido para conocer más acerca de este elemento.

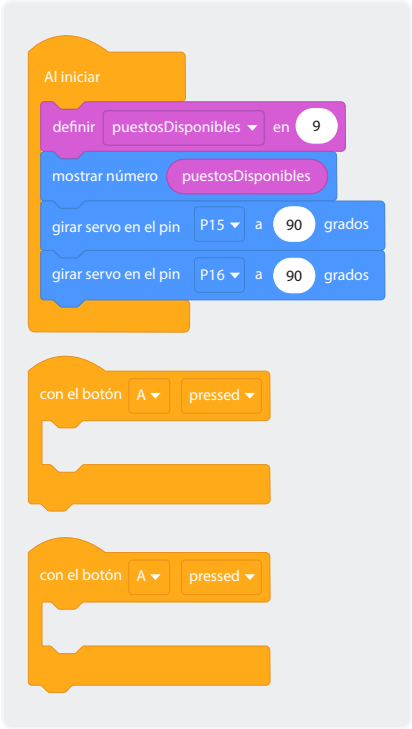
Ahora vamos a dar vida a nuestro parqueadero implementando la lógica de entrada de vehículos.



¿Has pensado alguna vez cómo un sistema automatizado decide si dejar entrar un vehículo a un parqueadero? ¿Qué factores crees que debe considerar?

En esta sesión, programaremos la *micro:bit* para que responda cuando se presiona el botón A, simulando la llegada de un vehículo. Nuestro sistema deberá decidir si hay espacio disponible, actualizar el contador y controlar la barrera de entrada.

Figura 1. Configuración entradas A y B



Manos a la obra
Conectadas

Esta sección corresponde al 60% de avance de la sesión

Sigue las indicaciones que se te dan a continuación

Paso 1: Configuración de los botones

- ○ En *Tinkercad*, asegúrate de que las instrucciones para leer los botones A y el B de la *micro:bit* estén agregados al entorno de trabajo. Para esto, selecciona la opción Básico en donde encontrarás el bloque necesario, arrástralo 2 veces al entorno de trabajo y cambia uno a B. Debe quedar como en la *Figura 1.*

- ○ El botón A manejará el servomotor de la izquierda (P15, entrada al parqueadero) y el B el de la derecha (P16, salida del parqueadero).

Paso 2: Lógica de entrada de vehículos

- ○ Crea un programa que maneje la entrada de vehículos (botón A). Al presionar el botón A el servomotor correspondiente debe girar a 0 grados por 5 segundos y devolverse a su posición inicial de 90 grados.

Figura 2. Programación acciones con entradas A y B

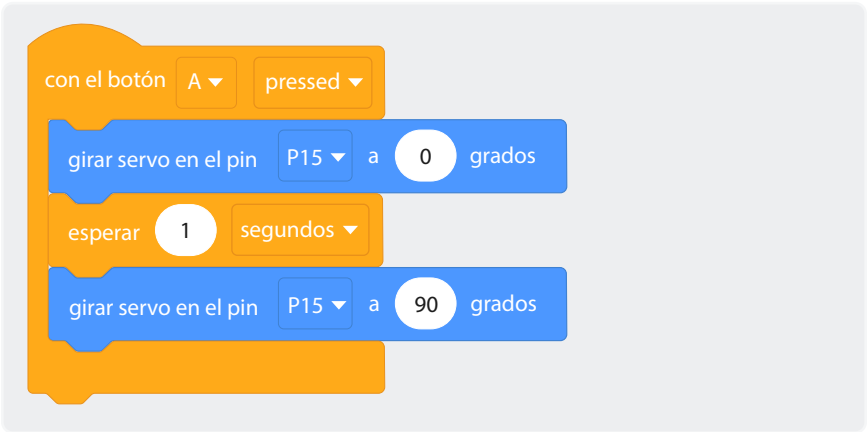
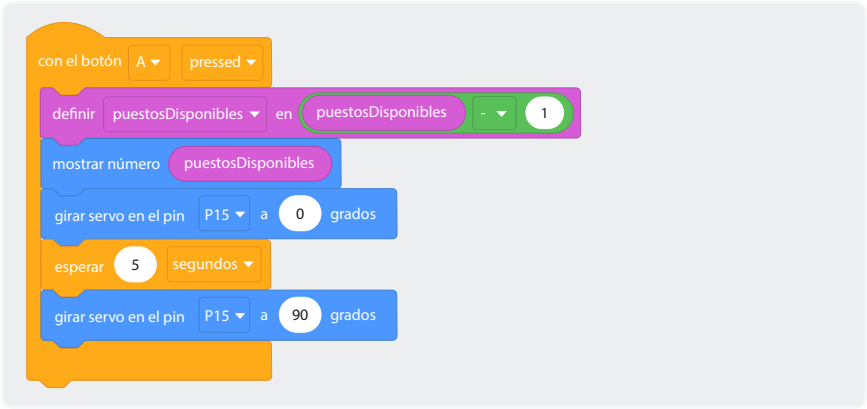


Figura 4. Programación condicional

- Implementa la lógica para decrementar la variable *puestosDisponibles*. La idea es que, al ingresar un vehículo, se disminuya en 1 los puestos disponibles.

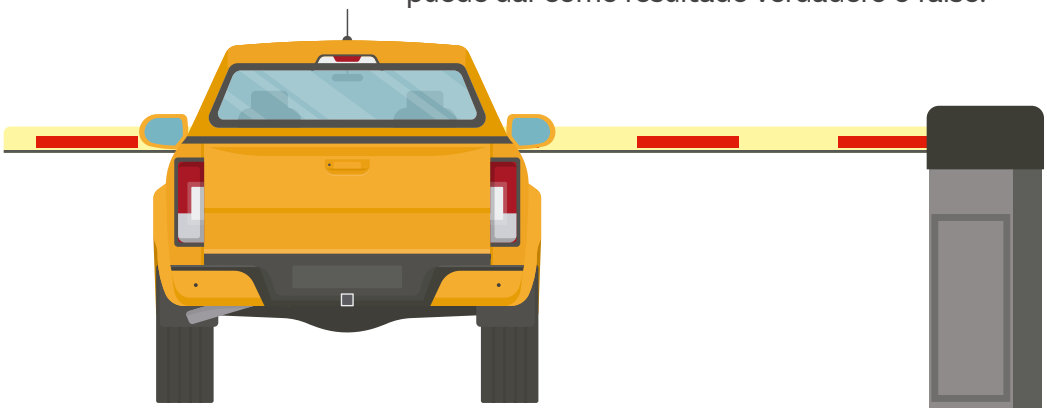
Figura 3. Programación variable



Paso 3: Manejo de casos especiales

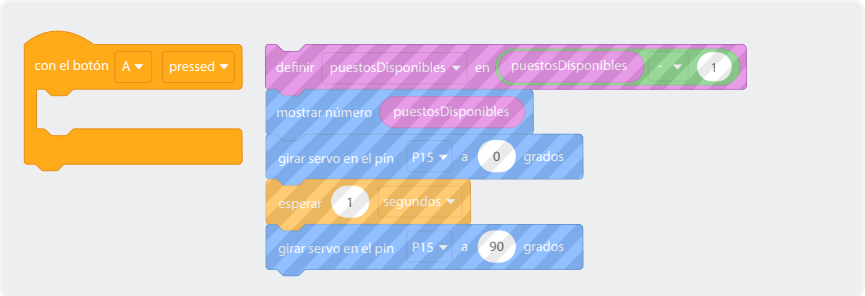
- Implementa una condición para verificar si el parqueadero está lleno.
- Muestra un ícono de X en la pantalla led cuando no hay espacios disponibles.
- Asegúrate de que la barrera no se abra cuando el parqueadero esté lleno.

Para hacerlo, vamos a usar el bloque Si – Sino que se encuentra en la opción Control. Este bloque nos permitirá ejecutar o no ejecutar una acción dependiendo de una comparación (valor booleano) que puede dar como resultado verdadero o falso.



Antes de arrastrar ese bloque al entorno de trabajo y colocarlo dentro del bloque “Botón A” de sacar los bloques que se encuentran en él. Debe quedar como se presenta en la *Figura 5.*

Figura 5. Extraer comandos de entrada de botón A



Ahora, arrastra el bloque “Si – Sino” dentro del bloque “Con el botón A” a fin de que quede como se presenta en la *Figura 6.*

Figura 6. Ingresar condicional a entrada de botón A

Acto seguido, arrastra el código que sacaste del bloque “Con el botón A” a la parte del “si no”, debe quedar como la *Figura 7.*

Figura 7. Ingresar comandos a condicional

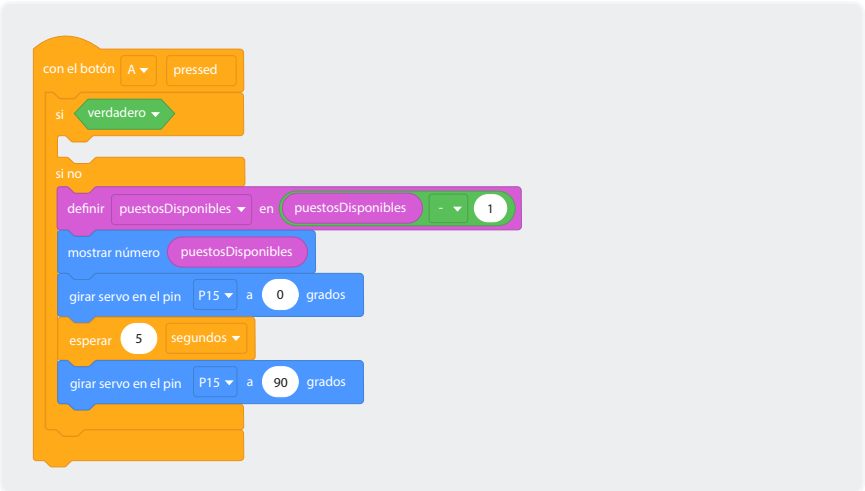
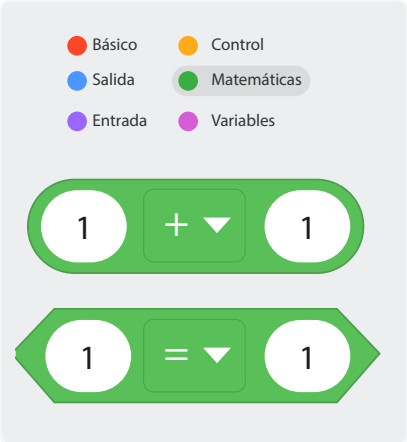


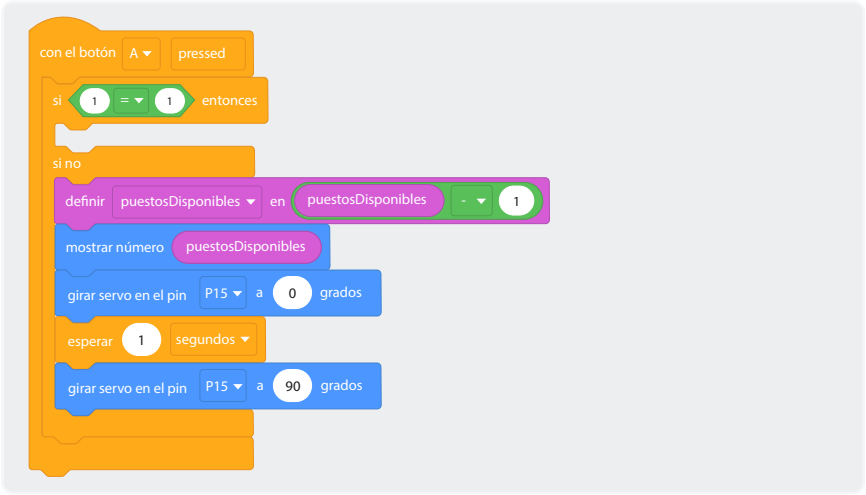
Figura 8. Utilización de operadores



Ahora procederemos a programar el condicional, para esto, iremos a la opción Matemáticas y seleccionamos el bloque que se resalta con rojo en la imagen, arrástralo al entorno de trabajo y colócalo en el bloque Si reemplazando donde dice verdadero.

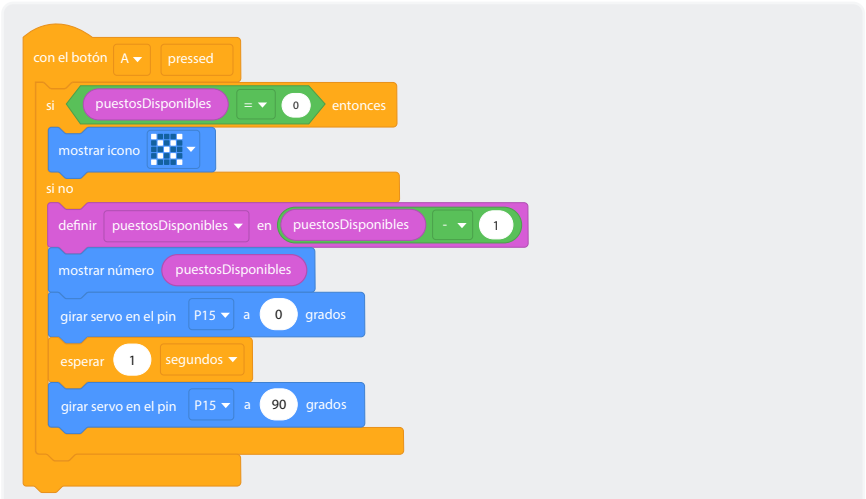
Debe quedar como se presenta en la *Figura 9.*

Figura 9. Ingresar operadores en condicional



Ahora vamos a hacer la comparación. Para esto utilizaremos la variable "puestosDisponibles" y la colocaremos en el condicional, no sin antes cambiar el 1 que está ubicado a la derecha por un 0. Este cambio permitirá que cuando la variable llegue a 0, se muestre el ícono "X" que representa que no hay más puestos en el parqueadero. El programa debe quedar como se presenta en la *Figura 10.*

Figura 10. Presentación de programa completado



Anexo

Anexo 2.1

Una vez hayas terminado podrás examinar el *Anexo 2.1* para observar otra solución al programa. Identifica por lo menos tres diferencias en los códigos y explica lo que hacen. Luego responde:

¿Ambos programas cumplen la misma función?
¿Dirías que un programa es mejor que el otro? ¿Por qué?

Prepárate para compartir tus respuestas.

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión de forma individual respondiendo las preguntas de forma que mejor reflejen tu progreso:

1. ¿Puedes implementar condicionales en la simulación de un sistema en *Tinkercad*?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

2. ¿Puedes actualizar y mostrar el contador de espacios disponibles en la pantalla led simulada?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

3 ¿Puedes reconocer y programar múltiples estados del sistema?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

Si tus respuestas a las preguntas anteriores fueron "Parcialmente" o "Aún no", regresa a simular condicionales en la simulación de TinkerCAD y reconocer múltiples estados del sistema. Si después de esto, todavía tienes dudas, acude a tu docente por apoyo adicional.

Antes de cerrar la sesión, te proponemos las siguientes preguntas. Responde y discute con tu grupo.

- ◯ ¿Qué riesgos presenta un sistema que solo usa un botón para la entrada?
- ◯ ¿Cómo podrías mejorar la seguridad sin complicar demasiado el sistema?
- ◯ Si un usuario presiona el botón, pero no entra, ¿qué consecuencias tiene para los otros usuarios?
- ◯ ¿Por qué es importante validar que el contador no baje de cero?
- ◯ ¿Qué otras variables necesitarías agregar para hacer el sistema más robusto?

Grado 11° **Guía 3**

TIC

Sesión 3

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

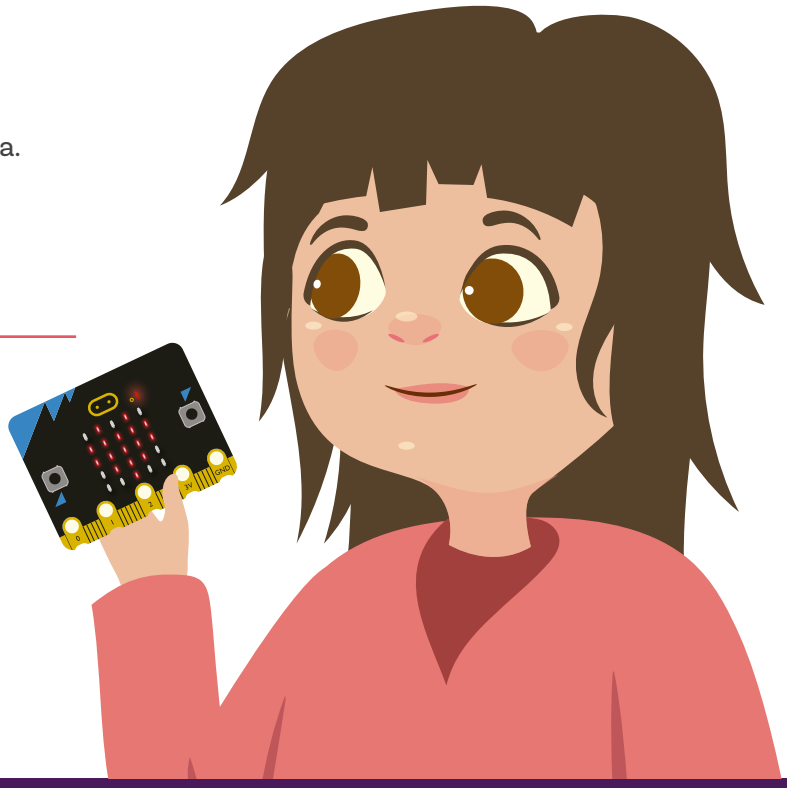
- Implementar la lógica de salida de vehículos usando el botón B de la *micro:bit* en *Tinkercad*.
- Controlar un segundo servomotor simulado para representar la barrera de salida.

Duración sugerida

20% 40% 40%

Material para la clase

- ○ Computador con acceso a Tinkercad



Enlace

Video YouTube

Lo que sabemos, lo que debemos saber

Esta sección corresponde al 20% de avance de la sesión

En la sesión anterior implementamos la lógica de entrada de vehículos. Ahora completaremos nuestro sistema usando la lógica de salida.

¿Has considerado cómo un sistema automatizado maneja la salida de vehículos de un parqueadero? ¿Qué diferencias crees que existen entre el proceso de entrada y el de salida?

Estas preguntas son fundamentales para comprender la automatización que implementaremos en nuestra *micro:bit*. La salida de vehículos implica una serie de procesos específicos como la actualización del contador de espacios disponibles y el control de la barrera, que son distintos a los de entrada.

En esta sesión, programaremos la *micro:bit* para que responda cuando se presiona el botón B, simulando la salida de un vehículo del parqueadero. Nuestro sistema deberá actualizar el contador, controlar la barrera de salida y manejar situaciones especiales.

La automatización de un parqueadero requiere un sistema preciso y confiable que pueda manejar diferentes situaciones. En el caso específico de la salida de vehículos, necesitamos asegurarnos de que nuestro programa pueda realizar varias tareas secuenciales: detectar cuando un vehículo quiere salir (mediante el botón B), actualizar el contador de espacios disponibles, controlar el mecanismo de la barrera y manejar casos especiales como cuando el parqueadero está vacío.

Manos a la obra

Conectadas

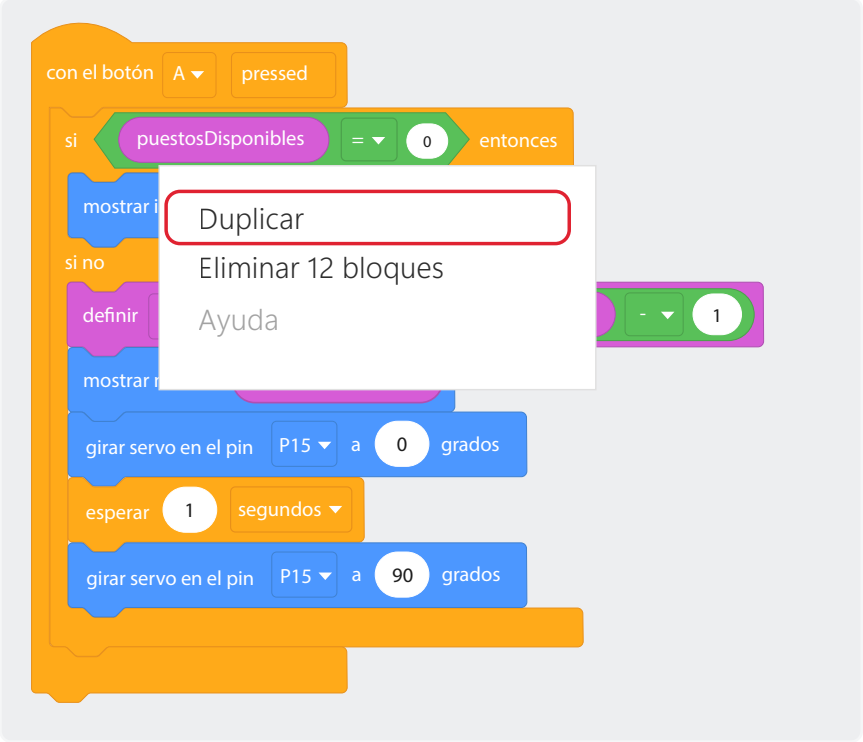
Esta sección corresponde al 60% de avance de la sesión

Sigue las indicaciones que se te dan a continuación

Paso 1: Configuración de los botones

- Ahora vamos a programar el botón “B”. Para esto vamos a reutilizar los bloques que realizamos para el botón A pero haciendo una serie de cambios. Primero, haz clic derecho sobre el bloque “Si” del bloque “Botón A” y selecciona Duplicar.

Figura 1. Instrucción para duplicar bloques



El conjunto de bloques que se duplica vas a arrastrarlo al bloque “Con el botón B”. Luego vas a realizar cambios en los ítems que están señalados para que cumpla correctamente con el funcionamiento indicado.

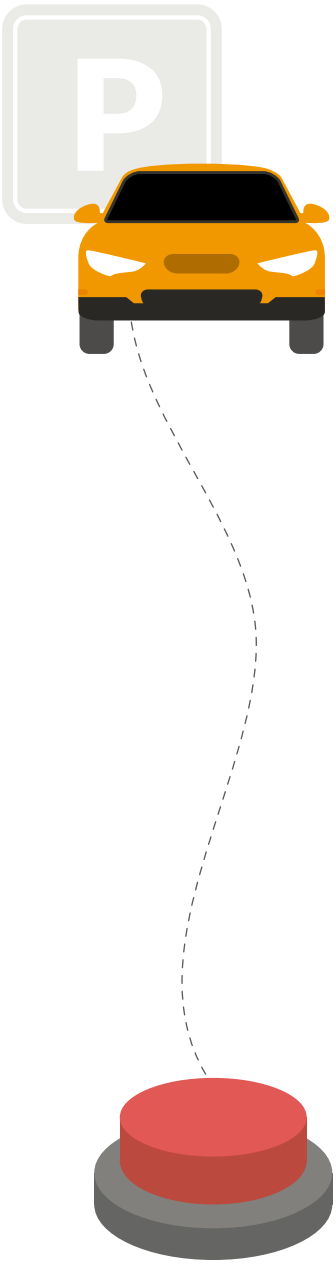
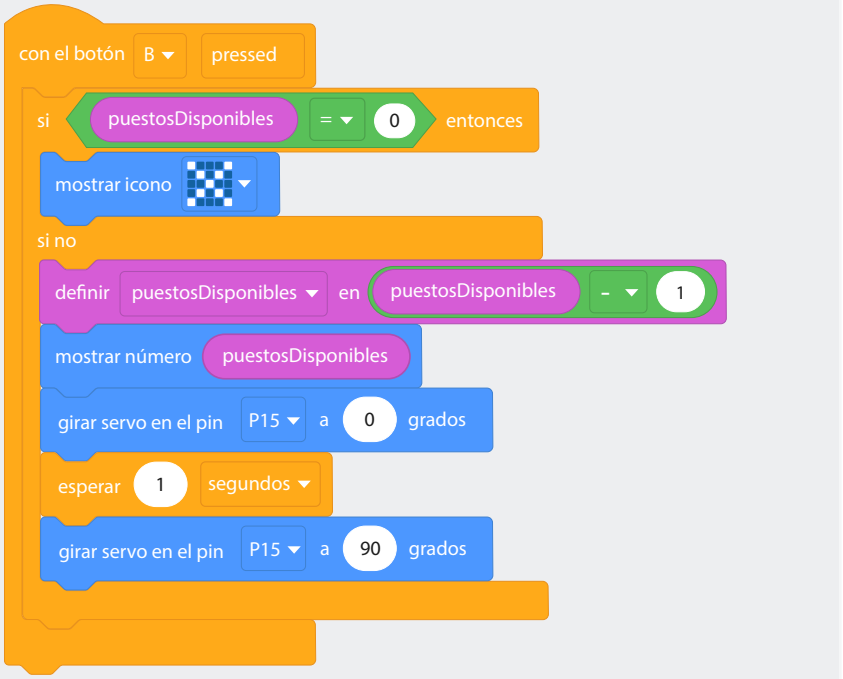


Figura 2. Instrucciones con Botón B



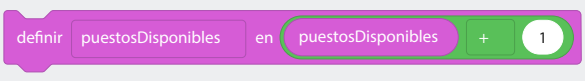
Paso 2: Lógica de salida de vehículos

- ○ Comienza por cambiar el bloque de comparación de la siguiente forma: cambia el 0 por 9 que es el número máximo de puestos del parqueadero.

Figura 3. Cambio en bloque de comparación

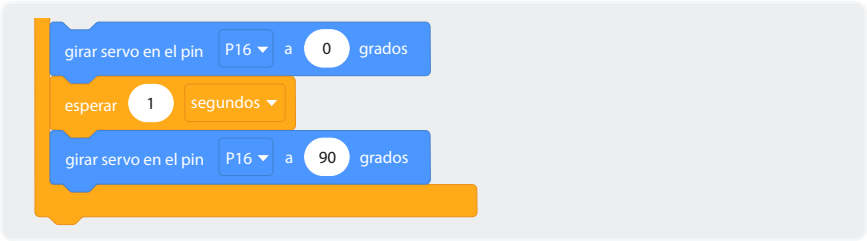
- ○ Cambia el signo "-", del bloque "definir" por el signo "+", ya que al salir un automóvil se libera un puesto en el parqueadero.

Figura 4. Definir valor de variable



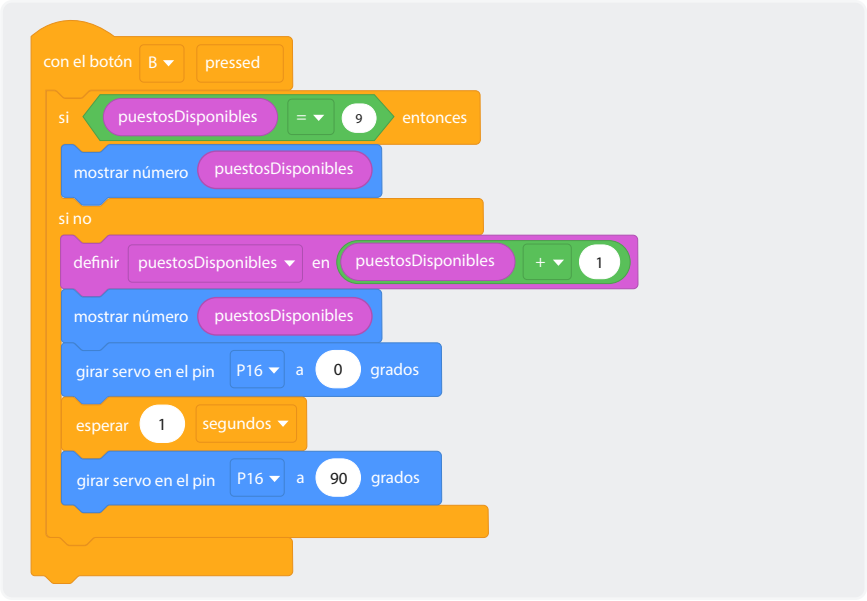
- ○ Cambia los pines de P15 a P16 para que se controle el servomotor correspondiente.

Figura 5. Configuración de pines



- ○ Actualiza la pantalla led con el nuevo número de espacios disponibles.

Figura 6. Actualización de pantalla led

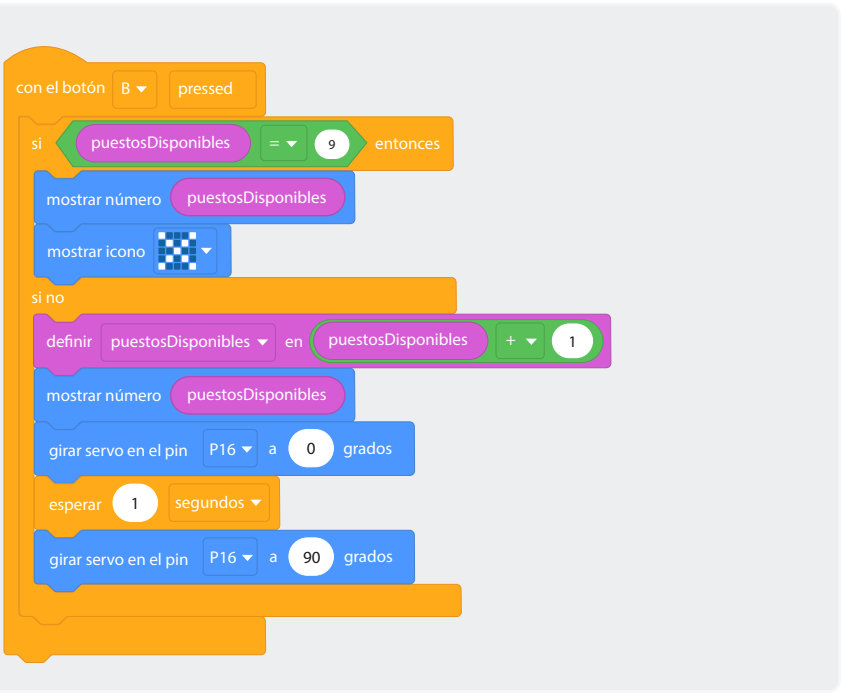


Paso 3: Manejo de casos especiales

- ○ Muestra un mensaje “X” cuando se intenta salir de un parqueadero vacío.

- ○ Asegúrate de que la barrera no se abra cuando no hay vehículos para salir. Como puedes darte cuenta, las instrucciones tanto del botón A como del botón B son muy similares.

Figura 7. Programación de mensaje cuando se intenta salir de un parqueadero vacío



Realiza pruebas a tu código.

¿Se comporta como debería?
¿Qué mejoras se te ocurren?

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

De forma individual, regresa a revisar los aprendizajes de la sesión. Elije la opción de respuesta que mejor describa lo que alcanzaste.

1. ¿Puedes implementar la lógica completa de entrada y salida de vehículos en *Tinkercad* usando la *micro:bit*?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

2. ¿Puedes controlar múltiples servomotores simulados para representar diferentes barreras?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

3. ¿Puedes manejar y mostrar el estado del parqueadero en diversas situaciones?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

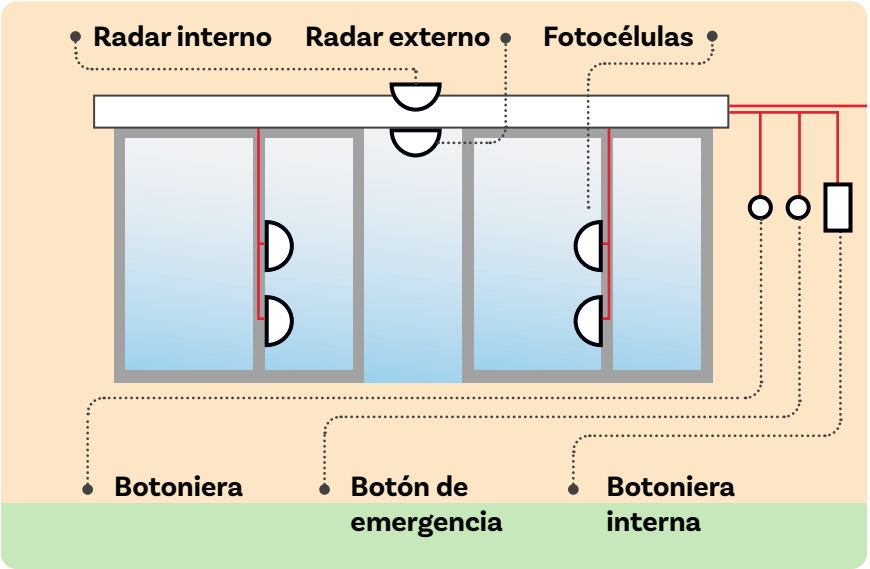
Hasta el momento has simulado el sistema de entrada y salida a un parqueadero pero...

¿Cómo te lo imaginas?

Utiliza este espacio para realizar un dibujo o diagrama donde señales los diferentes elementos que has utilizado.

A continuación, te presentamos un ejemplo en la *Figura 8* de una puerta peatonal automática que puede servirte de guía para realizar tu propio diagrama.

Figura 8. Puerta peatonal automática



Sesión 4

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

- Optimizar el código para mejorar su eficiencia y legibilidad.
- Implementar características adicionales al sistema de parqueadero.
- Utilizar técnicas avanzadas de programación en *micro:bit* con *Tinkercad*.
- Evaluar y mejorar el diseño general del sistema.

Duración sugerida

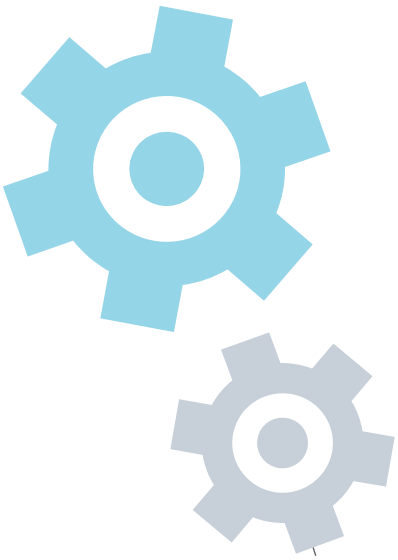
20% 40% 40%

Material para la clase

- ○ Anexo 5.1

Lo que sabemos, lo que debemos saber

Esta sección corresponde al 20% de avance de la sesión



En las sesiones anteriores, hemos creado un sistema funcional de parqueadero inteligente. Ahora vamos a llevarlo al siguiente nivel, optimizando el código y añadiendo nuevas características.

> *¿Has pensado en cómo los sistemas reales se mejoran con el tiempo? ¿Qué características adicionales crees que serían útiles en un sistema de parqueadero inteligente?*

En esta sesión final optimizaremos nuestro código, añadiremos nuevas funcionalidades y reflexionaremos sobre cómo nuestro proyecto podría escalar a un sistema real.

Para profundizar en el desarrollo de nuestro sistema de parqueadero, debemos orientar nuestros esfuerzos hacia la optimización y la escalabilidad. Un sistema de parqueadero inteligente en el mundo real requiere características avanzadas como análisis de patrones de uso, predicción de disponibilidad, integración con sistemas de pago y comunicación con otros dispositivos. Esta sesión nos permitirá no solo mejorar nuestro código actual, sino también visualizar cómo nuestro proyecto podría evolucionar hacia un sistema más sofisticado y completo. La reflexión sobre estas mejoras nos ayudará a comprender mejor cómo los sistemas reales se desarrollan y adaptan a necesidades cambiantes.

Manos a la obra

Conectadas

Esta sección corresponde al 60% de avance de la sesión

A continuación, encontrarás una serie de instrucciones que te ayudarán a seguir aprendiendo.

Paso 1: Optimización del código

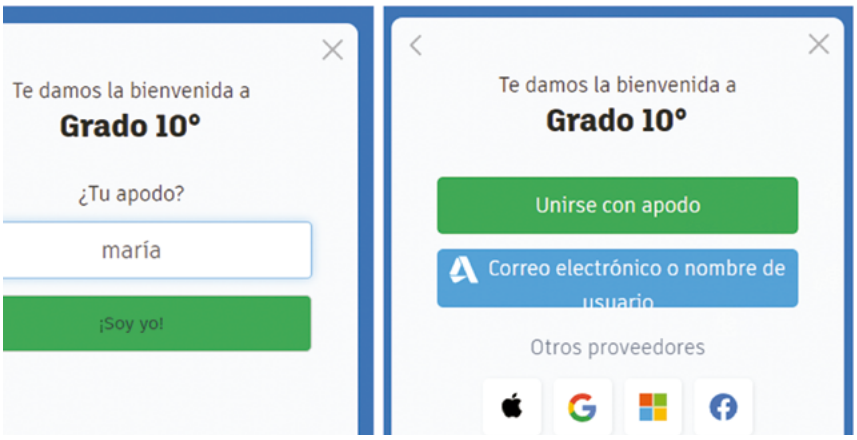
Revisa el código actual e identifica áreas de mejora.

Paso 2: Implementación de nuevas características

Implementa un sistema de alarma visual con dos leds (se recomiendan naranja y rojo). El led naranja debe encender cuando el número de puestos disponibles sea menor o igual a 3 y el led rojo debe encender cuando el número de puestos disponibles sea 0.

Para hacerlo, debes colocar primero los 2 leds y conectarlos tal como se muestra en la *Figura 1*.

Figura 1. Conexiones de Leds



Como puedes notar, el led rojo está conectado al (P1) y el led naranja está conectado al (P2). Asegúrate de mantener el código de colores rojo para positivo y negro para negativo, de esta manera tu diseño será más fácil de entender.

Figura 2. Bloque de control - Siempre

Figura 3. Bloques condicionales "si"

Luego de realizar las conexiones, vamos a programar lo necesario para que estos leds realicen su función. Para esto, debes agregar el bloque Siempre que se ubica en las opciones de Control, luego agrégalo a tu entorno de trabajo como se muestra en la *Figura 2*.

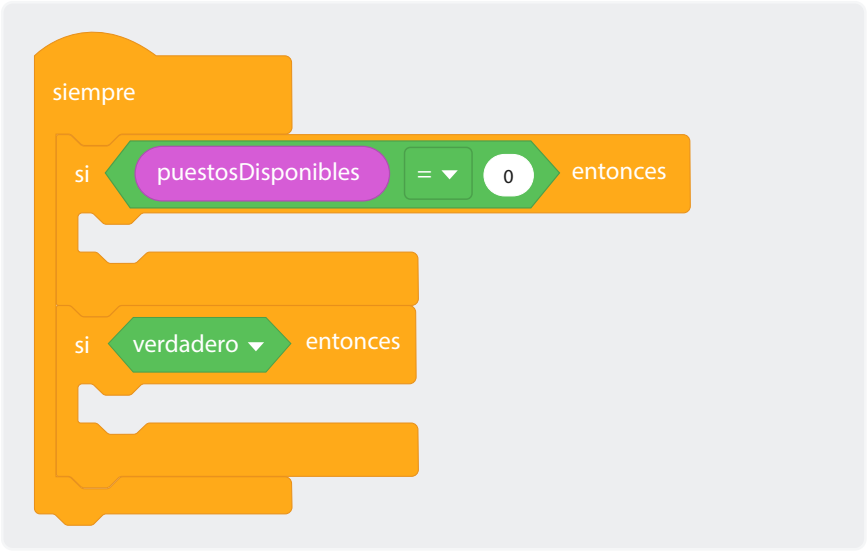
Tras haber agregado el bloque "Siempre" procede a programarlo para que cumpla con las funcionalidades que se propusieron para cada led. Sigue las instrucciones.

Dentro del bloque "siempre" vas a agregar 2 bloques "si", recuerda que estos bloques los encuentras en la opción Control. Colócalos uno encima del otro, tal como muestra la *Figura 3*.

Ahora procederemos a colocar las comparaciones necesarias. Para esto, ten en cuenta las restricciones que se dieron con anterioridad, el led rojo encenderá cuando la cantidad de puestos (puestosDisponibles) sea igual a 0 y el led naranja encenderá cuando los puestos disponibles estén entre 1 y 3.

Inicialmente, arrastra un bloque de comparación al bloque "si" este bloque lo encuentras en la opción "Matemáticas" tal como lo hiciste en la sesión 3. Agrégale la variable "puestosDisponibles" e iguálala a 0 como muestra la *Figura 4*.

Figura 4. Declaración de variable



Luego, al siguiente bloque "si" agrégale el operador lógico "y" que encuentras en la opción Matemáticas.

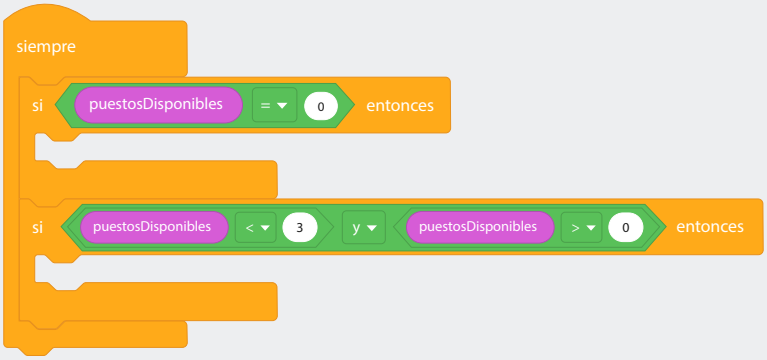
Figura 5. Operador lógico

Después cambia cada comparación por la condición necesaria, en este caso es que “puestosDisponibles” sea menor que 3 y “puestosDisponibles” mayor que 0. El operador debe quedar como en la *Figura 6.*

Figura 6. Comparación con operadores lógicos

Y el bloque completo como la *Figura 7.*

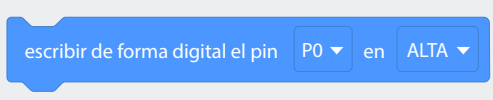
Figura 7. Visualización completa de condicionales



Tras agregar el bloque e insertarle los condicionales “si”, vamos a colocar el código necesario para que los leds hagan su función.

Recuerda que los leds que colocaste están conectados a P1 (rojo) y P2 (naranja), para que estos enciendan vamos a utilizar una señal digital que puede ser Alta para encendido o Baja para apagado. Para implementar esto, debes ir a la opción Salida y buscar el bloque “Escribir de forma digital el pin”.

Figura 8. Programación de escritura digital en pin

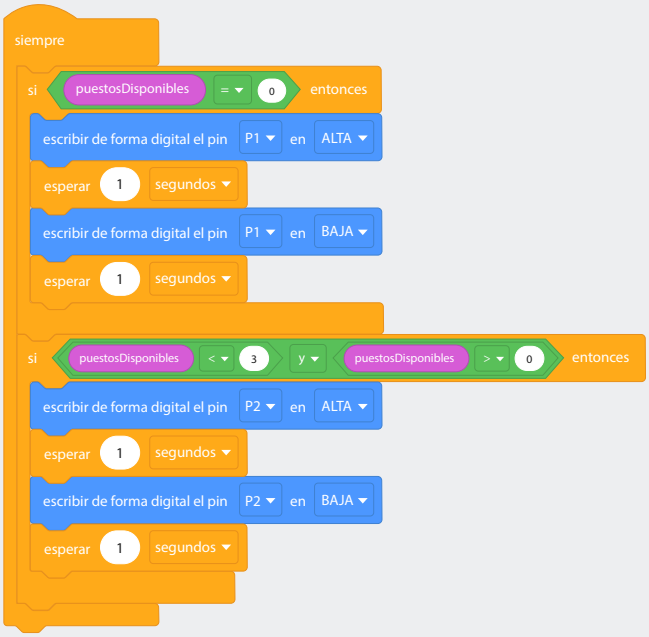


Además, utilizaremos un bloque de espera para crear la sensación de parpadeo. Este bloque lo encuentras en la opción Control.

Figura 9. Programación de bloque de espera

Organízalos como se muestra en la *Figura 10*.

Figura 10. Visualización de programa completo



Paso 3: Pruebas y refinamiento final

Realiza pruebas de todas las funcionalidades.

> *¿El programa se comporta como esperas?*

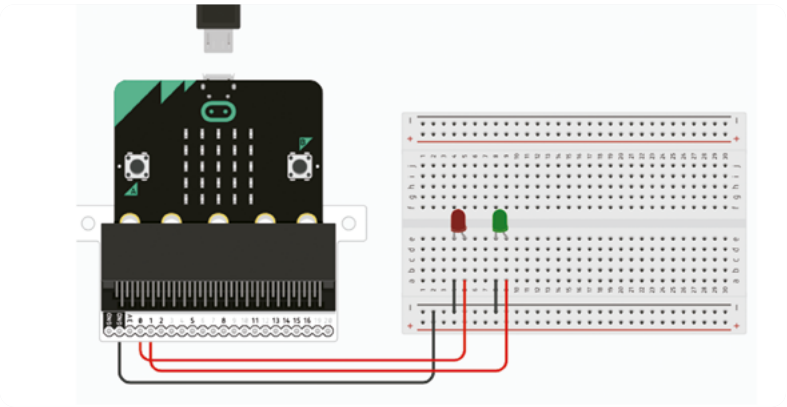
Ajusta el código según sea necesario basándote en los resultados de las pruebas.

Figura 12. Escrituras digitales en pines

Para ir más lejos

Para ir más lejos puedes explorar cómo colocar luces en tus circuitos y controlarlos con *micro:bit*, realiza este el montaje de la *Figura 11.*

Figura 11. Control de leds con *micro:bit*



Ahora vamos a programarlos para que alternen de tal manera que encienda el led rojo por 1 segundo, luego se apague y encienda el verde por 1 segundo y se apague. Utiliza este conocimiento para aplicarlo en el reto.

Como te puedes dar cuenta, hay una equivalencia entre la programación por bloques y la programación en lenguaje *Python*. Analiza cada bloque y compáralo con cada línea para que encuentres la lógica.

```
# Python code
#
def on_forever():
  pins.digital_write_pin(DigitalPin.P0, 1)
  pins.digital_write_pin(DigitalPin.P0, 0)
  basic.pause(1000)
  pins.digital_write_pin(DigitalPin.P1, 1)
  pins.digital_write_pin(DigitalPin.P1, 0)
  basic.pause(1000)
basic.forever(on_forever)
```

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

De forma individual, regresa a revisar los aprendizajes de la sesión. Elije la opción de respuesta que mejor describa lo que alcanzaste.

1. ¿Puedes optimizar y extender código existente en *micro:bit* usando *Tinkercad*?
   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no
2. ¿Puedes implementar características avanzadas en sistemas simulados?
   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no
3. ¿Puedes evaluar las implicaciones de escalar un proyecto a un sistema real?
   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

Para cerrar la sesión, te proponemos reflexionar y discutir las siguientes preguntas:

¿Cómo ha evolucionado tu comprensión de los sistemas automatizados a lo largo de este proyecto?
¿Qué desafíos crees que enfrentarías al implementar este sistema en un parqueadero real?
Pensando en el reto central, ¿cómo crees que las habilidades que has desarrollado te preparan para futuros proyectos de tecnología?

Sesión 5

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

- Refinar el código existente para mejorar su eficiencia y legibilidad.
- Implementar funciones auxiliares para reducir la repetición de código.
- Realizar pruebas exhaustivas del sistema en diferentes escenarios.

Duración sugerida

20% 40% 40%

Material para la clase

- ○ Anexo 5.1

Enlace

Acerca de la Agenda 2030 para el Desarrollo Sostenible

Lo que sabemos, lo que debemos saber

Esta sección corresponde al 20% de avance de la sesión

En las sesiones anteriores hemos implementado las funcionalidades básicas de nuestro sistema de parqueadero. Ahora es momento de refinar nuestro código y asegurarnos de que funcione correctamente en todas las situaciones posibles.

¿Has pensado en cómo los sistemas reales se prueban antes de ser implementados? ¿Qué tipo de situaciones excepcionales podrían ocurrir en un parqueadero real que nuestro sistema debería manejar?

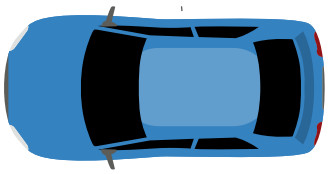
Después de haber implementado el código básico de nuestro sistema de parqueadero, es momento de fortalecer nuestro código para manejar situaciones del mundo real. Un sistema de parqueadero debe ser capaz de manejar diversos escenarios excepcionales como fallos en los sensores, interrupciones de energía, o intentos de salida cuando el parqueadero está vacío. Para lograr esto, necesitaremos implementar códigos auxiliares que nos ayuden a detectar y manejar estos casos especiales, realizar pruebas exhaustivas de cada componente, y asegurarnos de que nuestro sistema responda apropiadamente en todas las situaciones posibles.

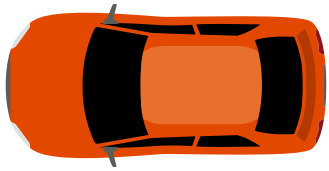
Manos a la obra Conectadas

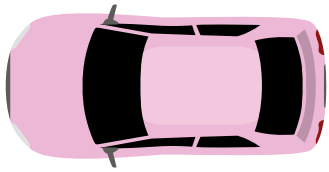
Esta sección corresponde al 60% de avance de la sesión

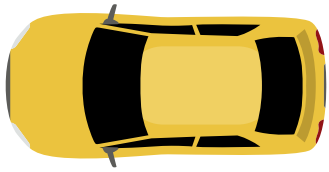
Paso 1: Pruebas al software

Antes de poner en funcionamiento cualquier sistema, necesitamos asegurarnos de que todo funciona como esperamos. Es como cuando revisas tu tarea antes de entregarla: quieres estar seguro de que todo está bien.









Para lograrlo, realiza la siguiente batería de pruebas:

Prueba 1: Prueba de Inicio
Imagina que acabas de encender el sistema por primera vez:

¿Qué número aparece en la pantalla? Debe mostrar "9"
¿Las barreras están cerradas?
Los servomotores deben estar en 90°.
¿Todas las luces están apagadas?
Los pines no deben estar enviando señales.

Verifica lo valores y completa la tabla 1:

Tabla 1. Verificación de elementos

Elemento por verificar	Estado Esperado	Estado Real	¿Funciona? (Sí/No)
Número en pantalla	9		
Barrera Derecha	90°		
Barrera Izquierda	90°		
Luces de alerta	Apagadas		

Prueba 2. Probando la entrada de carros
Vamos a simular que llegan algunos carros:

Presiona el botón A (simula que llega un carro).
Observa:

¿El número en la pantalla bajó de 9 a 8?
¿La barrera se abrió (servo a 0°) y luego se cerró (servo a 90°)?
¿Todo el proceso tomó aproximadamente 1 segundo?

Prueba 3. Prueba cuando está lleno
Presiona el botón A varias veces hasta que la pantalla muestre 0.
Intenta presionar el botón A una vez más.
Observa:

¿Aparece una "X" en la pantalla?
¿La barrera se quedó quieta?
¿La luz roja de advertencia (pin P1) está parpadeando?

Prueba 4. Probando la salida de carros
Ahora simulemos que los carros salen:

Con algunos carros adentro (por ejemplo, 5 espacios disponibles), presiona el botón B (simula que sale un carro).
Observa:

¿El número aumentó de 5 a 6?
¿La barrera de salida se abrió y cerró correctamente?
¿El proceso tomó el tiempo correcto?

Prueba 5. Prueba cuando está vacío
Espera a que la pantalla muestre 9 (parqueadero vacío).
Presiona el botón B.
Observa:

¿Se muestra la X después del 9?
¿La barrera se quedó quieta?

Prueba 6. Probando las luces de advertencia
Es importante verificar que las alertas funcionan.

Llena el parqueadero (presiona A hasta llegar a 0).
Observa:

¿La luz roja (pin P1) parpadea cada segundo?

Ahora deja solo 1 o 2 espacios libres.
Observa:

¿La luz amarilla (pin P2) parpadea cada segundo?

Llena la siguiente tabla con tus observaciones:

Situación	Estado luz roja	Estado luz naranja	Funciona
Parqueadero lleno			
2 espacios libres			
1 espacio libre			
Normal (>2 espacios)			

Basado en los resultados obtenidos con estas pruebas y la lógica propuesta para la realización del parqueadero, ¿qué puedes concluir del funcionamiento de tu programa?

Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión y autoevalúa el grado al que los has alcanzado.

1. ¿Puedes reconocer y programar múltiples estados del sistema?
   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

2 ¿Puedes implementar pruebas para sistemas simulados en *Tinkercad*?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

3 ¿Puedes manejar casos límite y situaciones excepcionales en sistemas automatizados?

- ◯ Sí
- ◯ Parcialmente
- ◯ Aún no

Aprovecha este espacio final para hacer un mapa conceptual en el que muestres algo de lo que aprendiste, por ejemplo, colocando las definiciones de palabras y algún ejemplo de uso de los tipos de variables empleadas.

Por último, te proponemos un reto con tu grupo usando una rutina llamada Pensar, Presentar e Integrar (P-P-I):

Primero respondemos individualmente, luego, cada persona en el grupo y en su turno le presenta al resto del equipo sus respuestas.

Finalmente, el grupo integra una respuesta unificada.
Las preguntas que te proponemos:

¿?

¿Cómo ha cambiado tu enfoque de programación después de esta sesión de refinamiento y pruebas?

¿Qué aspectos del desarrollo de software crees que son más importantes: la funcionalidad inicial o el refinamiento y las pruebas?

Pensando en el reto central, ¿cómo crees que las mejoras implementadas hoy han fortalecido tu sistema de parqueadero inteligente?

Anexo 1.1 Reto

Tu reto es crear un sistema de parqueadero inteligente utilizando una *micro:bit* como unidad de control central. El sistema debe gestionar un parqueadero con 9 espacios disponibles inicialmente. Utilizarás dos servomotores conectados a los pines P15 y P16 de la *micro:bit* para simular las barreras de entrada y salida respectivamente. La pantalla led integrada de la *micro:bit* se usará para mostrar constantemente el número de espacios disponibles. Los botones A y B de la *micro:bit* simularán la llegada y salida de vehículos: el botón A representará un vehículo entrando, mientras que el botón B representará un vehículo saliendo.

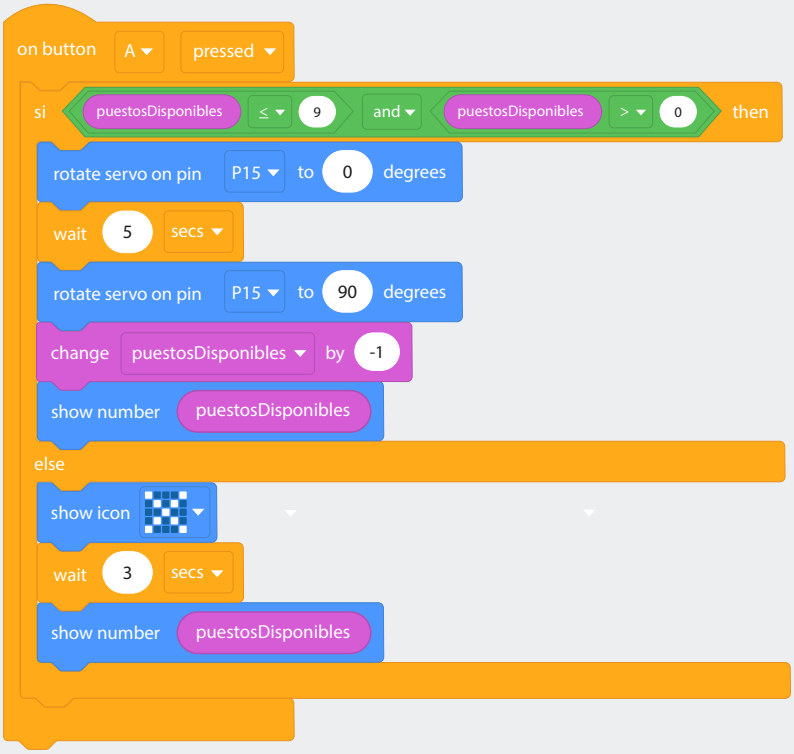
El sistema debe ser capaz de:

1. Disminuir el contador de espacios y abrir la barrera de entrada (mover el servomotor en P15 a 0 grados) cuando se presiona el botón A, siempre y cuando haya espacios disponibles.
2. Aumentar el contador de espacios y abrir la barrera de salida (mover el servomotor en P16 a 0 grados) cuando se presiona el botón B, siempre que el parqueadero no esté vacío.
3. Mostrar un ícono de No en la pantalla led cuando se intente ingresar a un parqueadero lleno o salir de un parqueadero vacío.
4. Mantener las barreras abiertas por 5 segundos antes de cerrarlas (volver a 90 grados).
5. Actualizar y mostrar el número de espacios disponibles después de cada operación. Tu reto incluye implementar esta lógica, asegurando que el sistema funcione de manera fluida y maneje correctamente todos los casos posibles, incluidos los casos límite cuando el parqueadero está lleno o vacío.

Para ir más lejos: adicionalmente, el sistema debe incorporar un conjunto de led para mejorar la señalización visual. Se utilizarán cuatro led en total: un par de led (rojo y verde) para cada barrera (entrada y salida). El led rojo permanecerá encendido mientras la barrera correspondiente esté abajo y se apagará cuando la barrera esté arriba. El led verde, por el contrario, se encenderá cuando la barrera esté arriba y se apagará cuando esté abajo. Esta característica proporcionará una indicación visual clara del estado de cada barrera, mejorando la experiencia del usuario y la seguridad del sistema.

Anexo 2.1 – Una posible solución

Versión en bloques de *Tinkercad*:



Versión en *Python*:

```
def on_button_pressed_a():
 global puestosDisponibles
 if puestosDisponibles <= 9 and puestosDisponibles > 0:
  pins.servo_write_pin(AnalogPin.P15, 0)
  basic.pause(5000)
  pins.servo_write_pin(AnalogPin.P15, 90)
  puestosDisponibles += -1
  basic.show_number(puestosDisponibles)
 else:
  basic.show_icon(IconNames.No)
  basic.pause(3000)
  basic.show_number(puestosDisponibles)
input.on_button_pressed(Button.A, on_button_pressed_a)
```

Anexo 5.1 – Posible solución Sesión 4

Optimización del código
Revisa el código actual e identifica áreas de mejora.
Genera el código necesario para manejar la lógica común de entrada y salida.

Implementación de nuevas características
Añade un contador de tiempo para simular el cobro por tiempo de estacionamiento.
Implementa un sistema de alarma cuando el parqueadero está casi lleno.

Pruebas y refinamiento final
Realiza pruebas exhaustivas de todas las funcionalidades.
Ajusta el código según sea necesario basándote en los resultados de las pruebas.

```
# Python code
#
MAX_ESPACIOS = 9
CASI_LLENO = 2
puestosDisponibles = MAX_ESPACIOS
tiempoTotal = 0
basic.show_string("" +
str(puestosDisponibles))
pins.servo_write_pin(AnalogPin.P15, 90)
pins.servo_write_pin(AnalogPin.P16, 90)

def mover_barrera(pin, abrir):
    angulo = 0 if abrir else 90
    pins.servo_write_pin(pin, angulo)
    basic.pause(5000)
def manejar_vehiculo(es_entrada):
    global puestosDisponibles, tiempoTotal
    if es_entrada and puestosDisponibles > 0:
        mover_barrera(AnalogPin.P15, True)
        puestosDisponibles += -1
        basic.show_arrow(ArrowNames.NORTH)
    elif not es_entrada and puestosDisponibles
< MAX_ESPACIOS:
        mover_barrera(AnalogPin.P16, True)
        puestosDisponibles += 1
        basic.show_arrow(ArrowNames.SOUTH)
        tiempoTotal += 1  # Simula cobro por
tiempo
    else:
        basic.show_icon(IconNames.NO)
        basic.pause(3000)
    actualizar_display()
    if puestosDisponibles <= CASI_LLENO:
        basic.show_icon(IconNames.DIAMOND)
        basic.pause(1000)

def on_button_pressed_a():
    manejar_vehiculo(True)

def on_button_pressed_b():
    manejar_vehiculo(False)

input.on_button_pressed(Button.A, on_
button_pressed_a)
input.on_button_pressed(Button.B, on_
button_pressed_b)

def on_forever():
    global tiempoTotal
    basic.pause(60000)  # Espera 1 minuto
    tiempoTotal += 1
    if tiempoTotal % 5 == 0:  # Cada 5 minutos
        basic.show_string("T:" + str(tiempoTotal))
        basic.pause(1000)
        actualizar_display()

basic.forever(on_forever)

    pins.servo_write_pin(pin, 90)

def actualizar_display():
    basic.show_number(puestosDisponibles)
```

TIC

Apoya:

Educación

BRITISH COUNCIL

Colombia Programa
{EL CÓDIGO A TU FUTURO}