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

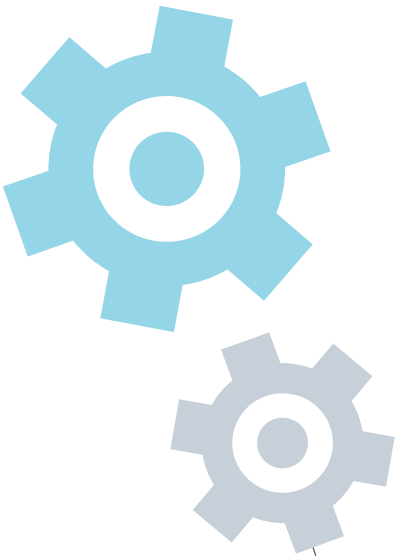
20% 40% 40%

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

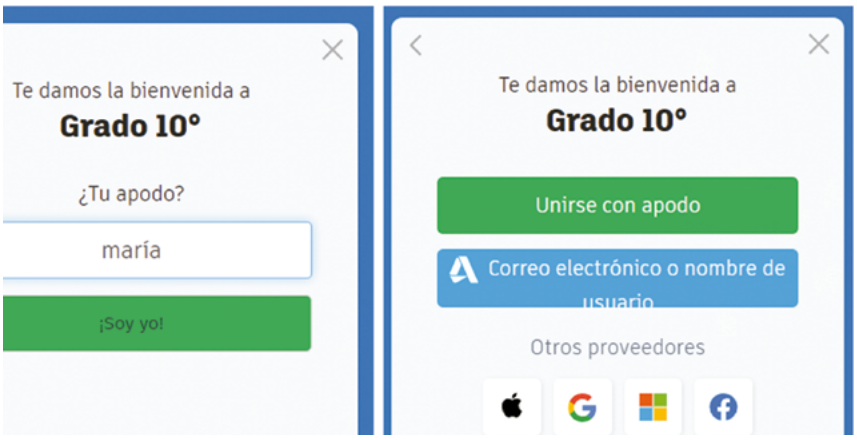
Revisa el código actual e identifica áreas de mejora.

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*.

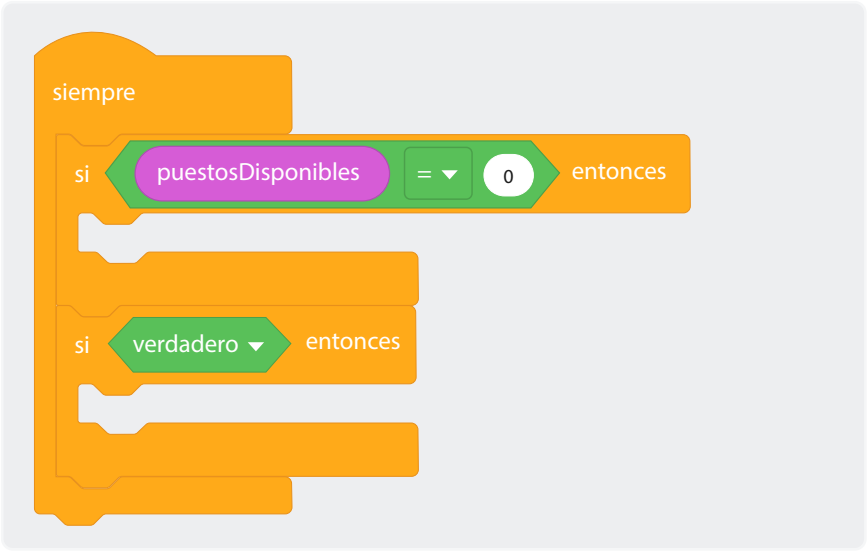
Tras haber agregado el bloque "Siempre" procede a programarlo para que cumpla con las funcionalidades que se propusieron para cada led. Sigue las instrucciones.

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.

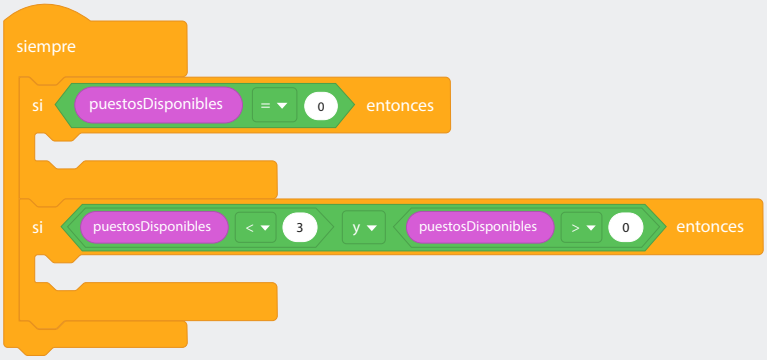
Figura 5. Operador lógico

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.*

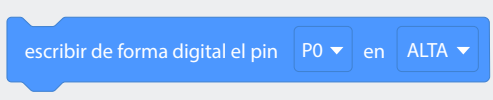
Figura 7. Visualización completa de condicionales



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

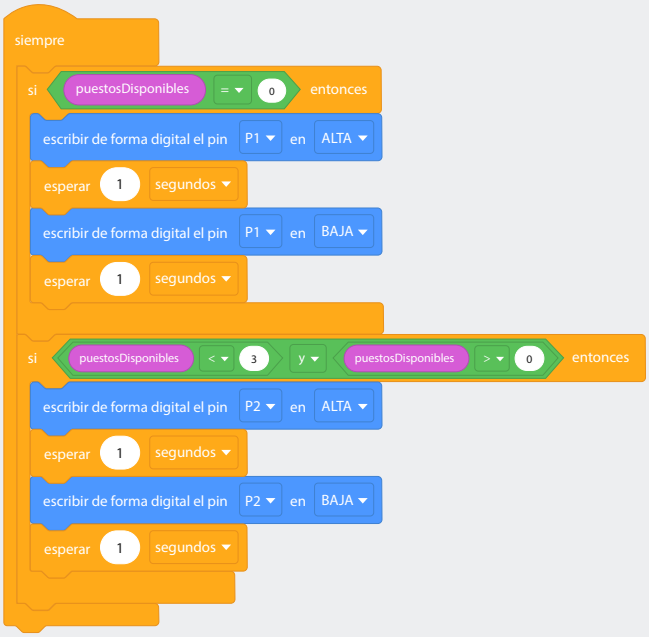


Además, utilizaremos un bloque de espera para crear la sensación de parpadeo. Este bloque lo encuentras en la opción Control.

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?*

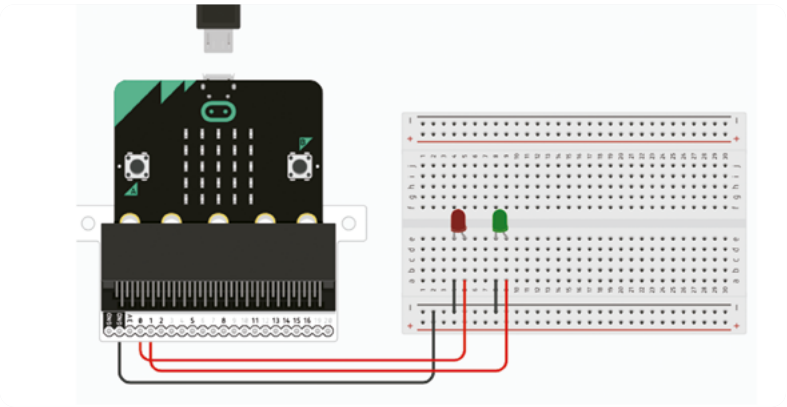
Ajusta el código según sea necesario basándote en los resultados de las pruebas.

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?