CONECTANDO CON LA MICRO:BIT

ESCRITO POR

Cigdem Sengul

&

Anthony Kirby

TRADUCIDO POR

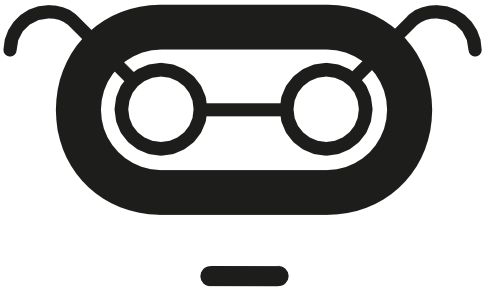
Josefina Cibils

micro:bit

NOMINET

Autores: Cigdem Sengul y Anthony Kirby

https://microbit.nominetresearch.uk/networking-book/



CONTENIDO

Prefacio 8

Acerca del libro 8

Acerca de los autores 9

Agradecimientos 9

Recursos y Actualizaciones 9

Resumen 9

1 Comunicación por cables 11

1. Introducción 11

1.2 Lo que necesitarás 11

1.3 Contexto 11

1.4 Programación: “Transferencia Simple de Corazón” 12

1.5 Tarea 1: Mira la “Transferencia Simple de Corazón” 13

1.6 Tarea 2: Conecta tu micro:bit y prueba el telégrafo 13

1.7 Tarea 3: Prueba los archivos Hex "Transferencia Simple de Corazón " 14

1.8 Tarea 4: Programa una “transferencia de corazón” 14

1.9 Actividad extensiva 15

1.10 Problemas 15

1.11 Recursos 15

2 Comunicación inalámbrica y de broadcast 17

2. Introducción 17

2.2 Lo que necesitarás 17
2.3 Contexto 17
2.4 Programación: Recibe y envía mensajes de broadcast 19
2.5 Tarea 1: Configura tu radio 19
2.6 Tarea 2: Recibe un mensaje de broadcast 20
2.7 Tarea 3: Envía un mensaje de broadcast 20
2.8 Actividad extensiva 21
2.9 Problemas 21
2.10 Recursos 21

3 Comunicación grupal 23
3. Introducción 23
3.2 Lo que necesitarás 23
3.3 Contexto 23
3.4 Programación: Crear grupos y mensajes dentro de los grupos 24
3.5 Tarea 1: Crea grupos 25
3.6 Tarea 2: Envía y recibe mensajes de grupo 25
3.7 Actividad extensiva 25
3.8 Problemas 25
3.9 Recursos 26

4 Juego 1: Shakey Donkey 28
4. Introducción 28
4.2 Lo que necesitarás 28
4.3 Programación: Juega a Shakey Donkey 28
4.4 Problemas 30

5 Comunicación Unicast: Uno a Uno 32
5. Introducción 32
5.2 Lo que necesitarás 32
5.3 Contexto 32
5.4 Programación: Envía y recibe mensajes Unicast 34
5.5 Tarea 1: Configura tu radio 34
5.6 Tarea 2: Diseña tu encabezado 34
5.7 Tarea 3: Crea tu paquete y envíalo 35
5.8 Tarea 4: Recibe tu paquete 35
5.9 Desafío: Filtro de remitentes 35

5.10 Actividad extensiva **35**
5.11 Problemas **36**
5.12 Recursos **36**

6 Unicast bidireccional 38
6. Introducción **38**
6.2 Lo que necesitarás 38
6.3 Contexto **38**
6.4 Programación: Ping 41
6.5 Tarea 1: Prepárate para Unicast 41
6.6 Tarea 2: Envía un Ping 41
6.7 Tarea 3: Recibe un Ping 41
6.8 Tarea 4: Recibe un Pong y calcula el tiempo de ida y vuelta 41
6.9 Ejercicios **41**
6.10 Problemas **42**
6.11 Recursos **42**

7 Juego 2: Piedra, papel, tijeras por la radio 44
7. Introducción **44**
7.2 Lo que necesitarás 45
7.3 Programación: Piedra, papel, tijeras **45**
7.4 Tarea 1: Comienza con el juego simple 45
7.5 Tarea 2: Trabajando con la radio por Unicast 45
7.6 Tarea 3: Completa la tabla de reglas 45
7.7 Tarea 4: Juega 46
7.8 Ejercicios **47**
7.9 Problemas **47**
7.10 Recursos **47**

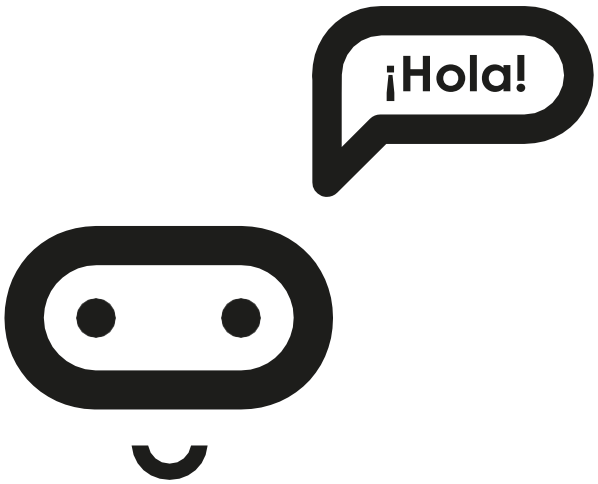
8 Manejo de Errores: Retransmisiones 49
8. Introducción **49**
8.2 Lo que necesitarás 49
8.3 Contexto **49**
8.4 Programación: Retransmisiones **51**
8.5 Tarea 1: Crea paquetes de errores 51
8.6 Tarea 2: Envía una secuencia de mensajes 51
8.7 Tarea 3: Retransmisión por defecto 52

8.8 Actividad extensiva 52
8.9 Problemas 53
8.10 Recursos 53

9 Manejo de Errores: Acuse de recibo 55
9. Introducción 55
9.2 Lo que necesitarás 55
9.3 Contexto 55
9.4 Programación: ¡Para y Espera! 58
9.5 Tarea 1: Diseña tus datos y tus paquetes ACK 58
9.6 Tarea 2: Tiempo de espera y retransmisión 58
9.7 Tarea 3: Prueba la confiabilidad de *Parar y Esperar* 58
9.8 Actividad extensiva 59
9.9 Problemas 59
9.10 Recursos 59

10 Juego 3: Batalla Naval por la Radio 61
10. Introducción 61
10.2 Lo que necesitarás 62
10.3 Cómo funciona el juego 62
10.4 Una muestra del juego 63
10.5 Programación: Batalla Naval 64
10.6 Tarea 1: Configura el juego 64
10.7 Tarea 2: Dispara 65
10.8 Tarea 3: Recibe un disparo 65
10.9 Tarea 4: Recibe el resultado del disparo: "Tocado" o "Agua" 65
10.10 Actividad extensiva 66
10.11 Problemas 66
10.12 Recursos 67

Índice 70



PREFACIO

Acerca del libro

Este libro presenta una serie de actividades para enseñar los conceptos básicos de las redes informáticas. Si bien no aprenderás todos los aspectos de las redes informáticas, esperamos que sirva como un buen punto de partida.

Para redes micro:bits, usamos la radio de la micro:bit personalizada para comunicación por radio. Cuando uno escucha la palabra radio, lo que viene a la mente es la radio que lanza canciones desde su canal de transmisión de radio favorito. Pero, una radio, o un transceptor de radio (transmisor / receptor), se utiliza en las comunicaciones para generar y recibir ondas de radio que contienen información como audio, video o datos digitales. Y todas las micro:bits tienen radios incorporadas[1].

Cada capítulo presenta desafíos interesantes en comunicaciones de radio y redes con micro:bits. Después de algunos de los capítulos, ¡hay un juego emocionante que esperar! En las secciones de programación, usarás el Editor de bloques de JavaScript en https://makecode.microbit.org/ para desarrollar soluciones para superar esos desafíos[2].

Al escribir este libro, no asumimos ningún conocimiento de comunicaciones de radio o redes.

Sin embargo, suponemos que has escrito programas con una micro:bit. Por ejemplo, esperamos que estés familiarizado con las variables, las sentencias si-entonces-sino y los bucles. Las actividades en cada capítulo proporcionarán amplias oportunidades para poner este conocimiento en práctica.

[1] La CPU en el micro: bit es un Nordic Semiconductor nRF51822 y contiene un módulo de radio incorporado de 2.4GHz. Esta radio se puede configurar para ejecutar el protocolo Bluetooth Low Energy (BLE), pero en este libro, utilizaremos la comunicación más simple de micro:bit a micro:bit.

[2] Esta versión del libro usa JavaScript Blocks Editor; también estamos trabajando en una versión de MicroPython.

Acerca de los autores

Cigdem y Anthony son investigadores, lo que significa que trabajamos en nuevas ideas y productos. Trabajamos para una empresa llamada Nominet, que maneja la parte de Internet que controla cómo se usan los nombres (como www.bbc.co.uk) cuando personas, computadoras o dispositivos como tablets o teléfonos inteligentes se conectan a otras computadoras a través de Internet. Estamos muy contentos de tener la oportunidad de trabajar con micro:bits y la Fundación Micro:bit.

Entender cómo las computadoras hablan entre sí es algo que creemos que es importante, ¡por eso escribimos este libro! Hemos disfrutado de diseñar las tareas y los desafíos en el libro, y esperamos que tú también lo hagas.

Anthony & Cigdem

Agradecimientos:

- Gracias a la BBC y a los creadores de micro:bit, y a Zach Shelby y Jonny Austin de la Fundación Micro:bit por su apoyo.
- Gracias a David Whale (@whaleygeek) por toda su ayuda e inspiración.
- Gracias a James Burgin y Anna Adolphson en Nominet, por su ayuda con los videos y el diseño gráfico, y a Alistair Braden por su revisión y sugerencias.
- Un agradecimiento especial a Adam Leach, director de nuestro equipo de investigación en Nominet, por darnos la oportunidad de trabajar en esto.

Recursos & Actualizaciones

Las actualizaciones de este libro, formatos alternativos y enlaces a versiones en línea están disponibles en https://microbit.nominetresearch.uk/networking-book/

Recursos de enseñanza adicionales (planes de clase para cada capítulo) también se publicarán aquí.

Resumen

Comunicación por cables

Este capítulo es una introducción y una demostración divertida de trabajo en red. Las micro:bits pueden comunicarse cuando están conectados con cables. A través de cables, enviarás imágenes entre micro:bits.

Comunicación inalámbrica y de broadcast

Comenzarás a utilizar la comunicación por radio en este capítulo y aprenderás a comunicarte por radio. Con la comunicación broadcast, una micro:bit puede enviar mensajes a muchas otras micro:bits. Pero, ¡ten cuidado! Si todas las micro:bits hacen eso, es como si todos estuvieran hablando a la vez.

Comunicación grupal

Al formar grupos pequeños, enviarás y recibirás desde un número limitado de micro:bits. Esto es más manejable que la transmisión. Pero, seleccionar un identificador único para tu grupo será un desafío interesante.

Juego 1: Shakey Donkey

Este es un juego que usa la radio micro:bit. Mira si puedes descubrir cómo jugar y cómo funciona el juego.

Comunicación Unicast

La comunicación grupal y broadcast son divertidas. Pero a veces quieres hablar con una sola persona. Esto se llama comunicación *unicast*. Para hacer esto, necesitarás un identificador único para tu micro:bit.

Unicast Bidireccional

De nada sirve hablar con alguien si no recibes una respuesta. En este capítulo, programarás tu micro:bit para enviar un mensaje y obtener una respuesta. Además, calcularás cuánto tiempo demorará una respuesta. Al hacer esto, también programarás una de las herramientas más importantes utilizadas en Internet: Ping.

Juego 2: Piedra-Papel-Tijeras por Radio

Esto no es como el juego tradicional Piedra-Papel-Tijeras. ¡Funciona a través de la radio!

Manejo de Errores: Retransmisiones

Nada es perfecto, especialmente la comunicación por radio. ¿Qué pasa si tu mensaje se pierde en el camino? En este capítulo, probarás métodos para lidiar con la pérdida de mensajes. Por ejemplo, ¿te ayuda si envías tus mensajes más de una vez? Esto se llama retransmisión.

Manejo de Errores: Acuse de recibo

¡Es un desperdicio retransmitir si el otro lado ya recibió el mensaje! El receptor necesita una respuesta estándar (o un acuse de recibo) para evitar esto. En el lado del envío, si no recibes un acuse de recibo, puedes asumir que tu mensaje no fue recibido. En este capítulo, probarás qué tan bien funcionan los reconocimientos para mejorar la confiabilidad.

Juego 3: Batalla Naval por Radio

Has llegado lejos. ¡Ahora estás listo para otro juego clásico! Escribirás una versión del famoso juego Batalla Naval usando tus micro:bits. Tu experiencia en comunicación por radio y redes te ayudará en el camino.

¡Comencemos!

1. COMUNICACIÓN POR CABLES

1. Introducción

¡Todo está conectado hoy en día! Las computadoras y los dispositivos se conectan entre sí para formar redes. Y estas redes se conectan para formar Internet. Cuando decimos computadoras o dispositivos, estos pueden ir desde un portátil tradicional hasta un teléfono celular, una lavadora o un sensor de humedad. Por supuesto, también puede ser tu micro:bit. Cada vez más, Internet se está convirtiendo en un Internet de las cosas.

En este capítulo, formarás tu propia red usando cables para conectar dos micro:bits. Al hacer esto, aprenderás:

- **el concepto de un medio de comunicación y de señales**
- **el concepto de binario y bit**
- **el concepto de una red**

1.2 Lo que necesitarás

- **2 micro:bits**
- **4 puntas de clip de cocodrilo**
- **1 soporte de batería y 2 pilas AAA**
- **1 compañero de equipo**

1.3 Contexto

Para que dos micro:bits puedan enviarse mensajes entre sí, de alguna manera deben estar conectados, ya sea por cable o de forma inalámbrica, lo llamamos un *medio de comunicación*.

> **Definición 1. — Medio de comunicación.**
> Un medio de comunicación es la ruta física sobre la que se transmite una señal.

Un mensaje podría ser una cadena como "Hola", un número como "9" o una imagen de icono. Las micro:bits convierten cada mensaje en una señal para enviarlo a través del medio de comunicación.

Definición 2. — Señal.
Las señales son los voltajes u ondas electromagnéticas transmitidas en un medio físico cableado o inalámbrico.

Por ejemplo, toma el caso de cuando decimos "Hola" en un teléfono fijo. El auricular del teléfono convierte los sonidos en una señal de voltaje eléctrico. Entonces, esta señal se transmite al teléfono receptor por medio de cables; y en el receptor, se convierte nuevamente en sonido.

Ejercicio 1.
¿Cuál es el medio físico inalámbrico que hace posible la comunicación por radio?

Ni las computadoras, ni las micro:bit pueden procesar señales sin convertirlas en datos binarios: 0s y 1s. Además, los datos binarios procesados por las computadoras deben convertirse en señales antes de que puedan viajar en un medio de comunicación.

Definición 3. — Bit.
Un bit es la unidad de datos más pequeña en una computadora. Es como un átomo. Un se representa con 1 o 0.

Un grupo de 8 bits es un *byte*. La Tabla 1 muestra otras equivalencias. Al conectar computadoras o cualquier dispositivo a través de diferentes medios de comunicación, creamos *redes*.

Nombre	Tamaño
Byte (B)	8 bits
Kilobyte (KB)	1024 bytes
Megabyte (MB)	1024 kilobytes
Gigabyte (GB)	1024 megabytes
Terabyte (TB)	1024 gigabytes

Tabla 1: Grupos de bits.

Definición 4. — Red.
Una red informática es una colección de computadoras o dispositivos, que están conectados para comunicarse entre sí. En una red informática, hay al menos dos computadoras. Dos o más redes pueden conectarse para formar una red más grande: una red de redes. ¡Internet es una red masiva de redes!

En este capítulo, crearás una red de dos micro:bits conectadas a través de cables.

1.4 Programación: Transferencia Simple de Corazón

En esta sección, conectarás dos micro:bits a través de cables. Enviarás un icono de Corazón de una micro:bit a otra. La Figura 1.1 muestra cómo debería verse un icono de corazón en la pantalla de la micro:bit.[1] Esta actividad se realiza mejor con un compañero de equipo. A continuación, pasarás por cuatro actividades para programar tus micro:bits.

[1]Esta imagen es de micro:bit Educational Foundation en www.microbit.org

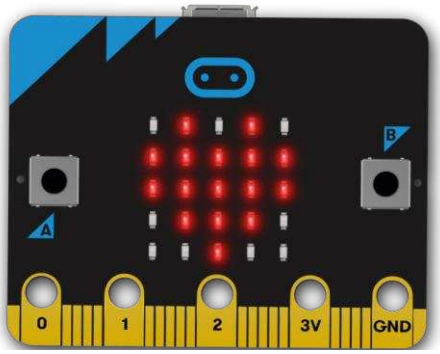


Figura 1.1: Micro:bit mostrando un ícono de corazón.[1]

1.5 Tarea 1: Mira la "Transferencia Simple de Corazón"

Descripción: Creamos un video para mostrar cómo deberían funcionar sus conexiones y programa en esta actividad.

Ve el video en https://microbit.nominetresearch.uk/networking-book/simple_heart_transfer.html

Instrucciones: Mira la transferencia simple de corazón en el video.

Importante: No omitas esta Tarea. Te ayudará a probar si funcionan los archivos que descargaste para la Tarea 2. También te ayudará a escribir tu programa para este capítulo.

1.6 Tarea 2: Conecta tu micro:bit y prueba el telégrafo

Descripción: Conectarás tus micro:bits utilizando cables y usarás un programa para verificar las conexiones. Puedes seguir las instrucciones a continuación, o hay instrucciones paso a paso más detalladas en la actividad de telégrafo micro:bit [2] en el sitio web de micro:bit.

Instrucción: Usando clips de cocodrilo, conecta el pin 3V entre las dos micro:bits, y conecta los pines GND. Luego conecta el pin 1 en una micro:bit al pin 2, y viceversa. Ten cuidado de conectar correctamente las conexiones de los clips de cocodrilo: dos de los cables se conectan directamente (3V → 3V y GND → GND) pero los otros dos se cruzan (1 → 2 y 2 → 1).

Mira la Figura 1.2 para ver un ejemplo, y observa cuidadosamente los colores (no necesitas usar los mismos colores, por supuesto, pero deben hacer las mismas conexiones).

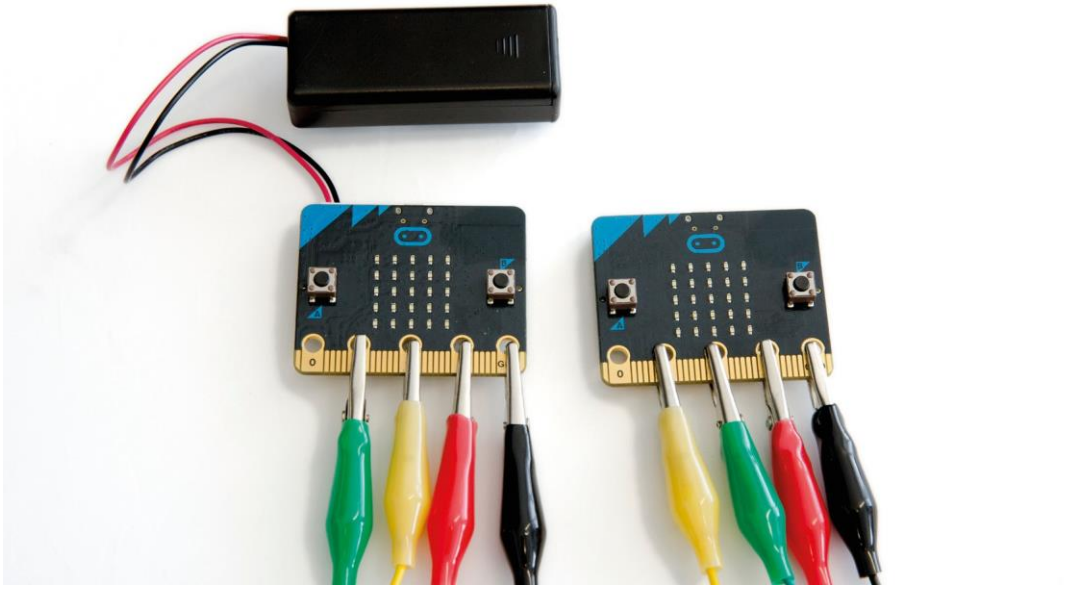


Figura 1.2: Cableado micro:bits. Dos de los cables se conectan directamente (3V → 3V y GND → GND) pero los otros dos se cruzan (1 → 2 y 2 → 1).

[2] https://www.microbit.co.uk/td/lessons/telegraph/activity

Para probar, usa el programa de la Figura 1.3; presiona el botón A en cada micro:bit y verifica que el LED se ilumina en el otro. Usarás los bloques del menú Pins. Este menú está en Avanzado. Haz clic en el enlace Más para ver todos los bloques.

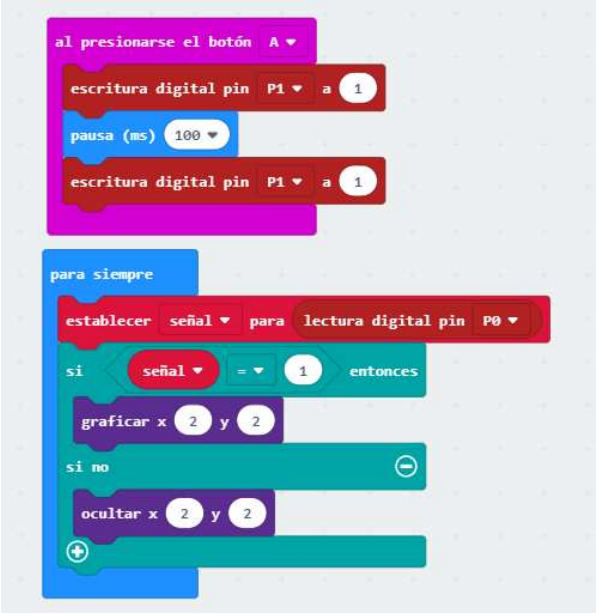


Figura 1.3: Programa de telégrafo. Presionando el botón A se envía una señal al otro lado usando el Pin 1. La micro:bit receptora escucha en el Pin 2 para verificar si se recibió una señal. Si hay una señal, enciende el (2,2) píxel en la pantalla.

1.7 Tarea 3: Prueba los archivos Hex "Transferencia Simple de Corazón"

Descripción: Proporcionamos dos archivos en https://microbit.nominetresearch.uk/networking-book/microbit1_ wired_simpleheart_secret.hex y en https://microbit.nominetresearch.uk/networking-book/microbit2_wired_ simpleheart_secret.hex para que pruebes cómo debería funcionar el programa final. Estos archivos se ejecutarán en tu micro:bits, pero no podrás visualizar el código utilizando el editor de bloques de JavaScript.

Instrucción: Descarga el código *Simple Heart Transfer* (Transferencia Simple de Corazón) en tus micro:bits. Hay dos archivos hexadecimales diferentes para la micro:bit 1 y la micro:bit 2. Prueba el programa inclinando tus micro:bits y verificando cuando se muestre el icono de corazón.

1.8 Tarea 4: Programa una transferencia de corazón

Descripción: En esta tarea, programarás tus micro:bits para obtener un comportamiento similar al observado en las Tareas 2 y 3. Para realizar esta tarea, deberás pensar en las siguientes preguntas:

1. ¿A qué entrada reaccionará la micro:bit en tu programa?
2. ¿Cómo se envían datos entre las micro:bits?
3. Sugerencia: ¿Crees que se está enviando un icono real de Corazón?

Instrucción: Para la pregunta 1, mira las opciones en el menú de entrada del editor Bloques de JavaScript. Para la pregunta 2, usa el ejemplo del programa Telegraph en la Figura 1.3. Para la pregunta 3, aquí hay otra gran sugerencia.

Sugerencia: Supongamos que la micro:bit 2 sabe que recibirás un icono de corazón de la micro:bit 1.

Programa tu micro:bit 1 para que:

1. Muestre un icono de corazón hasta que se incline sobre la micro:bit 2.
2. Cuando se incline sobre la micro:bit 2, envía un pulso a la micro:bit 2 por medio del pin correcto.
3. Cuando la micro:bit 1 reciba un pulso en su pin correcto, mostrará un ícono de corazón.

Programa tu micro:bit 2 para que:

1. Muestre un icono de corazón cuando recibe un pulso en su pin correcto.
2. Cuando se incline sobre la micro:bit 1, envíe un pulso a la micro:bit 2 por medio del pin correcto.

1.9 Actividad Extensiva

Ejercicio 2.
Mira el video en https://microbit.nominetresearch.uk/networking-book/pixel_heart_transfer.html.
Basándote en este video, discute con tu compañero de equipo cómo pueden enviar datos más complejos a través de cables. Haz una propuesta y discute con

Ejercicio 3.
Mira los dos videos en la sección de Recursos. ¿Cómo están relacionados con tu tarea? Discute.

1.10 Problemas

Problema 1.1 ¿Qué es un bit?

Problema 1.2 ¿Cuántos bits hay en un kilobyte?

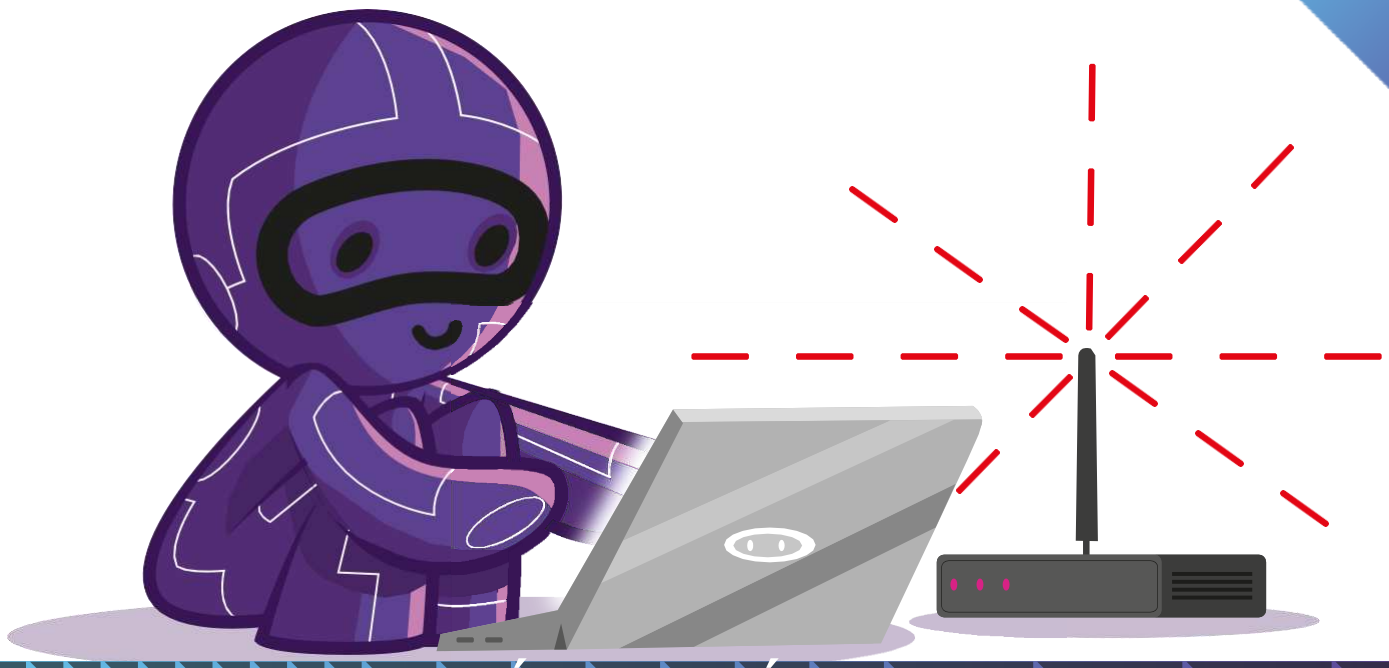
Problema 1.3 Explica el uso de Tierra (GND) y pines de 3 V en tu micro:bit.

Problema 1.4 ¿Cuántos bits enviaste al receptor en tu programa "Transferencia Simple de Corazón"?

Problema 1.5 ¿Cómo se envían los bits por el cable en tu programa?

1.11 Recursos

- **Video: Cual es el internet (Code.org) - https://youtu.be/Dxcc6ycZ73M**
- **Video: Internet: Cables y Wifi (Code.org) - https://youtu.be/ZhEf7e4kopM**
- **BBC Bitesize, Presentación de Binary - http://www.bbc.co.uk/education/guides/zwsbwmn/revision**



2. COMUNICACIÓN INALÁMBRICA Y DE BROADCAST

2. Introducción

La comunicación inalámbrica (radio), por ejemplo WiFi y teléfonos móviles, es una forma popular de conectarse a Internet. En el Capítulo 1, conectaste dos micro:bits a través de cables. En este capítulo, conectarás tus micro:bits usando radios.

Al hacer esto, no solo aprenderás cómo usar la radio de tu micro:bit sino también la comunicación de transmisión. En la comunicación inalámbrica, normalmente una micro:bit envía mensajes a todas las micro:bits. En resumen, este capítulo abarca:

- **la comunicación *inalámbrica* y cómo configurar la radio micro:bit**
- **el concepto de transmisión y *dirección de broadcast***
- **recibir y enviar diferentes tipos de mensajes (por ejemplo, un número o una cadena) usando broadcast**
- **cuándo la transmisión es útil, y cuándo no lo es**

2.2 Lo que necesitarás

- **2 micro:bits**
- **2 portabaterías y 4 pilas AAA**
- **1 compañero de equipo**

2.3 Contexto

La comunicación inalámbrica usa radiación electromagnética (ondas de radio y microondas) para enviar información. Las ondas de radio son esencialmente ondas electromagnéticas que irradian desde una antena (como las antenas de un enrutador WiFi). Entonces, la comunicación inalámbrica siempre se transmite. En otras palabras, las señales de los enrutadores WiFi pueden ser escuchadas por otros dispositivos WiFi sintonizados en la misma frecuencia de radio.

Lee más acerca de la frecuencia en la sección de Lectura Adicional al final.

Definición 1. — *Broadcast*.
En las redes, la comunicación de *broadcast* significa que el mensaje de un único emisor se transmite a todos los receptores en una red.

Pero, ¿significa esto que la transmisión solo es posible con comunicaciones inalámbricas? No, pero es más engorroso. Por ejemplo, en la comunicación por cable, la transmisión es posible repitiendo el mismo mensaje en todos los cables.

Finalmente, los receptores pueden negarse a recibir mensajes de broadcast si no están etiquetados con una dirección de broadcast.

Definición 2. — Dirección de broadcast.
Una dirección de broadcast es una dirección especial que dice que todos los dispositivos en la red deberían recibir este mensaje.

En una micro:bit, la dirección de broadcast se puede configurar estableciendo la ID de grupo de la radio de la micro:bit. Todas las micro:bits deben tener la misma ID de grupo para que funcione la transmisión. Experimentarás el broadcast con micro:bits en la Sección 2.3.

Lecturas adicionales

Veamos la comunicación inalámbrica con un poco más de detalle. Ya aprendiste que las ondas de radio son esencialmente ondas electromagnéticas. Los científicos han descubierto que las ondas electromagnéticas se pueden organizar juntas en una escala llamada espectro electromagnético. La figura 2.1 muestra el espectro electromagnético y las diferentes ondas electromagnéticas [a].

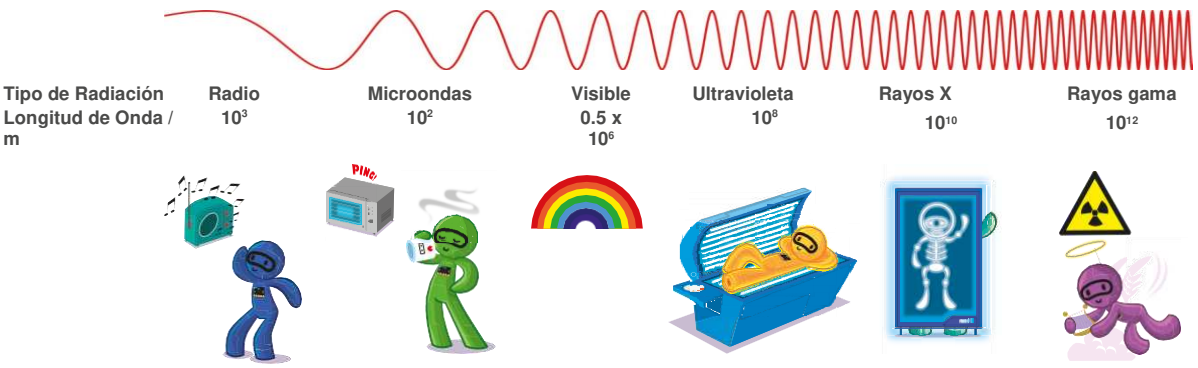


Figura 2.1: Espectro Electromagnético.

Una cosa para notar en la Figura 2.1 es que las ondas de radio están dentro de las frecuencias 30KHz y 300 GHz en el espectro electromagnético. Las ondas de radio incluyen microondas, que tienen frecuencias entre 300MHz y 300GHz. Las ondas de radio viajan rápido: se mueven a la velocidad de la luz, ¡que es de alrededor de 300,000 km por segundo! Vamos a definir la frecuencia de manera más formal. La frecuencia de una onda es la cantidad de ondas que pasan por un punto en un segundo. La unidad de frecuencia es hertz (Hz). Al igual que en los ejemplos anteriores, normalmente verás que las frecuencias se dan como megahertz (MHz) o gigahertz (GHz). 1 MHz es igual a 1 millón (106) Hz. 1 GHz es igual a 1 billón (109) Hz. La radio de tu micro:bit opera en el rango de frecuencia de 2402 MHz a 2480 MHz. ¿Qué otras tecnologías inalámbricas operan en el mismo rango que la radio de la micro:bit?

Sugerencia: la sección de recursos al final de este capítulo también te será útil. Además de la frecuencia, otro parámetro importante de las ondas electromagnéticas es la longitud de onda. La longitud de onda de una onda es la distancia entre un punto en la onda y el mismo punto en la siguiente onda. La unidad de longitud de onda es metros. La Figura 2.2 muestra un ejemplo de una longitud de onda.

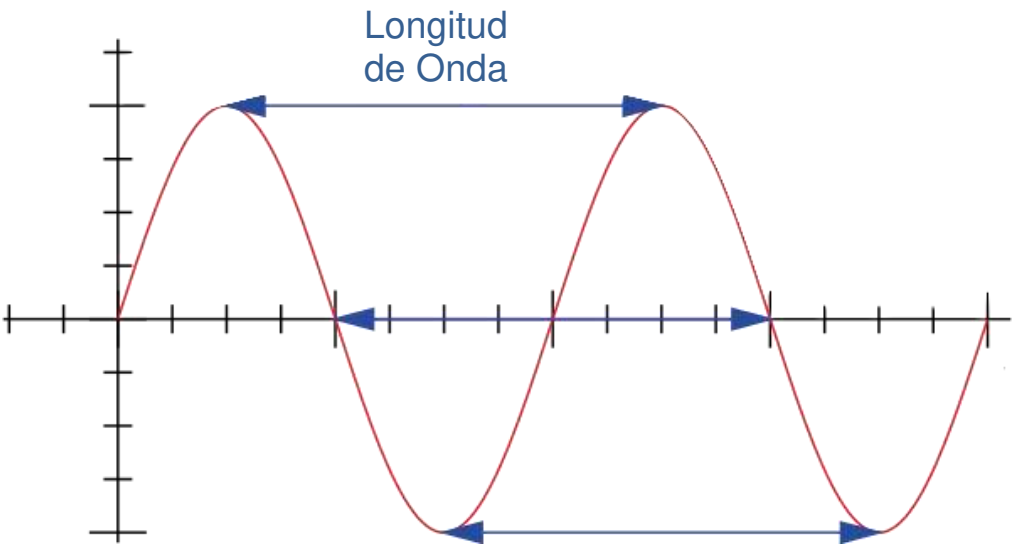


Figura 2.2: Longitud de onda es la distancia entre un punto en la ola y el mismo punto en la siguiente ola. Se mide como la distancia entre dos picos.

La frecuencia y la longitud de onda están relacionadas. La relación entre la frecuencia y la longitud de onda viene dada por una fórmula:

$$\text{longitud de onda (metros)} = \frac{\text{Velocidad de la luz (metros/segundos)}}{\text{Frecuencia (Hertz)}} \qquad (2.1)$$

De la Ecuación 2.1, vemos que cuanto mayor es la frecuencia, más corta es la longitud de onda. Puedes ver esto también en la Figura 2.1. ¿Qué tan largas crees que son las ondas de radio de tu micro:bit?

2.4 Programación: Recibe y envía mensajes de broadcast

En esta actividad, aprenderás cómo puedes recibir un mensaje de una transmisión micro:bit. Además, enviarás mensajes de broadcast tu mismo. Si estás ejecutando esta actividad con tu profesor en un aula, la micro:bit de tu profesor será la emisora de la transmisión e intentarás recibir mensajes de esta micro:bit. Si estás ejecutando esta actividad solo o con un amigo, puedes encontrar los códigos de ejemplo para la transmisión micro:bit bajo https://microbit.nominetresearch.uk/networking-book/. Puedes usar estos ejemplos para probar tu código de receptor descargándolo a una segunda micro:bit. Estos archivos se ejecutarán en tus micro:bits, pero no podrás visualizar el código utilizando el editor de bloques de JavaScript. Completarás las siguientes tres tareas para experimentar con la transmisión.

2.5 Tarea 1: Configura tu radio

Descripción: Para la comunicación broadcast, necesitas que todas tus micro:bits tengan la misma identificación de grupo de radio. Esta ID de grupo será la dirección de transmisión. Esto es como sintonizar el canal correcto para recibir una transmisión de TV.

Instrucción: Programa la ID de grupo de tu micro:bit receptora en 0. Esta es la ID de grupo usada en los programas emisores de broadcast de ejemplo [1]. Para esto, usa el bloque de código para configurar el grupo de radio en el editor de Bloques JavaScript. Está bajo el menú Radio, como se muestra en la Figura 2.3. Puedes obtener más información sobre los bloques de radio en https://makecode.microbit.org/reference/radio.

[1]Si estás utilizando tus propios programas para enviar una transmisión, puedes seleccionar la ID del grupo que desees.

[a]Imagen por Dicklyon (Richard F. Lyon) - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=7184592

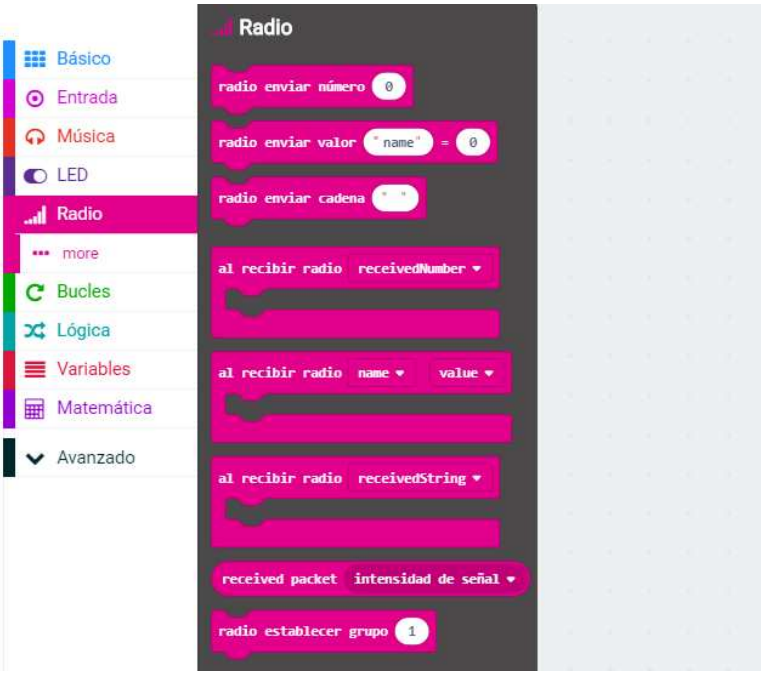


Figura 2.3: Configuración del grupo de radio en el editor de bloques de JavaScript.

2.6 Tarea 2: Recibe un mensaje de broadcast

Descripción: En esta tarea, programarás tus micro:bits para recibir un mensaje de una transmisión micro:bit. Utilizarás los programas emisores de broadcast de ejemplo para probar tu programa receptor.

Al escribir tus programas receptores, hay dos preguntas en las que debes pensar.

1. ¿Qué bloques en el editor de bloques de JavaScript necesitas usar para recibir un mensaje de radio?
2. Usando estos bloques, ¿puedes recibir cualquier tipo de mensaje, por ejemplo, un número o una cadena?

Instrucción: Primero, comenzarás programando tus micro:bits para recibir un número.

Descarga https://microbit.nominetresearch.uk/networking-book/SendNumber.hex en tu micro:bit receptora.

Este programa emisor usa el grupo de radio 0 para transmitir y envía un número entre 0 y 9, cada vez que presionas el botón A. Programa tu micro:bit para recibir y mostrar un número. Prueba tu programa utilizando la micro:bit emisora.

Segundo, programarás tu micro:bit para recibir una cadena. Descarga https://microbit.nominetresearch.uk/networking-book/SendString.hex en tu micro:bit emisora. Este programa también usa el grupo de radio 0 y envía una cadena cada vez que se presiona el botón A. Programa tu micro:bit para recibir y mostrar la cadena. Prueba tu programa utilizando la micro:bit emisora. ¿Qué recibiste?

2.7 Tarea 3: Envía un mensaje de broadcast

Descripción: Ahora es tu turno de enviar mensajes de broadcast. Si ejecutas este ejercicio en un grupo grande, con varias micro:bits, ¡deberías observar que estás recibiendo muchos mensajes! ¿Puedes adivinar quién envía qué mensaje?

Instrucción: Programa tu micro:bit para que pueda enviar un número cuando presionas el botón A y una cadena si presionas el botón B. Extiende tu programa receptor para que puedas recibir una cadena o un número. Para esto, usarás un buen truco para el bloque "on radio received". Al presionar el pequeño botón de configuración en el bloque aparece un menú. Este menú te permitirá arrastrar valores adicionales en Packet block. Notarás que el bloque original " on radio received" se extenderá para mostrar estos valores adicionales. La figura 2.4 muestra cómo funciona el truco.

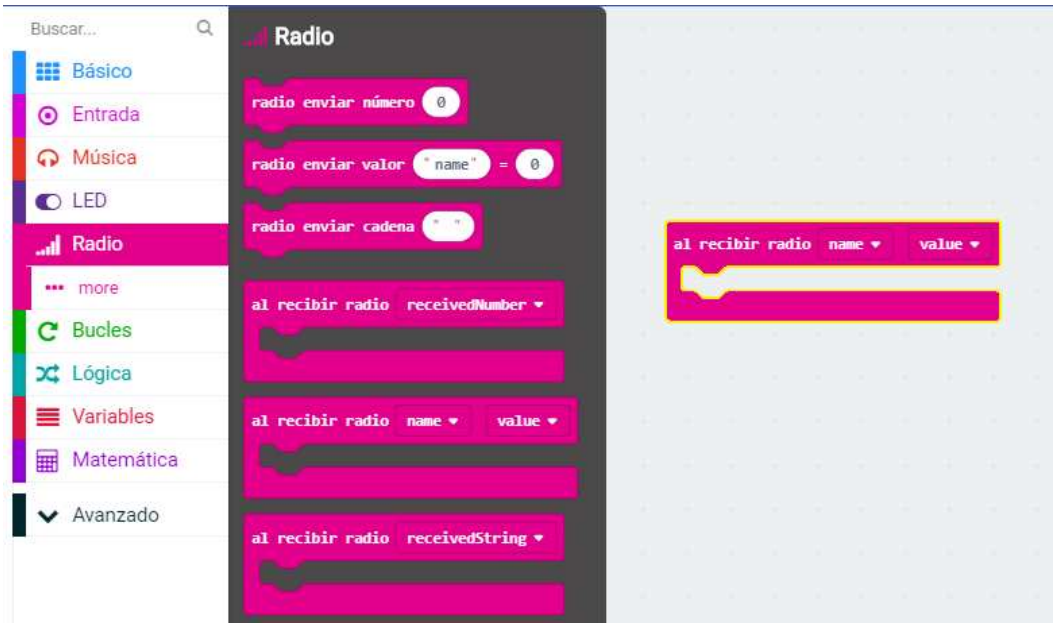


Figura 2.4: Hacer una radio con la micro:bit para recibir una cadena o un número.

2.8 Actividad extensiva

Ejercicio 1.
Extiende tu programa en la Tarea 2 para recibir una cadena. Muestra una cara "Triste" en la pantalla de tu micro:bit hasta que recibas un mensaje de "Hola". A continuación, muestra una cara "feliz" durante 2 segundos.

Ejercicio 2.
Discute algunos problemas con la comunicación broadcast. ¿Siempre es útil o necesario enviar mensajes a todos? ¿Qué hay de la privacidad? ¿Es un problema que todos reciban todos los mensajes?

2.9 Problemas

Problema 2.1 ¿Con qué rango de frecuencia funciona la radio de tu micro:bit?

Problema 2.2 ¿Cuál es la velocidad de la luz?

Problema 2.3 Usando la Ecuación 2.1, calcula la longitud de onda de la radio de tu micro:bit.

Problema 2.4 ¿Es más fácil transmitir usando comunicación por cable o inalámbrica? ¿Por qué?

2.10 Recursos

- **BBC Bitesize, El Espectro Electromagnético - http://www.bbc.co.uk/schools/gcsebitesize/science/edexcel/electromagnetic_spectrum/electromagneticspectrumrev1.shtml**
- **BBC Bitesize, Introducción a las Ondas - http://www.bbc.co.uk/schools/gcsebitesize/science/aqa_pre_2011/radiation/anintroductiontowavesrev2.shtml**
- **Video: Cómo funciona el Wi-fi? (Brit Lab) - https://youtu.be/xmabFJUKMdg**
- **Con cable, por qué todo lo inalámbrico es de 2,4 GHz?- https://www.wired.com/2010/09/wireless-explainer/**

3. COMUNICACIÓN GRUPAL

3. Introducción

En el capítulo anterior, experimentaste con la transmisión: enviando mensajes a todo el mundo. En este capítulo, aprenderás a enviar un mensaje para que sólo llegue a un grupo de personas más pequeño. Esta es una actividad que se lleva a cabo mejor con un grupo grande de amigos o compañeros de clase para que puedas experimentar con diferentes grupos y tamaños de grupo.

La comunicación grupal (también conocida como multicast) es un concepto interesante que permite varias de las tecnologías de Internet actuales. Por ejemplo, permite enviar videos lo más rápido posible a través de Internet. En este capítulo, aprenderás:

- **El concepto de *comunicación de grupo* y *dirección de grupo o multicast***
- **Cuándo la comunicación grupal es útil y cuándo no lo es**

3.2 Lo que necesitarás

- **2 micro:bits**
- **1 pizarra / pizarrón**
- **marcadores de pizarra / post-it**
- **1 compañero de equipo**

3.3 Contexto

En el capítulo anterior, todas las micro:bits recibieron mensajes de todas las demás micro:bits. Esto podría haber sido confuso (¡o divertido!). Ahora, intentemos limitar a quién puedes enviar mensajes y de quién puedes recibir mensajes. Esto se llama comunicación grupal. La comunicación grupal se usa en Internet para enviarla a muchas personas al mismo tiempo. Por ejemplo, la televisión por Internet y la videoconferencia usan comunicación grupal.

> **Definición 1. — Comunicación grupal.**
> En comunicación grupal o multicast, se envía un mensaje solo a las computadoras del grupo.

Para esto, los mensajes deben estar etiquetados con una dirección de grupo o multicast.

Definición 2. — Dirección de grupo.
Una dirección de grupo o multicast es una dirección especial que indica que todos los dispositivos del grupo deben recibir este mensaje.

Para configurar la dirección de grupo (o ID de grupo) en la radio de tu micro:bit, volverás a utilizar el bloque de radio "radio establecer grupo" bajo el menú Radio como en el Capítulo 2. El principal desafío de este capítulo es crear grupos para la comunicación. ¿Cómo aprenden las computadoras y se unen a estos grupos?

¿Qué sucede cuando dejan un grupo? En este capítulo, tendrás la oportunidad de pensar en estas preguntas cuando experimentes con la creación de grupos.

Lectura adicional
Al configurar identificaciones de grupo para las micro:bits, observarás que los ID de grupo varían de 0 a 255. Esta es la representación decimal (base 10) de los ID de grupo. Pero también podemos escribir estas identificaciones de grupo en binario (base 2). Para el caso binario, necesitaremos 8 bits para obtener un ID de grupo máximo de 255.
Pensemos en la representación binaria de los ID de grupo. La Figura 3.1 muestra un ejemplo para el grupo ID 172 en 8 bits: 10101100. Observa que, comenzamos a leer bits de derecha a izquierda. Cada bit está numerado del 1 al 8, lo que corresponde a una potencia de dos. El bit más a la derecha, el bit 1, significa $2^0 = 1$. El bit 2 significa $2^1 = 2$ y continuamos así hasta que lleguemos al bit 8, lo que significa $2^7 = 128$. Cada ubicación de bit puede contener 0 o 1. Para encontrar el valor decimal de 10101100, necesitamos hacer algunos cálculos: para la ubicación de bit x, multiplicamos su valor de bit por 2^{x-1}. Para la primera ubicación de bit 8, el valor de bit es 1, y debemos multiplicar 2^8-1 = 2^7 = 128) por 1. Después de hacer esto para todas las ubicaciones de bit, agregamos todos los valores que encontramos. El resultado de esta adición es 172. Ahora, verifica el caso 11111111. ¿Es realmente igual a 255? Para obtener más información, consulta la página de revisión binaria BBC Bitesize, en la sección de Recursos.

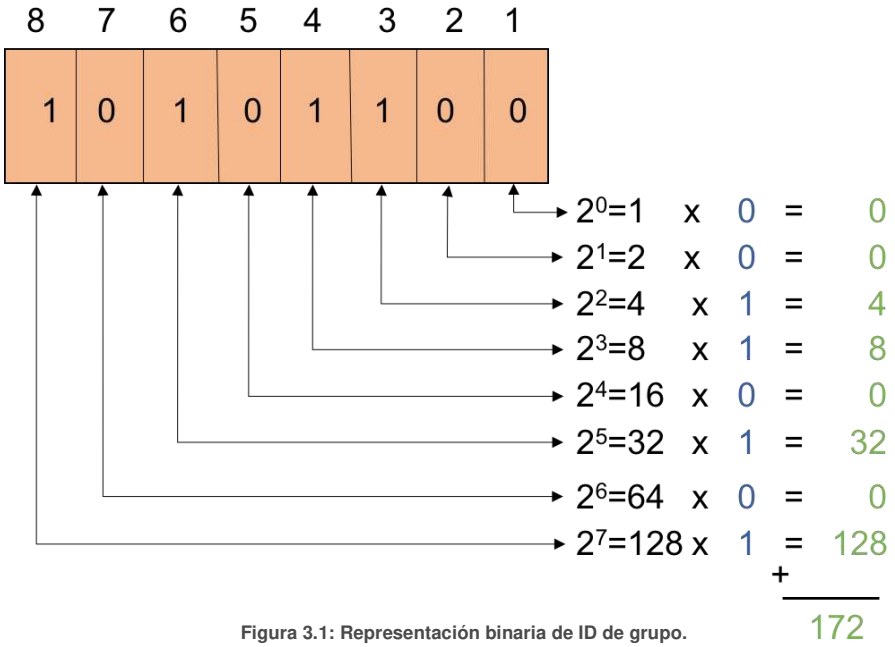


Figura 3.1: Representación binaria de ID de grupo.

3.4 Programación: Crea grupos y mensajes dentro de los grupos.

En este capítulo, debes trabajar en parejas o en grupos pequeños, con al menos 2 micro:bits en cada grupo. Completarán dos tareas para programar sus micro:bits para enviar mensajes y recibir mensajes de su grupo.

3.5 Tarea 1: Crea grupos

Descripción: En esta tarea, elegirás una ID de grupo única para tu grupo y configurarás tus radios con esta ID de grupo. Utilizarás el bloque de radio "radio establecer grupo" en tu programa en el editor de Bloques de JavaScript. Al elegir ID de grupo, debes pensar en la mejor forma de elegir este número.

Sugerencia: ¿Qué pasaría si dos grupos eligen el mismo número y cómo te asegurarías de que eso no ocurra?

Instrucción: Usa la pizarra y las notas post-it para elegir una ID de grupo. Asegúrate de que tu ID de grupo no sea la misma que la de cualquier otra identificación de grupo.

3.6 Tarea 2: Envía y recibe mensajes de grupo

Descripción: Utilizarás los programas del capítulo anterior para enviar y recibir mensajes a tu grupo. Cambiarás estos programas para contar la cantidad de mensajes que recibes. De esta manera, probarás si recibes mensajes que solo provienen de tu grupo.

Instrucción: Escribe un programa de remitente que envíe un número aleatorio entre 0 y 9, cuando presiones el botón A. Escribe un programa de receptor que incremente un contador cada vez que reciba un número. Cuando presionas el botón A en el receptor, muestra el valor del contador. Con tu grupo, prueba que estás recibiendo la cantidad correcta de mensajes. Prueba junto con otros grupos que no estás recibiendo sus mensajes.

3.7 Actividad extensiva

Ejercicio 1.
¿Qué tan fácil o difícil sería si las micro:bits pudieran crear grupos automáticamente? ¿Cómo elegirían una identificación de grupo? ¿Cómo se asegurarían de que nadie más tenga ese número? ¿La transmisión sería útil? Discute con tus compañeros de equipo.

Ejercicio 2.
¿Puede una micro:bit ser parte de dos grupos o más? ¿Cómo programarías tu micro:bit para hacer eso?

3.8 Problemas

Problema 3.1 Completa el espacio en blanco en esta oración: "Una comunicación de uno a muchos entre un emisor y un grupo de receptores es comunicación---".

a) unicast

b) multicast

c) broadcast

d) ninguna de las anteriores

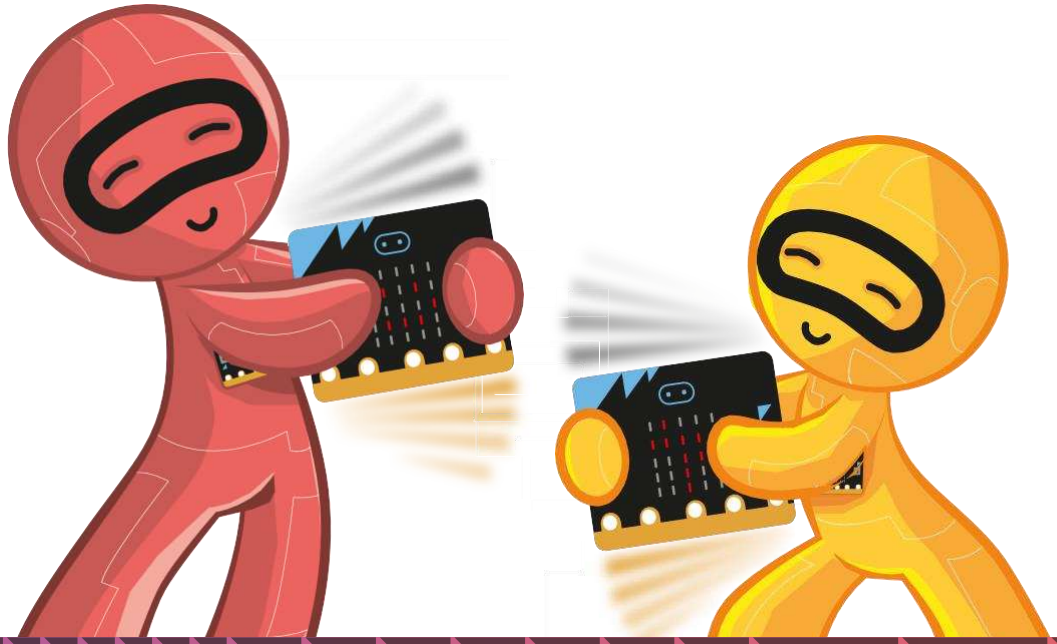
Problema 3.2 Supongamos que la ID del grupo es de 3 bits. Por ejemplo, 010 es una identificación de grupo. ¿Cuál es la cantidad máxima de grupos que puede tener en una red?

Problema 3.3 Si la ID del grupo era de 6 bits, ¿cuál es el número de grupo más grande que podrías elegir para tu micro:bit?

Problema 3.4 "En comparación con el Broadcast, los receptores en la comunicación grupal reciben más mensajes. "¿Verdadero o falso?

3.9 Recursos

BBC Bitesize, Revisión Binary - http://www.bbc.co.uk/education/guides/z26rcdm/revision



4. JUEGO 1: SHAKEY DONKEY

4. Introducción

Pongamos en práctica todo lo que has aprendido hasta ahora con un juego divertido. Si aún no lo has visto, Shakey Donkey es un juego de micro:bits que usa la radio [1]. Shakey Donkey se juega con dos jugadores, y mide qué tan rápido reaccionas ante un Donkey que aparece en tu micro:bit. El juego comienza con agitando las micro:bits. En el momento en que tu micro:bit muestra un Donkey, debes gritar "¡Donkey!" y agitar tu micro:bit para hacerlo desaparecer. Al final, cuando presiones el botón A, si tu micro:bit muestra una cara feliz, ¡has ganado!

En este capítulo, practicarás:

1. **el concepto de *comunicación grupal***
2. **usar una dirección de grupo o multicast**
3. **enviar y recibir mensajes**
4. **agitar y los botones de entrada**
5. **variables del programa y números aleatorios**

4.2 Lo que necesitarás

- **2 micro:bits**
- **1 pizarra / pizarrón**
- **marcadores de pizarra / post-it**
- **1 compañero de equipo**

4.3 Programación: Juega a Shakey Donkey

Descripción: Para poder jugar este juego en grupos de 2, establecerás una ID de grupo única para tu pareja. Luego, programarás el juego Shakey Donkey que se te entregó en tres partes en la Figura 4.1.

[1]Este juego es de David Whale. Le agradecemos por permitirnos usarlo en este libro.

Instrucción: Para configurar tus grupos, repite la tarea del Capítulo 3. ¡Asegúrate de que las identificaciones de tu grupo sean únicas!

El juego se juega agitando tu micro:bit cada vez que aparece el burro en tu pantalla, para deshacerte de él.

Entonces, lo primero que debes hacer es programar lo que tu micro:bit debería hacer si es agitada. Esto se muestra en la Figura 4.1a. Ten en cuenta que, en esta primera parte, tu programa envía un número.

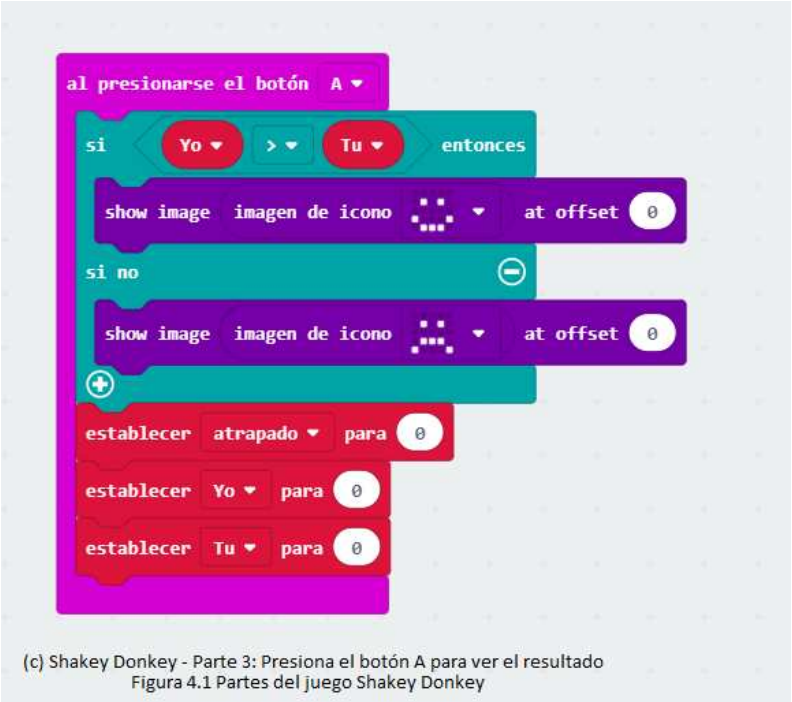
Entonces, necesitas un código para manejar un número recibido. Esta segunda parte se muestra en la Figura 4.1b. Añádelo a tu programa de editor de Bloques de JavaScript.

La tercera parte, que se muestra en la Figura 4.1c, maneja el caso cuando se presiona el botón A. Esta parte del programa decide si ganaste o no. Agrega esta parte a tu programa también.

Descarga el programa en tu micro:bit. Juega el juego Shakey Donkey con tu compañero de equipo. Luego, examina los problemas para explicar cómo funciona tu programa.

(a) Programa Shakey Donkey Parte 1: agita tu micro:bit para enviar tu tiempo de reacción

(b) Shakey Donkey - Parte 2: Recibe el tiempo de reacción del otro jugador y muestra el burro.



(c) Shakey Donkey - Parte 3: Presiona el botón A para ver el resultado
Figura 4.1 Partes del juego Shakey Donkey

4.4 Problemas

Primero veamos la Parte 1, en la Figura 4.1a.

Problema 4.1 Al principio, ¿cuál es el valor de la variable capturada para ambos jugadores? ¿Alguien necesita cambiar la variable *yo*?

Problema 4.2 ¿Quién puede enviarme su variable *yo* primero?

A continuación, veamos la Parte 2, en la Figura 4.1b.

Problema 4.3 Cuando recibes un número, establece la variable *caught*? ¿Qué significa la variable *caught*?

Problema 4.4 También cambia la variable *tu* por el Número recibido. ¿Qué representa el valor almacenado en la variable *tu*?

Ahora, veamos ambas Partes 1 y 2.

Problema 4.5 Imagina que ya comenzaste a jugar el programa. Viste algunos burros aparecer en tu pantalla, y los sacudiste. ¿Cómo cambió tu variable *yo*? ¿A qué es igual?

Finalmente, veamos la Parte 3, en la Figura 4.1c.

Problema 4.6 ¿Cómo sabes que has ganado? ¿El otro jugador sabe el resultado? ¿Cómo? Explica cómo se usan las variables *yo* y *tu* para decidir el ganador.

Problema 4.7 ¿Cómo te asegurarías de ganar este juego?

5. COMUNICACIÓN UNICAST: UNO A UNO

5. Introducción

Unicast, el envío de mensajes a un solo receptor, es la forma típica en que nos comunicamos en Internet. Por ejemplo, para ver una página web, enviamos mensajes de unicast a un servidor, que a su vez nos envía la página para mostrar en nuestro navegador. En este capítulo, enviarás mensajes de unicast, por ejemplo, a una micro:bit de un amigo o compañero de equipo. Al hacer esto, aprenderás algunas ideas básicas de redes de computadoras, que incluyen:

- **el concepto de *unicast***
- **el concepto de un *protocolo***
- **el concepto de *dirección y dirección IP***
- **el concepto de un paquete de datos y un encabezado**

5.2 Lo que necesitarás

- **2 micro:bits**
- **1 pizarra / pizarrón**
- **marcadores de pizarra / post-it**
- **1 compañero de equipo**

5.3 Contexto

Este capítulo cubre la comunicación *unicast*. Entonces, ¿qué es *unicast*?

> **Definición 1 — *Unicast*.**
> Transmisión de un mensaje a un solo receptor.

Al transmitir mensajes entre sí, las computadoras usan *protocolos.*

Definición 2 — Protocolo.
Un conjunto de reglas sobre cómo se envían los mensajes a través de las redes.

Simplemente, los protocolos definen cómo las computadoras deben enviar mensajes y qué deben hacer cuando reciben un mensaje. En Internet, cada computadora o dispositivo sigue el Protocolo de Internet (IP). Según el protocolo de Internet, a cada dispositivo se le asigna una *dirección* única, llamada *dirección IP*. Recuerde que ya ha utilizado direcciones especiales para difusión (*broadcast*) y multicast. En este capítulo, consideramos las direcciones de unicast. La dirección IP se usa para *unicast* en Internet[1].

Definición 3 — dirección IP.
Una cadena única que identifica las computadoras que usan el Protocolo de Internet para comunicarse a través de una red. Esta cadena se compone de 4 números decimales, que oscilan entre 0 y 255. Cada decimal está separado por puntos. Por ejemplo, 213.248.234.11 es una dirección IP.

Tu micro:bit también tiene una dirección (pero es un poco diferente). Ya has cambiado parcialmente la dirección de tu micro:bit, al cambiar la ID del grupo.

Cuando dos computadoras se comunican, el emisor envía un paquete de datos al receptor.

Definición 4 — *Paquete de datos*
Un paquete de datos *(data packet)* es una información enviada a través de una red. Este dato tiene una parte del mensaje real (por ejemplo, una imagen o un texto) y una o más partes del encabezado. Un encabezado contiene información útil para protocolos como el remitente y las direcciones IP del receptor.

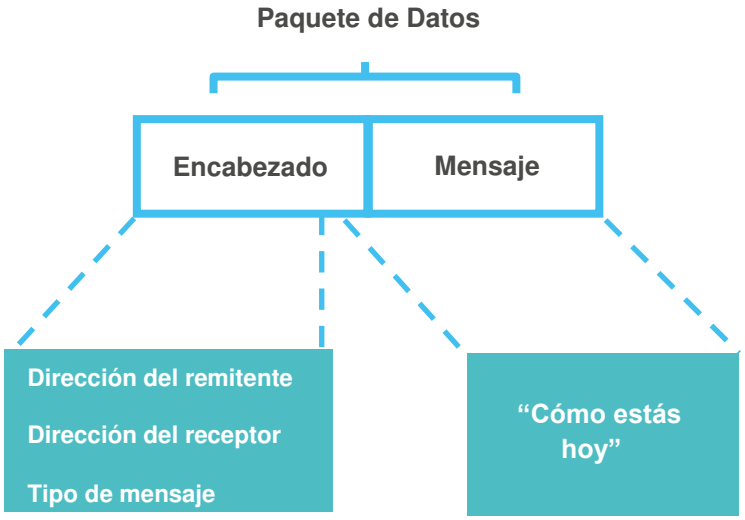


Figura 5.1: Un paquete de datos contiene un mensaje y un encabezado. Un encabezado contiene información para ayudar a un protocolo como las direcciones del remitente y del receptor, y los tipos de mensajes. Diferentes protocolos pueden agregar diferentes encabezados a un mensaje

La Figura 5.1 muestra cómo los datos y un encabezado forman un paquete de datos. En esta figura, además de las direcciones del remitente y del receptor, el encabezado del ejemplo también incluye un tipo de mensaje. El tipo de mensaje le dice al receptor si está recibiendo, por ejemplo, un texto o una imagen. Recuerda, en los capítulos anteriores, programaste tus receptores para recibir un tipo específico de mensaje. Si tus paquetes contienen un encabezado con el tipo de mensaje, entonces sería más fácil escribir el programa receptor. En este capítulo, para unificar a otras micro:bits, crearás un paquete de datos agregando un encabezado con las direcciones de origen y destino.

[1]También hay un rango especial de direcciones IP que se pueden usar para multicast y broadcast

5.4 Programación: Envía y recibe mensajes *Unicast*

En esta sección, programarás tus micro:bits para enviar y recibir mensajes de unicast completando cuatro tareas. Para empezar, necesitas dos micro:bits. Para que la unicast funcione, tu radio debe recibir todos los mensajes enviados, pero tu programa debe leer solo los que están dirigidos a usted. Esto es como ver todas las publicaciones entrando a tu casa, pero solo abriendo los sobres con tu nombre.

5.5 Tarea 1: Configura tu radio

Descripción: Para recibir cualquier paquete, enviado por cualquier persona, debes usar la transmisión como comunicación subyacente.

Instrucción: Establece tu ID de grupo de radio como lo hiciste en el Capítulo 2 para la comunicación broadcast.

5.6 Tarea 2: Diseña tu encabezado

Descripción: El emisor micro: bit necesita agregar un encabezado a cada mensaje antes de enviarlo. El encabezado del mensaje incluirá:

- **dirección del remitente**
- **dirección del receptor**

Para el encabezado del mensaje, crearás una cadena especial.

Instrucción: Primero construye las direcciones del remitente y del receptor. Con tu compañero de equipo, elije cadenas de dos letras como direcciones micro:bit. Necesitas una dirección para tu micro:bit y una dirección para la micro:bit de tu compañero de equipo. Por ejemplo, puedes usar tus iniciales: estas son "CS" y "AK" para los autores de este libro. **¡Importante! Tus direcciones deben ser únicas en todas las direcciones de micro:bits que están en la misma habitación que tú.** A continuación, une las cadenas para las direcciones del remitente y del receptor para crear un encabezado. Usa los bloques pertenecientes al menú Texto en el editor de Bloques JavaScript (mira la Figura 5.2), por ejemplo, *unir (join)*.

Figura 5.2: menú de texto en el editor de bloques de JavaScript

5.7 Tarea 3: Crea tu paquete y envíalo

Descripción: Ahora es el momento de crear tu paquete. Como se muestra en la Figura 5.1, un encabezado y un mensaje forman un paquete. Tu paquete final tendrá la siguiente información:

- **dirección del remitente**
- **dirección del receptor**
- **tu mensaje**

Instrucción: Elige una cadena como tu mensaje. Por ejemplo: "Hola". Usa los bloques de texto para unir tu cadena de mensaje con tu encabezado. Ahora, tu micro:bit emisora está lista para enviar paquetes de unicast.

5.8 Tarea 4: Recibe un paquete

Descripción: Cuando la micro:bit receptora recibe un paquete, decide si recibir o ignorar el paquete. Observa que la micro:bit receptora recibe una sola cadena, pero sabe que esta cadena se compone de:

- **Dirección del remitente: primeras 2 letras**
- **Dirección del destinatario: las próximas 2 letras**
- **Mensaje del remitente: el resto de la cadena**

El receptor necesita usar esta información para decidir qué paquetes son para sí mismo.

Instrucción: Divide la cadena recibida en la *dirección del remitente*, la *dirección del destinatario* y las variables de *mensaje del remitente*. Usa los bloques de texto, por ejemplo, subcadena y comparación. Verifica si la dirección del receptor es igual a la dirección de tu micro:bit. Si lo es, entonces tu micro:bit es la receptora correcta. Muestra la dirección del remitente y el mensaje en tu pantalla. Si tu micro:bit no es la receptora, sé un buen ciudadano e ignora el mensaje.

5.9 Desafío: Filtro de remitentes

Descripción: A veces, es posible que no desees recibir mensajes de parte de nadie. Para esto, escribirás un programa para que solo recibas mensajes de dos personas que conoces. Llamaremos a esto tu *lista de permitidos* (a menudo referida como una *lista blanca*).

Instrucción: Extiende el programa receptor para verificar también el campo de *dirección del remitente* en el encabezado. Verifica si esta dirección está en su lista de permitidos. En caso afirmativo, muestra la *dirección del remitente* y el mensaje. En caso negativo, ignora el mensaje. Prueba tu programa con direcciones dentro y fuera de su lista de permitidos.

5.10 Actividad extensiva

Ejercicio 1.
Puedes haber escrito dos programas separados: uno para el receptor y otro para el remitente. Cambia tu programa para que ambas micro:bits puedan enviar y recibir.

Ejercicio 2.
¿Has intentado escuchar mensajes enviados desde otras micro:bits en tu clase? ¿Cómo podría tu programa lograr esto? ¿Es esto lo correcto? ¿Cómo podrías proteger tus mensajes de los demás?

Ejercicio 3.
En este capítulo, hemos cubierto una forma de hacer unicast: poner las direcciones del remitente y del receptor en un encabezado de paquete de datos. Pero hay otra manera. Recuerda el Capítulo 3. Si configuras tu grupo solo para tu par de micro:bits, entonces es como si estuvieras haciendo unicast. Para unificarse así, elije una ID de grupo única, como lo hiciste en el Capítulo 3. Anúnciala en la pizarra para que nadie más la use. Escribe programas para tu par de micro:bits que envían y reciben usando esta ID de grupo de bot. ¿Cuáles son las limitaciones de hacer un unicast como este? Sugerencia: piensa en cuántos ID de grupo posibles hay. ¿Esto sería suficiente para todos en el mundo los que tengan una micro:bit?

5.11 Problemas

Problema 5.1 ¿De qué manera es la unicast similar al broadcast y a la comunicación grupal? ¿De qué manera es diferente?

Problema 5.2 ¿Cuáles no son direcciones IP?

a) -1.0.0.1

b) 278.0.10.0

c) 104.20.14.61

d) 127.0.0.1

e) 161.23.84;18

f) 161.73.246.13

g) 104.20.14.61.15

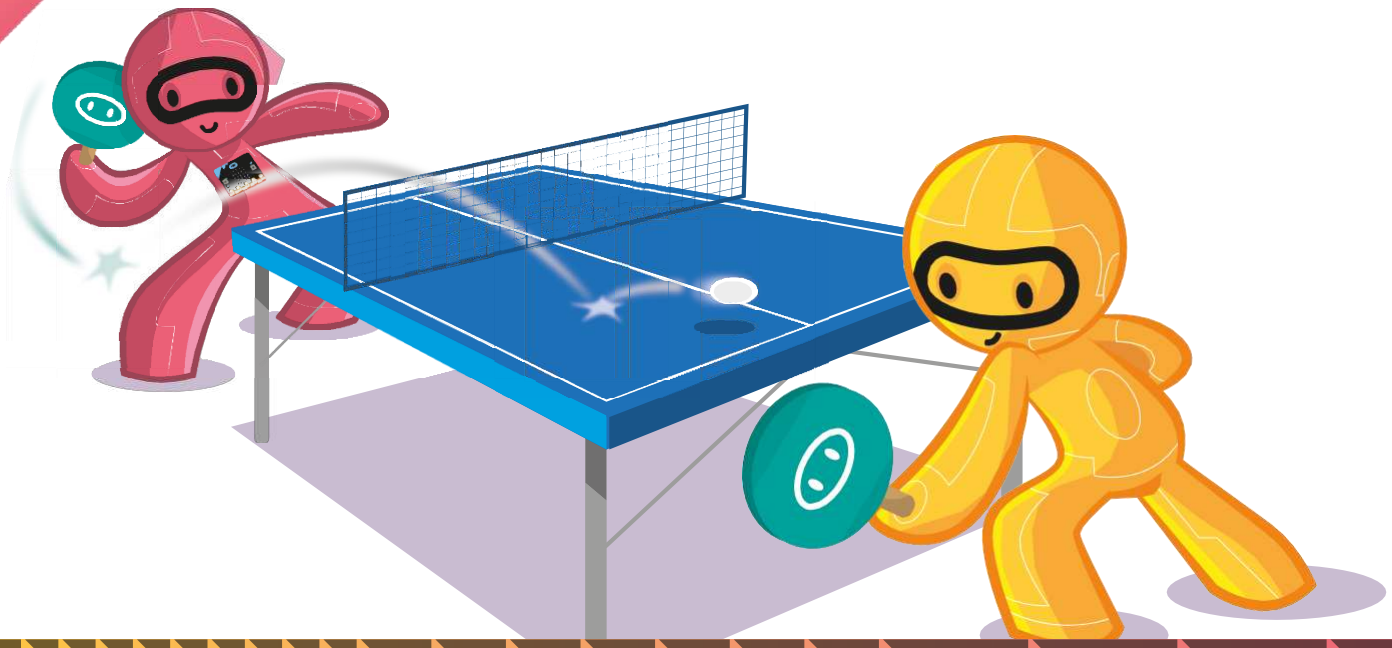
Problema 5.3 En este capítulo, usaste cadenas de dos letras para tus direcciones. ¿Cuántas personas diferentes pueden hacer unicast usando el tamaño de esta dirección?

Problema 5.4 Al seleccionar un tamaño de dirección para el encabezado de tu mensaje, ¿puedes elegir el tamaño que deseas? En tu programa, ¿qué sucede si aumentas el tamaño de tu dirección a 10 letras? ¿Ves algún beneficio? ¿O hay alguna limitación?

Problema 5.5 ¿Cómo afecta el tamaño del encabezado de un paquete de datos al tamaño real del paquete? Si el tamaño de tu paquete de datos era de 100 Bytes, y el tamaño de tu encabezado era de 10 Bytes, ¿qué tan grande podrían ser tus mensajes? ¿Qué sucede si el tamaño del encabezado aumenta a 50 Bytes?

5.12 Recursos

- **Video: direcciones IP y DNS (code.org) - https://youtu.be/5o8CwafCxnU**
- **Video: direcciones IP (CommonCraft) - https://www.commoncraft.com/video/ip-addresses**
- **BBC Redes bitesize - http://www.bbc.co.uk/education/topics/zjxtyrd**



6. UNICAST BIDIRECCIONAL

6. Introducción

En este capítulo, aprenderás sobre la comunicación bidireccional: enviar un mensaje a otra micro:bit y obtener una respuesta a su mensaje. También aprenderás sobre el programa Ping, que es una herramienta comúnmente utilizada para verificar si las computadoras todavía están conectadas a Internet. Este capítulo se basará en los aprendizajes del Capítulo 5. Las nuevas ideas son:

- **La idea de comunicación bidireccional**
- **El programa Ping**
- **El concepto de *round-trip-time (tiempo de ida y vuelta)***

6.2 Lo que necesitarás

- **2 micro:bits**
- **1 pizarra / pizarrón**
- **marcadores de pizarra / post-it**
- **1 compañero de equipo**

6.3 Contexto

La *comunicación bidireccional* permite la comunicación en dos direcciones entre dos computadoras.

> **Definición 1 —Comunicación Bidireccional.**
> Este es un modo de comunicación en el que los datos se transmiten en ambas direcciones pero no necesariamente al mismo tiempo.

En el capítulo anterior, tus micro:bits tenían roles claros: había un emisor y un receptor. En comunicación bidireccional, cualquiera de las micro:bits puede enviar y recibir mensajes. De esta forma, es posible crear protocolos bidireccionales. En estos protocolos, cuando una computadora envía un mensaje, espera cierta respuesta al mensaje.

Definición 2 — Ping.
Ping es un ejemplo de un protocolo de dos direcciones. Es ampliamente utilizado en Internet para probar si una computadora en red está encendida y conectada correctamente.

Un programa ping envía un *mensaje Ping* para comprobar si las computadoras están bien. Espera que este mensaje sea respondido, por ejemplo, con un mensaje *Pong*. Esto es como jugar al ping pong pero con computadoras y redes. Si el emisor no recibe una respuesta a su *Ping*, esto muestra que hay un problema con el receptor. También es un problema si toma mucho tiempo antes de que el emisor reciba una respuesta de *Pong*. Entonces, un programa ping mide el tiempo de ida y vuelta entre las dos computadoras para señalar estos problemas.

Definición 3 — Round-trip-time (RTT) (*tiempo de ida y vuelta*).
Ping es un ejemplo de un protocolo de dos direcciones. Es ampliamente utilizado en Internet para probar si una computadora en red está encendida y conectada correctamente.

En otras palabras, el emisor mide la diferencia en el tiempo cuando envió el *Ping* y cuando recibió el *Pong*.

RTT = Tiempo de recepción - tiempo de envío (6.1)

La Figura 6.1 muestra la relación entre Ping, Pong y el round-trip-time (tiempo de ida y vuelta).

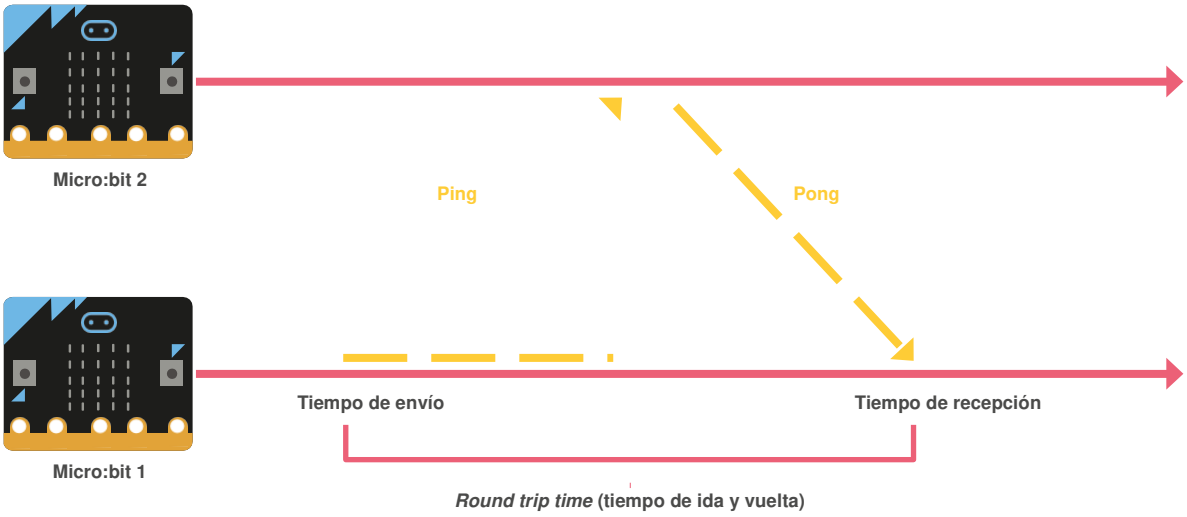


Figura 6.1: *Round-trip-time* (Tiempo de ida y vuelta). La Micro:bit 1 envía un mensaje Ping a la Micro: bit 2 en Tiempo de envío. La Micro:bit 2 responde con un mensaje Pong. La Micro:bit 1 recibe el mensaje Pong en Tiempo de recepción. La diferencia entre estos dos tiempos, Tiempo de recepción y Tiempo de envío, es el tiempo de ida y vuelta o *round-trip-time*.

Además del tiempo de ida y vuelta (RTT), el programa Ping reporta información estadística. La Figura 6.2 muestra un resultado de ejemplo como resultado del uso del comando:

ping www.google.com

en la website http://ping.eu/ping . En el ejemplo de la Figura 6.2, se enviaron cuatro mensajes Ping a www.google.com. El round-trip-time (tiempo de ida y vuelta) para cada mensaje se da con el valor de tiempo en cada línea. Por ejemplo, para el primer ping, el RTT es 10.2 ms (milisegundos). El programa también informa las estadísticas del ping. Por ejemplo, se enviaron 4 paquetes, se recibieron 4 paquetes. Esto significa 0% de pérdida de paquete. El promedio de RTT (mostrado como promedio) es de 10.184 ms.

Online service Ping

Ping – Shows how long it takes for packets to reach host

IP address or host name: www.google.com Go

--- PING www.google.com (172.217.23.228) 56(84) bytes of data. ---
64 bytes from 172.217.23.228: icmp_seq=1 ttl=54 time=**10.2** ms
64 bytes from 172.217.23.228: icmp_seq=2 ttl=54 time=**10.1** ms
64 bytes from 172.217.23.228: icmp_seq=3 ttl=54 time=**10.1** ms
64 bytes from 172.217.23.228: icmp_seq=4 ttl=54 time=**10.1** ms

--- www.google.com ping statistics ---

packets transmitted **4**
received **4**
packet loss **0** %
time **3002** ms

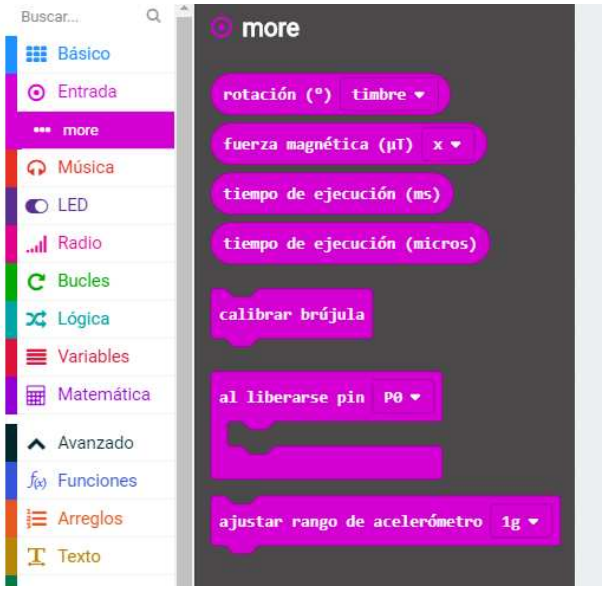
--- Round Trip Time (rtt) ---

min **10.149** ms
avg **10.184** ms
max **10.206** ms
mdev **0.074** ms

Figura 6.2: El resultado de ejecutar ping para enviar cuatro mensajes a www.google.com. El programa en línea ping http://ping.eu/ informa el tiempo de ida y vuelta y da un resumen estadístico de los resultados.

Con una micro:bit, para calcular el *round-trip-time* (tiempo de ida y vuelta) de tus mensajes, usarás la variable de *tiempo de ejecución*.

Definición 3 — tiempo de ejecución de la micro:bit.
Una variable que mantiene un registro de cuánto tiempo ha pasado desde que se activó o reinició la micro:bit (medido en milisegundos).



En el resto de este capítulo, usarás la variable de tiempo de ejecución para calcular el tiempo de ida y vuelta. Será muy útil registrar la hora en que enviaste el mensaje por primera vez y también cuándo recibiste una respuesta.

Sugerencia: Registrar el tiempo de ejecución o *running time* significa establecer una variable igual al tiempo de ejecución actual. Necesitarás combinar el bloque de "establecer" del menú *Variables* del editor Bloques JavaScript con el bloque de tiempo de ejecución del menú Entrada.

6.4 Programación: Ping

Esta actividad se realiza mejor con un equipo de dos. Juntos programarán sus micro:bits para ejecutar el programa Ping. Para esto, deberán completar cuatro tareas.

6.5 Tarea 1: Prepararse para el *unicast*

Descripción: Ping utiliza unicast entre la micro:bit emisora y receptora. Mira tus notas del Capítulo 5 y tu programa de unicast para recordar cómo hacer la unicast.

Instrucción: Comienza usando tu programa de unicast del Capítulo 5 como base. En este programa, decide qué micro:bit va a enviar los *Pings*, y qué micro:bit va a responder con *Pongs*. Establece las variables de dirección según tu decisión. Diseña tu encabezado del mensaje, el paquete *Ping* y el paquete *Pong*.

6.6 Tarea 2: Envía un Ping

Descripción: El emisor de ping registra el tiempo antes de enviar un paquete *Ping*. El mismo transmite el paquete Ping.

Instrucción: Usa el tiempo de ejecución para registrar el tiempo de envío del *Ping*. Envía un paquete *Ping* a la micro:bit receptora.

6.7 Tarea 3: Recibe un Ping

Descripción: La micro:bit receptora responde un mensaje *Ping* con un *Pong*.

Instrucción: Programa la micro:bit receptora para transmitir un paquete *Pong* cuando se reciba un paquete *Ping*.

6.8 Tarea 4: Recibe un Pong y calcula un tiempo de ida y vuelta (round-trip-time)

Descripción: Cuando la micro:bit emisora recibe el *Pong*, calcula el tiempo de ida y vuelta.

Instrucción: Programa el remitente para recibir un paquete de *Pong*. Cuando se reciba el *Pong*, registra el tiempo usando la variable de tiempo de ejecución. Muestra la diferencia entre los tiempos de recepción y envío en tu pantalla. Ejecuta tu programa 5 veces y anota los tiempos de envío que ves en tu pantalla.

Responde estas dos preguntas:

1. Cuál es el tiempo mínimo y máximo de *round-trip-time* (RTT)?
2. Cuál es el tiempo promedio de RTT?

6.9 Ejercicios

Ejercicio 1.
Extiende tu programa ping para enviar automáticamente más de un mensaje *Ping*. Pruébalo con 10 Pings. Calcula el tiempo promedio de RTT de estos mensajes *Ping*.

Ejercicio 2.
El programa ping informa el tiempo de ida y vuelta (RTT). ¿Qué pasaría si quisieras calcular el momento en que el mensaje tomó una sola dirección? ¿Es posible calcular tiempos de ida? En otras palabras, ¿es posible calcular cuánto tiempo lleva enviar un *Ping* a la receptora? ¿O cuánto tiempo lleva un *Pong* del receptor al remitente?

6.10 Problemas

Problema 6.1 En la Figura 6.2, ¿qué es 216.58.213.100?

Problema 6.2 ¿Qué es el *round-trip-time o tiempo de ida y vuelta*, y cómo se calcula?

Problema 6.3 Piensa en el siguiente escenario: la micro:bit 1 envía un *Ping* a la micro:bit 2 en el tiempo 5. Si el tiempo de ida y vuelta (RTT) es 10, ¿en qué tiempo recibe la micro:bit 1 el mensaje *Pong*?

Problema 6.4 En la Figura 6.2, ¿cuáles son los tiempos mínimos y máximos de ida y vuelta (RTT)?

Problema 6.5 La figura 6.2 muestra 0% de pérdida. ¿Cuál sería el porcentaje de pérdida si se pierden 2 mensajes Ping de 5?

6.11 Recursos

- **Video: ¿Qué es un Ping? - https://youtu.be/N8uT7LNVJv4**

7. JUEGO 2: PIEDRA, PAPEL, TIJERAS POR LA RADIO

7. Introducción

¡Juguemos a un juego de piedra, papel, tijera! Este juego se juega con dos jugadores. Cada jugador, al mismo tiempo, forma una de las tres formas (piedra, papel o tijera) con sus manos. Luego, usan estas reglas para decidir quién gana:

- **La piedra rompe las tijeras.**
- **Las tijeras cortan el papel.**
- **El papel envuelve la piedra.**
- **Si ambos jugadores eligen la misma forma, es un empate.**

La Figura 7.1 muestra estas reglas.

En este capítulo, programarás este juego usando tus micro:bits. Al hacerlo, practicarás:

- ***La comunicación Unicast***
- **La programación con variables**
- **La programación con condicionales**

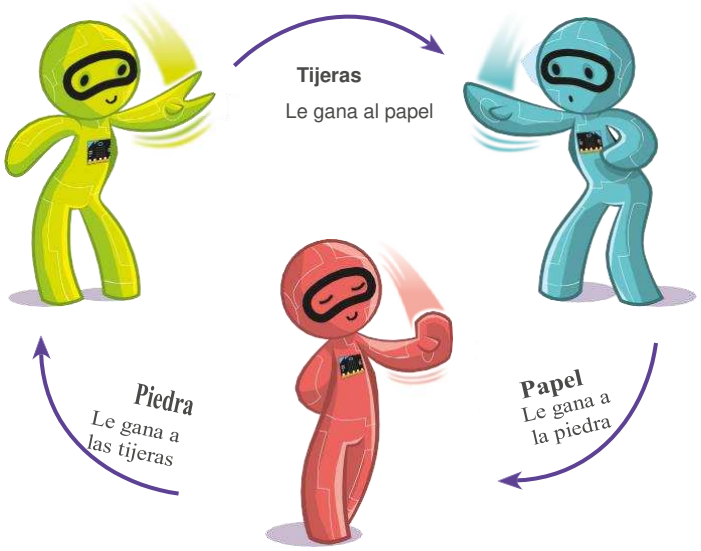


Figura 7.1: Juego de piedra, papel, tijeras. La piedra le gana a las tijeras. Las tijeras le ganan al papel. El papel le gana a la piedra.

7.2 Lo que necesitarás

- **2 micro:bits**
- **1 pizarra / pizarrón**
- **marcadores de pizarra / post-it**
- **1 compañero de equipo**

7.3 Programación: Piedra, papel, tijeras

Para programar este juego, lo mejor es trabajar con un compañero de equipo. La Tarea 1 es para familiarizarte con el juego y no usarás la radio. A partir de la Tarea 2, comenzarás a escribir las partes de tu programa para jugar este juego por la radio.

7.4 Tarea 1: Comienza con un juego simple

Descripción: Para familiarizarte con el juego, prueba la actividad Piedra, papel, tijeras en https://www.microbit.co.uk/blocks/lessons/rock-paper-scissors/activity. Ten en cuenta que el programa le da un número a piedra, papel y tijeras. Por ejemplo, papel = 0, piedra = 1 y tijeras = 2.

Instrucción: Programa el código que se muestra en la página de actividades de Piedra-Papel-Tijeras y descárgalo a tu micro:bit. Juega el juego con un amigo. Cada uno sacudirá sus micro:bits al mismo tiempo y luego decidirá quién gana usando las reglas que se muestran en la Figura 7.1.

7.5 Tarea 2: Trabajando con la radio por unicast

Descripción: Para jugar el juego por la radio, usarás el botón A para seleccionar papel, piedra o tijeras. Utilizarás el botón B para confirmar tu selección y enviarlo por la radio. Al igual que en la Tarea 1, usa papel = 0, piedra = 1 y tijeras = 2. Se enviará uno de estos números por la radio según la selección.

Instrucción: Escribe un programa para hacer lo siguiente:

1. Usa el botón A para seleccionar papel, piedra o tijeras. Cada vez que presiones el botón A, debería mostrar alternativamente un icono de papel, piedra o tijera.
2. Usa el botón B para confirmar tu selección, y transmítelo a la micro:bit de tu amigo por la radio como lo hiciste en el Capítulo 5.
3. Agrega un código para recibir un número. Cuando recibas un número, muestra el ícono correspondiente en la pantalla. Por ejemplo, si recibiste 0, muestra el icono de papel.

   Prueba con tu compañero de equipo que puede enviar y recibir sus valores a través de la radio.

7.6 Tarea 3: Completa la tabla de reglas

Descripción: Tu programa decide quién gana, cuando recibe un número de la micro:bit de tu compañero de equipo.

Instrucción: Para decidir quién gana, compara el número que elegiste con el número que recibiste. Proporcionamos una tabla de ejemplos incompleta para ayudarte a decidir el resultado en la Figura 7.2. Usando esta tabla, comparas *Mi mano* a la de tu oponente. Por ejemplo, si ambos números significan *Papel*, es un empate, y el resultado es una cara de *sorpresa*. Pero, si *Mi mano* es *Papel* y la mano del oponente es *Tijeras*, el resultado es una cara *triste*. Por el contrario, si *Mi mano* es *Tijeras* y la mano del oponente es *Papel*, entonces el resultado es una cara *feliz*. Usando las reglas en la Figura 7.1, llena el resto de la tabla.

		Mano del oponente	
Mi mano	**Papel (0)**	**Piedra (1)**	**Tijeras (2)**
Papel (0)			
Piedra (1)			
Tijeras (2)			

Figure 7.2: Tabla incompleta de piedra, papel, tijeras

7.7 Tarea 4: Juega

Descripción: Una vez que hayas llenado la tabla, debes decidir cómo programar estas reglas en tu código. Tu programa:

1. jugará el juego basado en las reglas de Piedra-Papel-Tijeras (ver Figura 7.1)
2. mostrará una *cara feliz* si ganaste, una *cara triste* si perdiste. Y si es un empate, mostrará una *cara de sorpresa*.

Instrucción: La Figura 7.3 muestra una plantilla para programar la tabla usando el bloque *si* en el menú *Lógica* del editor de bloques de JavaScript. Ten en cuenta que esto es solo una plantilla y está ahí para darte una idea de la estructura de tu programa. Por ejemplo, tu bloque “on radio received” tendrá que ser diferente para hacer la comunicación de unicast (vea el Capítulo 5).

Notarás que en la plantilla usamos dos variables: *selected* y *received*. La variable *Selected* se establece en *verdadero* cuando haces la selección de tu mano presionando el botón B. La variable *Recibido* se establece en *verdadero* cuando recibes la mano de tu oponente. En el bloque *para siempre*, el juego solo se juega cuando tanto las variables *Selected* como *Recibido* son *verdaderas*. Una vez que ingresas al bloque para jugar el juego, estas variables se restablecen a *falso* para la siguiente ronda.

Después de programar el juego, ¡juega con tu compañero de equipo! ¿Quién gana más seguido?

Figura 7.3: Piedra, papel y tijeras: una plantilla para programar las reglas

Ejercicio 1.
¿Cómo podrías expandir tu programa para jugar piedra / papel / tijeras / lizard / spock? Para obtener más información sobre esta extensión, consulta el sitio web: http://www.samkass.com/theories/RPSSL.html

7.8 Problemas

Problema 7.1 ¿Cómo se prueba un empate en tu programa?

Problema 7.2 Cómo cambia la Tabla 7.2, si papel = 2, piedra = 0 y tijeras = 1? Redibuja tu tabla.

Problema 7.3 Para jugar con un jugador diferente, ¿qué necesitas cambiar en tu programa? Recuerda que estás usando unicast para enviar tu mano.

Problema 7.4 ¿Qué sucede si le envías tu mano al otro jugador antes de que elijan la suya? ¿Habrá algún problema? ¿Podrían hacer trampa?

7.9 Recursos

- **Flash game: Piedra-Papel-Tijera: Tú vs. la Computadora - http://www.nytimes.com/interactive/science/rock-paper-scissors.html**

8. MANEJO DE ERRORES: RETRANSMISIONES

8. Introducción

En los capítulos anteriores, probablemente hayas notado que la comunicación inalámbrica no siempre es confiable. En otras palabras, no todos los mensajes que envías pueden ser recibidos por el otro lado. En este capítulo, aprenderás cómo aumentar las posibilidades de que se reciban tus mensajes. Entonces, ¿qué harías si se pierde un mensaje? Este capítulo cubrirá un método simple pero efectivo: retransmisiones.

En resumen, aprenderás sobre:

- **Errores de comunicación inalámbrica**
- **Las *Retransmisiones* como forma de mejorar la confiabilidad**

8.2 Lo que necesitarás

- **2 micro:bits**
- **1 compañero de equipo**

8.3 Contexto

En las comunicaciones inalámbricas, puede haber un error por varias razones. Por ejemplo, puede haber obstrucciones físicas, como paredes, puertas e incluso personas. ¡Las señales inalámbricas pierden potencia a medida que atraviesan estas obstrucciones y, a veces, rebotan! Cuantas más obstrucciones hay entre un emisor y un receptor, hay más posibilidades de que haya un error. Además, si el emisor y el receptor están demasiado lejos el uno del otro, es posible que no siempre puedan comunicarse. Imagina que hay muchos obstáculos entre dos personas, ¡es posible que no siempre escuchen lo que el otro está diciendo!

Otra razón para un error inalámbrico puede ser la *interferencia de radio*. Esto se debe a que se transmite la comunicación inalámbrica (recuerda el Capítulo 2). Esto significa que puede haber muchos broadcasters, y sus transmisiones pueden colisionar en los receptores. Estas emisoras *interfieren* entre sí.

Definición 1. — Interferencia.
En las comunicaciones inalámbricas, la interferencia es cualquier otra señal que interrumpe una señal mientras viaja a su destino.

Imagina un salón de clases donde todos hablan al mismo tiempo. Te perderás la mitad de las cosas que dice tu amigo. Las señales de otras personas interfieren con la señal de tu amigo en su camino hacia ti. En las redes, esto es una *pérdida de paquetes.*

Definición 2 — Pérdida de paquetes.
La pérdida de paquetes ocurre cuando uno o más paquetes de datos que viajan en una red informática no llegan a su destino. La pérdida de paquetes se mide como la proporción de paquetes perdidos y los paquetes enviados (ver Ecuación 8.1).

$$\text{Pérdida de Paquetes} = \frac{\text{Paquetes perdidos}}{\text{Paquetes enviados}} \quad (8.1)$$

Además, si hay demasiada interferencia, puedes recibir mensajes incorrectamente. Volviendo al ejemplo del aula, esto es como escuchar "Bota" cuando tu amigo grita "Gota". En las redes, esto es un error de paquete. Los errores de paquetes se miden como *tasas de error de paquete*.

Definición 3. — Tasas de error de paquete.
La tasa de error del paquete es la proporción de paquetes que se han recibido con uno o más errores y los paquetes enviados.

$$\text{Tasa de error de paquetes} = \frac{\text{Paquetes con error}}{\text{Paquetes enviados}} \quad (8.2)$$

En este capítulo, cubriremos un método simple para manejar estos errores, las retransmisiones, en las que el remitente retransmite automáticamente los mensajes varias veces para aumentar las posibilidades de recepción.

Definición 4. — Retransmisiones.
Las retransmisiones significan enviar mensajes muchas veces.

Figura 8.1, supongamos que el remitente sabe que el medio de comunicación pierde la mitad de sus paquetes. En otras palabras, la pérdida de paquetes es 0.5 (o 50%). La micro:bit emisora decide enviar cada paquete dos veces, para aumentar la posibilidad de que sus mensajes se transmitan. El primer paquete es la transmisión, y el segundo paquete es la retransmisión. Entonces, la cantidad de retransmisiones es 1.

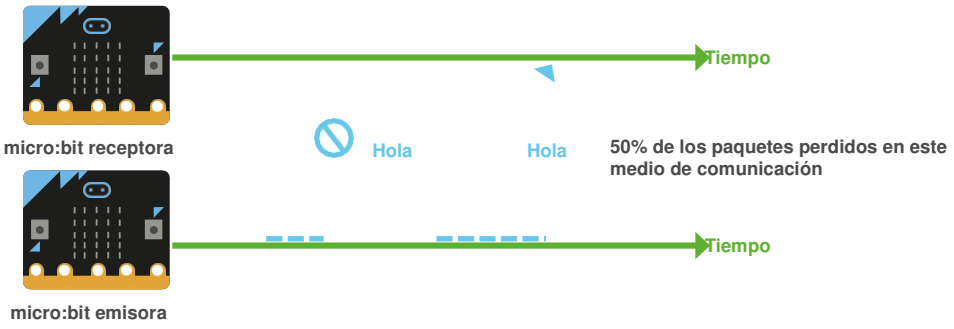


Figura 8.1: Las Retransmisiones pueden aumentar el éxito del mensaje. En el ejemplo, el remitente envía cada mensaje dos veces de manera predeterminada. Entonces, incluso si el primer "Hola" falló, ¡el receptor recibió el segundo "Hola"!

Es común usar retransmisiones combinadas con otro método. Por ejemplo, los remitentes pueden optar por retransmitir solo cuando están seguros de que ha habido un error. Exploraremos esta opción en el próximo capítulo, Capítulo 9.

8.4 Programación: Retransmisiones

Esta actividad se realiza mejor con un compañero de equipo. Comenzarás con la creación de errores de paquete en la Tarea 1, y luego probarás diferentes tasas de error de paquetes en la Tarea 2. En la Tarea 3, programarás la solución de retransmisiones para manejar estos errores. En esta tarea, también ejecutarás una serie de experimentos para medir qué tan bien funcionan las retransmisiones.

8.5 Tarea 1: Crea paquetes de errores

Descripción: En la comunicación inalámbrica, los paquetes pueden fallar aleatoriamente. Esto puede dificultar la prueba de su código para este capítulo. Con la finalidad de probar los errores, usarás un bloque personalizado en el editor de bloques de JavaScript para enviar mensajes con errores deliberados.

Las funciones de ErrorRadio son como las funciones de Radio pero tienen un parámetro de error adicional. Este parámetro se establece en 20 de forma predeterminada, lo que significa que, en promedio, fallan 20 paquetes de 100. Por lo tanto, la tasa de error del paquete es 0: 2 o 20%.

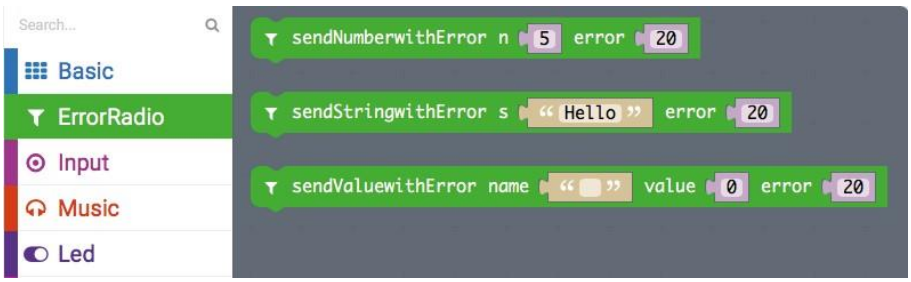


Figura 8.2: Bloques personalizados "ErrorRadio"

Instrucción:

Para usar los bloques personalizados en el editor de Bloques JavaScript, importa el archivo ErrorRadio.hex en tu editor de Bloques JavaScript https://microbit.nominetresearch.uk/networking-book/ErrorRadio.hex haciendo clic derecho sobre el enlace y seleccionando "Guardar enlace como…", para descargar el archivo. Con tu compañero de equipo, decide quién tendrá la micro:bit emisora y quién tendrá la micro:bit receptora. Sigue el enfoque del Capítulo 5 para poner direcciones de remitente y receptor en tus paquetes.

Puedes copiar y cambiar uno de los programas que has hecho para el Capítulo 5 para usar los bloques *ErrorRadio*. Escribe un programa de remitente simple que envíe un número con un error. Descárgalo a la micro:bit emisora. Escribe un programa receptor simple que reciba un número y lo muestre en la pantalla. Descarga este programa a la micro:bit receptora.

Cambia la tasa de error del paquete usando estos valores de error en tu programa: 0, 50 y 100. Prueba los errores del paquete observando la pantalla del receptor.

8.6 Tarea 2: Envía una secuencia de mensajes

Descripción: En esta sección, enviarás una secuencia de mensajes a la micro:bit receptora.

Instrucción: Extiende tu programa en la Tarea 1, para enviar esta secuencia:

Start 1 2 3 4 5 6 7 8 9 10 End

Puedes enviar el *Start* y el *End* utilizando los bloques de radio normales para enviarlos sin un error. Pero recuerda que la radio de tu micro:bit puede soltar mensajes. Por lo tanto, puede haber errores incluso al enviar el *Start* y el *End*. ¡Ninguna radio es perfecta!

Extiende el programa receptor para contar la cantidad de mensajes que recibe en esta secuencia. Haz experimentos configurando el parámetro de error en 25, 50 y 75. Calcula la pérdida de paquetes usando la Ecuación 8.1. Repite cada experimento tres veces. Completa la Tabla 8.1 con el resultado de tus experimentos. Por ejemplo, cuando el error se establece en 25, y tu recibes:

Start 1 5 6 7 8 9 10 End

Esto significa que recibiste 7 paquetes y perdiste 3. Tu pérdida de paquete es 0.3. La primera fila de la tabla se completa según este ejemplo. Agrega los valores de tu propio experimento. En función de los resultados de tu experimento, analiza con tu compañero de equipo cómo cambian los resultados del experimento a medida que cambia el valor del *error*.

Valor Error	Experimento n°.	Paquetes Recibidos	Paquetes perdidos
25	(ejemplo)	7	0,3
25	1		
25	2		
25	3		
50	1		
50	2		
50	3		
75	1		
75	2		
75	3		

Tabla 8.1: Resultado de los experimentos

8.7 Tarea 3: Retransmisión por defecto

Descripción: En esta tarea, programarás retransmisiones automáticas en el lado del remitente.

Instrucciones: Cambia el código del remitente de la Tarea 2 para enviar cada número en su secuencia más de una vez. Para probar tu código, establece el *error* en 75. Por ejemplo, estableciendo el número de retransmisiones en 1, enviarás la siguiente secuencia:

Start 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 End

Esto significa que el emisor envió 20 paquetes en total, de los cuales 10 son retransmisiones.

Cambia tu código de receptor para contar los números únicos recibidos. Cuenta también los duplicados. Calcula la pérdida del paquete. Por ejemplo, supongamos que tu receptor recibió, para el caso de una retransmisión:

Start 1 1 2 3 5 5 6 8 9 9 10 End

Esto significa que el receptor recibió 8 números únicos (1, 2, 3, 5, 6, 8, 9 y 10) y 3 duplicados (1, 5 y 9). Ten en cuenta que la pérdida de paquetes es de 9 paquetes de 20 (0,45). Pero con las retransmisiones, el receptor solo perdió 2 números de 10 (no recibió 4 y 7). Llamaremos a esta pérdida de paquetes de información mejorada, *pérdida de información*. Entonces, la pérdida de información con retransmisiones es 0.2. La primera fila de la tabla 8.2 se completa según este ejemplo.

Ejecuta cada experimento tres veces cada uno, para diferentes valores de retransmisión y completa el resto de la tabla.

Retransmisiones	Experimento n°.	Paquetes únicos recibidos	Duplicados	Pérdida de paquetes	Pérdida de información
1	(ejemplo)	7	3	0.45	0.2
1	1				
1	2				
1	3				
3	1				
3	2				
3	3				
5	1				
5	2				
5	3				

Tabla 8.2: Resultado de los experimentos

8.8 Actividad extensiva

Ejercicio 1.

- **En función de tus experimentos, analiza con tu compañero de equipo cómo ayuda el aumento de las retransmisiones. En la discusión, responde las siguientes preguntas:**
- **¿Cómo se reduce la pérdida de información a medida que aumenta la cantidad de retransmisiones?**
- **¿El método garantiza que todos los mensajes se reciben al menos una vez?**
- **¿Cómo mejorarías el método de retransmisiones?**

Ejercicio 2.

Imagina que vas a estudiar la pérdida de paquetes en diferentes ubicaciones dentro de una habitación utilizando dos micro:bits. Escribe un programa receptor y otro de envío para medir la pérdida de paquetes. ¿Qué observas? ¿Cómo cambia la pérdida de paquetes en diferentes ubicaciones?

8.9 Problemas

Problema 8.1 ¿Qué es interferencia? ¿Por qué sucede?

Problema 8.2 Si el remitente envió 20 mensajes y se perdieron 11 mensajes en el camino al destino, ¿cuál es la pérdida de paquetes?

Problema 8.3 Si la tasa de error del paquete es 0.2 y el emisor envió 40 paquetes, ¿cuántos paquetes tenían errores?

Problema 8.4 Supongamos que no sabes cuántos números habrá en una secuencia de mensajes, pero sabes que los números comenzarán en 1 y se incrementarán en 1. Por ejemplo, la secuencia de mensajes enviados puede ser:

Start 1 2 3 4 5 6 7 8 9 10 11 12 End

¿Qué sucede si pierdes los mensajes de inicio o fin? ¿Qué es peor: la pérdida de un mensaje de inicio, o fin? Si el único mensaje que recibe es un 4, ¿qué puede decir sobre la cantidad de mensajes perdidos?

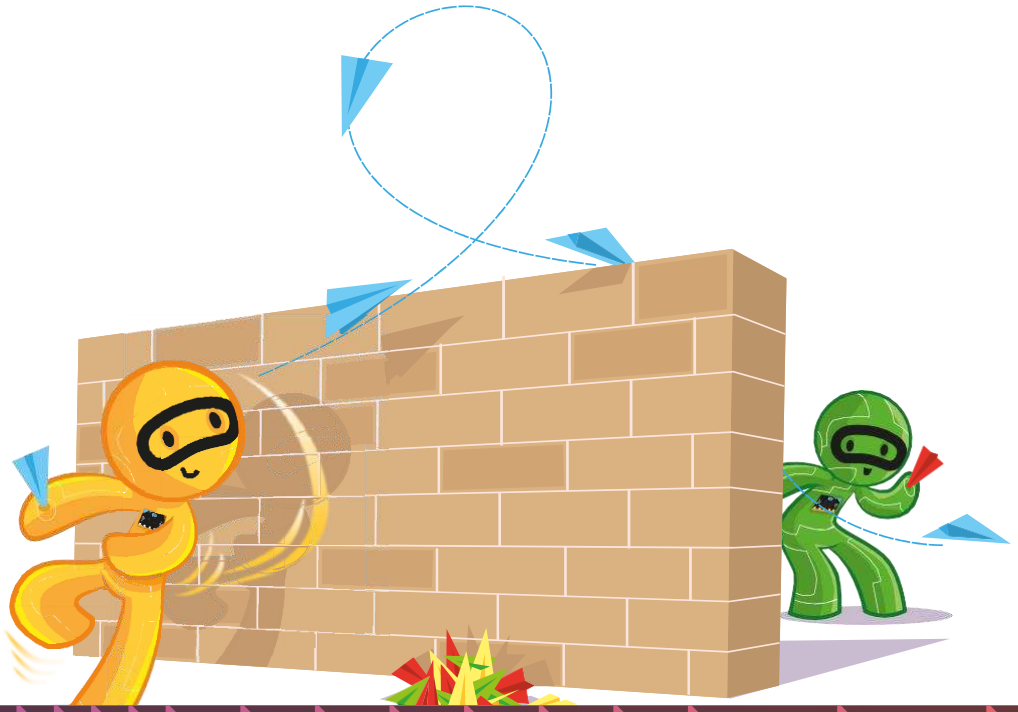
Problema 8.5 Supongamos que no sabes cuántos números habrá en la secuencia de mensajes. Y ellos no siguen ningún orden. Por ejemplo, la secuencia de mensajes enviados puede ser:

Start 3 5 10 2 End

¿Qué sucede si pierdes los mensajes de inicio o fin en la secuencia? ¿Qué es peor: la pérdida de un mensaje de inicio, o fin? Si el único mensaje que recibes es un 5, ¿qué puedes decir sobre la cantidad de mensajes perdidos?

8.10 Recursos

- **Video: Internet: Paquetes, Enrutamiento, y Confiabilidad - https://youtu.be/AYdF7b3nMto**



9. ERRORES DE MANEJO:ACUSE DE RECIBO

9. Introducción

En el capítulo anterior, has usado retransmisiones para tratar los errores de transmisión inalámbrica. En este capítulo, mejorarás al usar acuse de recibo. Al realizar esta actividad, aprenderás varios métodos y protocolos clave para el control de errores en las redes.

En resumen, aprenderás:

- **El concepto de *acuse de recibo***
- ***El concepto de Automatic Repeat Request/ARQ (Solicitud de repetición automática)***
- ***El protocolo Parar y Esperar***

9.2 Lo que necesitarás

- **2 micro:bits**
- **1 compañero de equipo**

9.3 Contexto

En el capítulo anterior, un mensaje se transmitió varias veces, incluso si el receptor ya recibió una copia anterior. ¡Esto es un desperdicio! Podrías haber estado transmitiendo nueva información en lugar de repetirte. Esto también es un desperdicio para el receptor, que necesita seguir descartando los duplicados.

> **Definición 1 — Acuse de recibo (ACK).**
> Los acuses de recibo son pequeños mensajes que el receptor envía de vuelta, para decirle al remitente que recibió un mensaje. El remitente entonces sabe que no necesita retransmitir y está listo para enviar el siguiente mensaje.

Para evitar esto, presentaremos un nuevo concepto llamado *acuse de recibo*. Si el remitente no recibe un acuse de recibo, debe retransmitir su mensaje.

Pero, ¿cuánto tiempo debe esperar el emisor para recibir un acuse de recibo? Esto se especifica por un *tiempo de espera*.

Definición 2. — Tiempo de espera.
El tiempo de espera es la cantidad de tiempo permitido para que el remitente abandone la espera de un acuse de recibo.

En otras palabras, si el remitente no recibe un acuse de recibo dentro de un período de tiempo de espera, decidirá que el paquete se debe haber perdido.

Los acuses de recibo se utilizan en un método de control de errores llamado *Solicitud de Repetición Automática (ARQ).*

Definición 3 — Automatic Repeat Request /ARQ (Solicitud de Repetición Automática)
La Solicitud de Repetición Automática es un método de control de errores. Utiliza acuses de recibo y tiempos de espera para retransmitir paquetes. Las retransmisiones pueden continuar hasta que el emisor reciba un acuse de recibo o se alcance un número máximo.

La ARQ se usa tanto en Internet como en redes móviles.

En su forma más simple, una Solicitud de Repetición Automática usa *el protocolo ARQ Parar y Esperar.*

Definición 4 — Protocolo ARQ Parar y Esperar.
En el protocolo ARQ Parar y Esperar, el remitente:
1. envía un paquete
2. espera el acuse de recibo (ACK) pero se da por vencido después del período de tiempo de espera
3. si el tiempo de espera se acaba, ve al paso 1
4. si hay un ACK, obtienes un nuevo paquete, ve al paso 1.

En el *protocolo Parar y Esperar*, el remitente no puede enviar un nuevo paquete hasta que reciba el acuse de recibo del anterior.

Entonces, ¿cómo maneja el *protocolo Parar y Esperar* las pérdidas de paquetes? A continuación, veremos algunos ejemplos.

La Figura 9.1 muestra un ejemplo de una transmisión exitosa. El remitente envía "Hola" y el receptor responde con un ACK. El remitente recibe el ACK antes de que termine el tiempo de espera, por lo que sabe que el paquete se recibió correctamente. Ahora, el remitente puede comenzar a enviar otro mensaje.

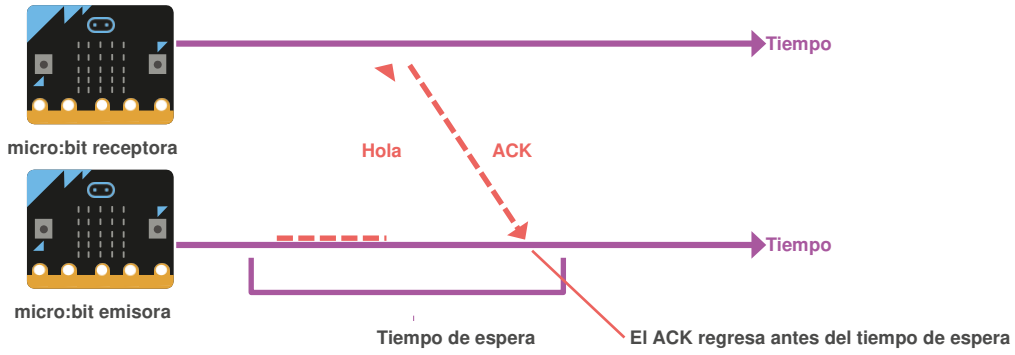


Figura 9.1: Protocolo ARQ Parar y Esperar: el receptor envía un ACK al remitente, por lo que el remitente sabe que el mensaje "Hola" llegó correctamente.

Ahora, veamos algunos casos de error. La figura 9.2 muestra que se pierde el primer mensaje del remitente.

Entonces, el receptor no envía un ACK. Cuando finaliza el tiempo de espera, el remitente no ha recibido un ACK. Por lo tanto, retransmite el mensaje. El segundo intento es exitoso y el emisor recibe un ACK a tiempo (antes del tiempo de espera).

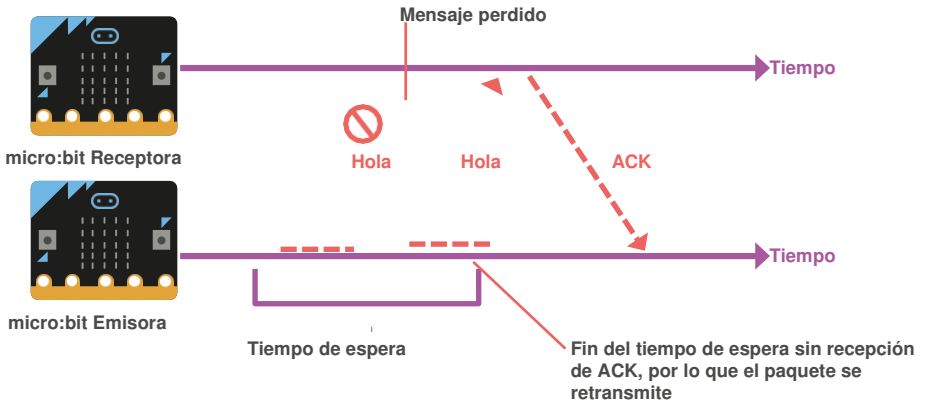


Figura 9.2: Protocolo ARQ Parar y Esperar: el mensaje se pierde, por lo que el remitente lo retransmite.

La Figura 9.3 muestra un ejemplo donde se recibe el mensaje del remitente, pero se pierde el ACK del receptor. Nuevamente, cuando termina el tiempo de espera, el remitente no ha recibido un ACK. Entonces, retransmite su mensaje. El receptor recibe el mensaje duplicado y nuevamente, envía un ACK. Esta vez, el ACK tiene éxito y las cosas pueden ir normalmente.

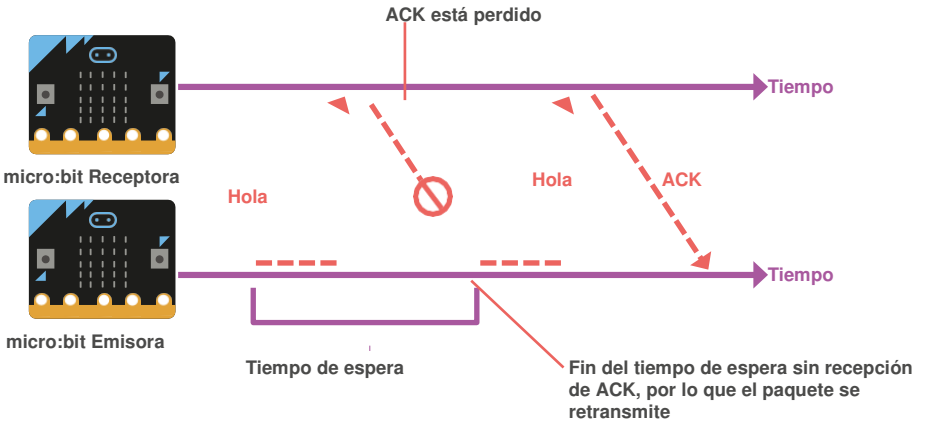


Figura 9.3: Protocolo ARQ Parar y Esperar: el mensaje fue recibido, pero el ACK se pierde, por lo que el remitente retransmite el mensaje.

Estos ejemplos muestran que el protocolo ARQ Parar y Esperar maneja bastante bien el paquete de datos y las pérdidas de ACK. Pero, ¿siempre funciona? La figura 9.4 muestra un problema que puede ocurrir cuando los mensajes o ACK se retrasan. En otras palabras, los tiempos de espera finalizan antes de que se puedan recibir los ACK.

En el ejemplo, cuando el emisor envía el primer "Hola", el receptor recibe este mensaje y devuelve un ACK. Pero el emisor agota el tiempo de espera antes de recibir este ACK. Por lo tanto, retransmite el segundo "Hola". Luego, recibe el mensaje ACK retardado. ¿Pero a qué paquete se refiere este ACK? ¿El primer "Hola", o el segundo? ¡Esto confunde al receptor también! ¿El segundo "Hola" es un nuevo paquete o un duplicado? Para resolver esta confusión, el protocolo necesita usar *números de secuencia*.

Definición 5 — Números de Secuencia

Un número de secuencia es un número elegido por el emisor e incluido en el encabezado del paquete. Cuando el receptor envía un ACK, incluye el siguiente número de secuencia para decirle al remitente que recibió el primer paquete y está listo para el siguiente.

Por ejemplo, cuando el remitente envía "Hola, 0", este es un mensaje de "Hola" con un número de secuencia 0. Al recibir este paquete, el receptor enviará "ACK, 1", que dice "Recibí el paquete 0, envíe mi paquete 1 a continuación".

No utilizaremos números de secuencia en las tareas de esta sección, pero podrías intentar agregarlos como una actividad extendida.

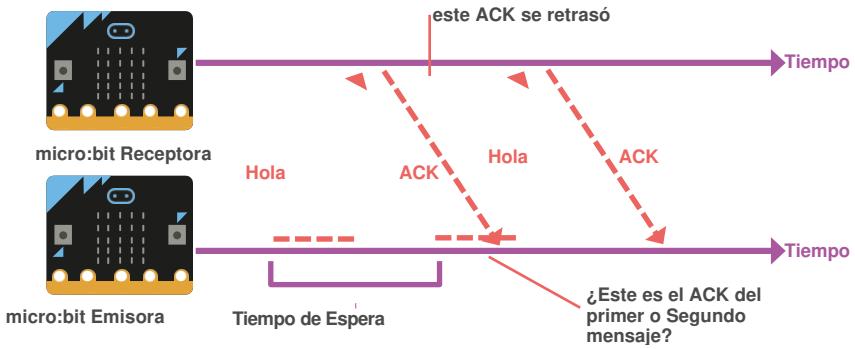


Figura 9.4: Protocolo ARQ Parar y Esperar: ¿Qué sucede si un mensaje se retrasa? No está claro cuál ACK se refiere a qué mensaje.

9.4 Programación: ¡Para y Espera!

Para programar el protocolo ARQ Parar y Esperar, trabajarás con un compañero de equipo. Al igual que en el Capítulo 8, usarás los bloques personalizados *ErrorRadio* para enviar mensajes con errores. La comunicación es unicast, por lo que aún usarás direcciones de origen y destino en tus mensajes tal como lo hiciste en el Capítulo 5. ¡No olvides que tus receptores deben verificar si los mensajes recibidos están dirigidos a ellos!

9.5 Tarea 1: Diseña tus datos y tus paquetes ACK

Descripción: Antes de que puedas enviar y recibir paquetes, primero deberás decidir cómo serán tus datos y tus paquetes ACK.

Instrucción: Discute cuál es la información mínima que debes tener en tus paquetes. Crea dos variables de cadena para datos y paquetes ACK, usando los bloques de Texto en el editor de Bloques JavaScript.

9.6 Tarea 2: Tiempo de espera y retransmisión

Descripción: Para programar el *Parar y Esperar*, necesitas un mecanismo de tiempo de espera. Después de cada transmisión, debes esperar el ACK o el tiempo de espera. Deberás decidir cuánto tiempo debe durar el tiempo de espera.

Instrucción: Para realizar esta tarea, puedes comenzar desde cero o cambiar el código del Capítulo 8 para la micro:bit emisora. En el lado del remitente, programa cómo esperar el ACK. En el menú Básico, la función de pausa será útil para el mecanismo de tiempo de espera. Si tu pausa finaliza antes de recibir un acuse de recibo, debes retransmitir el paquete. Si recibes el ACK antes de que finalice la pausa, debes recordar esta información cuando la pausa finalice y usarla para enviar tu próximo mensaje.

Para probar el programa, también debes programar el receptor. El receptor envía un paquete ACK por cada paquete de datos que recibe.

9.7 Tarea 3: Prueba la confiabilidad de *Para y Espera*

Descripción: En esta tarea, experimentarás con el protocolo Parar y Esperar que programaste. Para esto, agregarás un contador en el lado del remitente para contar el número de retransmisiones. En el lado del receptor, necesitas un contador para comprender el efecto que los acuses de recibo tienen en las retransmisiones.

Instrucción: Decide el tiempo de parar / esperar. Envía cinco números a la micro:bit de tu compañero de equipo usando el protocolo Parar y Esperar. Ejecuta el protocolo con diferentes valores de error (25 y 75), repitiendo cada experimento tres veces.

En la tabla, las retransmisiones son la cantidad de veces que un paquete debe ser reenviado. Los duplicados son la cantidad de veces que el receptor recibió retransmisiones innecesarias. Asumamos que el remitente envió lo siguiente:

1_1_1_2_2_3_4_4_4_4_5_5

Las retransmisiones están subrayadas: hubo 7 retransmisiones. Y el receptor recibió lo siguiente:

1_2_2_3_4_5_5

Los duplicados están subrayados: se recibieron 2 duplicados. La primera fila de la Tabla 9.1 se rellena como un ejemplo. Usa los resultados de tu experimento para completar el resto. Al comparar las retransmisiones con los duplicados, analiza qué tan bueno es el protocolo para manejar los errores.

Valor de Error	N° Experimento	Retransmisiones	Duplicados
25	(ejemplo)	7	2
25	1		
25	2		
25	3		
75	1		
75	2		
75	3		

Tabla 9.1: Resultados de los experimentos

9.8 Actividad extensiva

Ejercicio 1.
Discute cómo los acuses de recibo funcionan mejor que usar solo retransmisiones. ¿Ves algún problema en el uso de acuses de recibo?

Ejercicio 2.
Discute cómo la duración del período de tiempo de espera afecta el protocolo. Por ejemplo, ¿qué sucede si tu tiempo de espera es demasiado corto o demasiado largo? ¿Qué sucede si los acuses de recibo se retrasan?

Ejercicio 3.
Investiga el "Protocolo de bits alternos", que utiliza secuencias de números de 1 bit para ayudar con los problemas que se analizan en la figura 9.4.

9.9 Problemas

Problema 9.1 ¿Qué quiere decir ARQ?
Problema 9.2 En el protocolo ARQ Parar y Esperar, si se envían 10 paquetes, ¿cuántos acuses de recibo se necesitan?

9.10 Recursos

- **Video: Internet: Paquetes, Enrutamiento, y Confiabilidad - https://youtu.be/AYdF7b3nMto**

10. JUEGO 3: BATALLA NAVAL POR LA RADIO

10. Introducción

En esta actividad, programarás la versión micro:bit de un famoso juego clásico llamado *Battleship* o Batalla Naval. Batalla Naval se juega desde la Primera Guerra Mundial con lápiz y papel[1]. Un juego de mesa de plástico fue lanzado en 1967, y ahora hay muchas versiones electrónicas y aplicaciones[2].

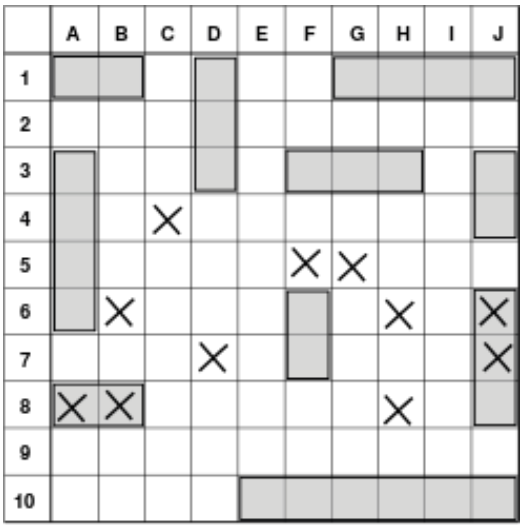


Figura 10.1: Tablero de Batalla Naval.

Veamos cómo funciona este juego, usando el tablero de ejemplo de la figura 10.1. En este ejemplo, cada jugador usa su propio tablero de 10x10, y la flota de cada jugador tiene 10 naves de diferentes tamaños (los rectángulos grises) colocados en el tablero: 4 naves de tamaño 2, 3 naves de tamaño 3, 2 naves de tamaño 4, y 1 buque de tamaño 6. La disposición de los buques está, por supuesto, oculta al oponente. Una vez que ambos jugadores hayan colocado sus naves en las casillas, comienzan a adivinar la posición de las naves de sus oponentes disparando. En el tablero de ejemplo, las cruces marcan los disparos del oponente. Ten en cuenta que algunas de estas cruces no alcanzaron ningún barco, y algunas lo hicieron. El oponente ha hundido el barco en las casillas 8A-8B. El barco en las casillas 6J-7J-8J fue golpeado dos veces, y otro disparo en 8J lo hundirá. Los jugadores también mantienen un segundo tablero para marcar los tiros que ya han probado.

[1] https://en.wikipedia.org/wiki/Battleship_(game)
[2] Por ejemplo, ver: https://battleship-game.org y http://www.mathplayground.com/battleship.html

Registran cada disparo, para poder decidir qué disparar a continuación.

Para programar Batalla Naval en las micro:bits, usarás tus conocimientos de redes. Este juego requiere la comunicación unicast y bidireccional, que aprendiste en los Capítulos 5 y 6. Si programas la variante en el Ejercicio 10.1, usarás tu aprendizaje de los Capítulos 8 y 9. En resumen, practicarás:

- **El concepto de *comunicación unicast*, *comunicación bidireccional y retransmisiones***
- **Enviando y recibiendo mensajes**
- **Entradas de botones**
- **La pantalla y sus coordenadas**
- **Variables y números aleatorios**
- **Matrices**
- **Bucles**

10.2 Lo que necesitarás

- **2 micro:bits**
- **1 compañero de equipo**

10.3 Cómo funciona el juego

Usando la pantalla de la micro:bit como una placa de Batalla Naval: ¡Dado que una micro:bit solo tiene una pantalla de 5x5, la placa de tu nave de guerra debe ser más pequeña! Esto no permite muchos barcos. Entonces, tu flota será de 5 naves, cada una con un tamaño de solo 1.

Cuando disparas un tiro, necesitarás saber si diste en el objetivo o no (*Tocado o Agua*). Por lo tanto, debemos reservar la fila superior (como en la figura 10.2) para mostrar los *Tocados* y *Agua*. Si la micro:bit de tu oponente dice que tienes un *Tocado*, tu micro:bit encenderá el LED que está más a la izquierda en la fila superior. Si desafortunadamente fue *Agua* tu micro:bit encenderá el LED derecho.

Como tu micro:bit posee una pantalla limitada, no podrás mostrar tus *Tocados* y *Agua* en la pantalla. Tal vez sea un desafío de memoria que se puede agregar al juego, o puedes hacer un seguimiento en papel, como los niños que jugaron el juego en épocas anteriores.

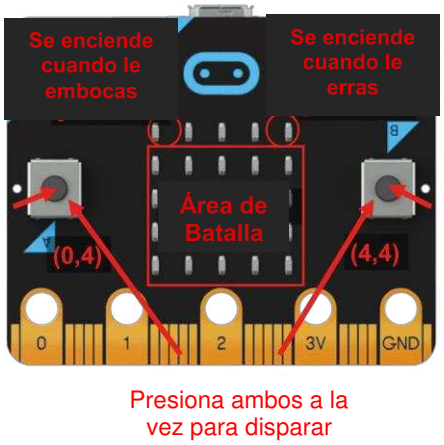
Disparos: Para dar disparos, usarás los botones: Primero seleccionarás un número de fila y columna para elegir un objetivo, y luego presionarás ambos botones para disparar. Ten en cuenta que cuando las coordenadas LED se dan como "(x, y)", x es el número de columna e y es el número de fila, y los números comienzan en cero. Para obtener más información, consulta https://www.microbit.co.uk/device/screen . El botón A se usará para seleccionar el número de columna y el botón B se usará para seleccionar el número de fila. Entonces, para disparar un tiro a (2,3), tendrás que presionar el botón A dos veces, presionar el botón B tres veces y luego presionar ambos botones A y B juntos. Para verificar tu comprensión, habla con tu compañero de equipo sobre cómo puede dar un disparo a (0,4).

Cuando presiones ambos botones para disparar un tiro, tu programa enviará un mensaje a la micro:bit de tu oponente. Entonces, por ejemplo, si quieres disparar un tiro a (4,4), enviarás las coordenadas (4,4).

Cuando la micro:bit de tu oponente recibe un disparo, verificarás si es un *Tocado* o *Agua*: te enviará un mensaje con su radio diciendo que es "*Tocado*" o "*Agua*".

Cuando recibas un "*Tocado*", tu micro:bit encenderá el LED en la esquina izquierda de la fila superior. Cuando recibas "*Agua*", tu micro:bit encenderá el LED en la esquina derecha de la fila superior.

Figura 10.2: Batalla Naval en la micro:bit



10.4 Una muestra del juego

Veamos cómo se verán las cosas en tu micro:bit. Al principio, tendrás todas tus naves colocadas en las 4 filas inferiores, como en la Figura 10.3. Entonces, ambos jugadores tienen 5 naves colocadas en el área de batalla.

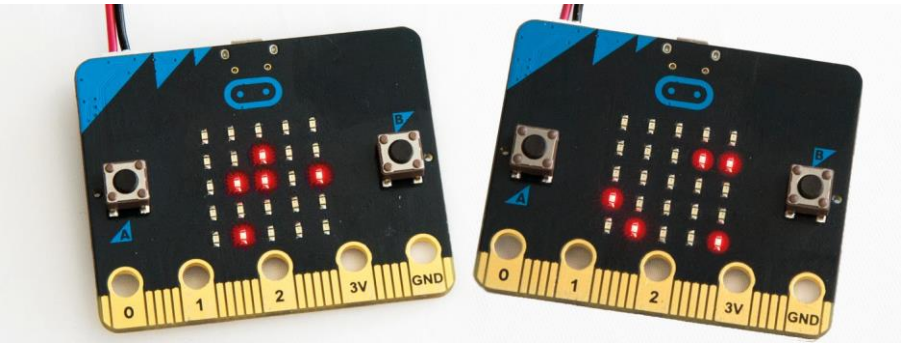


Figura 10.3: Juego de Batalla Naval: etapa inicial con naves colocadas al azar.

El atacante (a la izquierda) presiona el botón A tres veces y el botón B una vez. Al presionar ambos botones al mismo tiempo, se dispara y se envía un mensaje por la radio para la posición (3,1). Hay un barco en este lugar, ¡y esto es un éxito! Entonces, en la figura 10.4, se ilumina el LED situado más a la izquierda en la fila superior de la micro:bit del atacante. Y en la pantalla del oponente, el LED en la posición (3,1) se apaga, porque este barco se hundió.

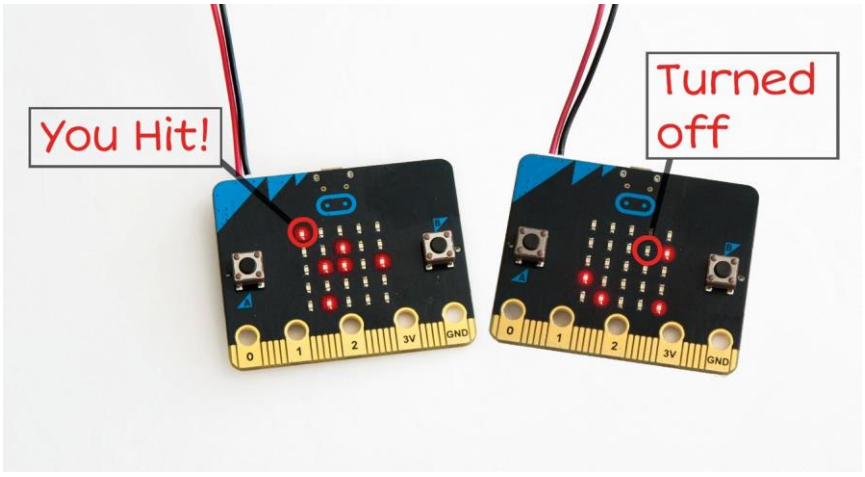


Figura 10.4: Juego Batalla Naval: ¡éxito! ¡Has hundido un barco!

Veamos también una situación de error (mira la Figura 10.5). En este caso, nada debería cambiar en el tablero del oponente. Pero en la pantalla del atacante, en la fila superior, el LED de la derecha se enciende para mostrar que falló el disparo.

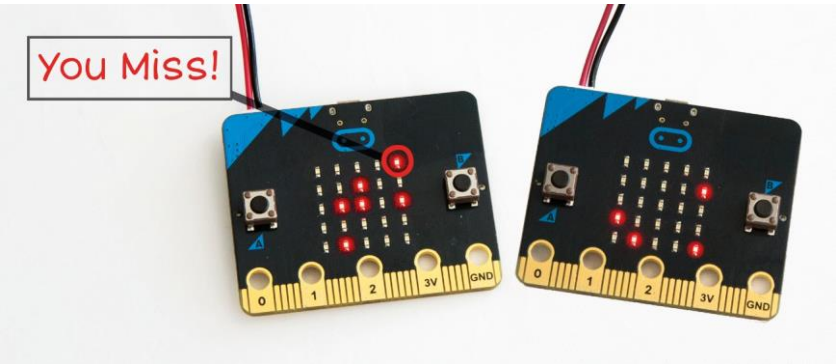


Figura 10.5: Juego Batalla Naval: ¡Un disparo desafortunado!

10.5 Programación: Batalla Naval

Batalla Naval es un juego de dos personas. Ambos jugadores pueden ejecutar programas idénticos, o cada uno puede programar su propia versión, siempre que acepten los detalles de los mensajes de radio. Al escribir un programa más complejo como este, te resultará más fácil si lo divides en partes y pruebas cada parte a medida que lo escribes. (¡Esta es una habilidad valiosa en la medida que aprendes más sobre la programación!)

Para ayudar con esto, hemos dividido el programa en cuatro tareas: una vez que hayas completado la tarea final, podrás jugar el juego con tu compañero de equipo. Si encuentras algún error en tu programa, trabaja con tu compañero de equipo para arreglarlo hasta que el juego se desarrolle como se describe en la Sección 10.2.

10.6 Tarea 1: Configura el juego

Descripción: Esta parte debe hacerse antes de que comience el juego. Colocarás 5 naves en tu tablero. Piensa en colocar aleatoriamente 5 puntos en el área de batalla, que es una matriz de 4 x 5. Responde las siguientes preguntas:

- **¿Cómo representarás el área de batalla en tu programa como una estructura de datos?**
- **¿Cómo seleccionarás coordenadas al azar (*número_ columna*) para 5 naves, donde *número_ columna* es 0..5 y *número_fila* es 1..5?**
- **¿Cómo representarás la información de que hay un barco en cada una de estas coordenadas?**

También configurarás tu configuración de radio y paquete para enviar mensajes unicast.

Instrucción: Crea las estructuras de datos y las variables necesarias que representan las naves en el campo de batalla. Configura tu radio y paquetes para comunicación unicast.

Prueba si tu programa muestra 5 naves al azar en las 4 filas inferiores de la pantalla, como en la Figura 10.3.

10.7 Tarea 2: Dispara

Descripción: Cuando se presiona el botón A, define el número de columna para un disparo. Para ello necesitas contar cuántas veces se presionó este botón para obtener el número de columna. Cuando se presiona el botón B, define el número de fila para el mismo disparo. Nuevamente, cuenta la cantidad de veces para obtener el número de fila.

Importante: Si no presionas ni el botón A ni el botón B, *número_ columna* = 0 y *número_fila* = 0. Un disparo con *número_fila* = 0 es un tiro desperdiciado porque no puede haber naves en esa fila superior. También asegúrate de que si se presiona cualquiera de los botones más de cuatro veces, deberías comenzar a contar nuevamente desde 0. En otras palabras, los contadores de botones deben incrementarse presionando cada botón de la siguiente manera: 0, 1, 2, 3, 4, 0, 1, 2, 3, 4 etc.

Al presionar ambos botones al mismo tiempo se realiza el disparo, por lo que tu programa debe enviar *número_ columna* y *número_fila* a través de la radio a tu oponente. Decide cómo enviar este mensaje en un paquete, y acuerda esto con tu compañero de equipo si están escribiendo programas por separado.

Instrucción: Programa el botón para A, B y A + B. La sección del programa para los botones A + B enviará un mensaje de radio.

Para probar la exactitud de tu código, agrega un pequeño código de prueba para que cuando dispares, además de enviar un mensaje de radio, también encienda el LED en (*número_ columna*, *número_fila*). Usa esto para verificar que el apuntamiento esté funcionando correctamente. Esto es solo un código de prueba, por lo tanto, quítalo una vez que estés seguro de que funciona.

10.8 Tarea 3: Recibe un disparo

Descripción: Cuando recibas un disparo por radio de tu oponente, se comprobará si tienes un barco en el (número de columna, número de fila) del disparo. Si tienes un barco allí, entonces has sido golpeado y hundido: enviarás un mensaje de "*Tocado*" a tu oponente y quitarás el barco de la pantalla. Si tu oponente falla, enviarás un mensaje de "*Agua*".

Instrucción: Dependiendo de cómo se formateó el paquete, decodifica (*número_ columna*, *número_fila*) del paquete recibido. Si tienes un barco encendido (*número_ columna*, *número_fila*), es un *Tocado*: apaga el LED en esa posición. Si tienes una estructura de datos separada como variable para representar tus naves, actualízala también. Envíale un mensaje de "*Tocado*" a tu oponente. Si se trata de una falla, envía un mensaje de "*Agua*" a tu oponente.

10.9 Tarea 4: Recibe el resultado del disparo: "*Tocado*" o "*Agua*"

Descripción: Enciende los LED en la fila superior según el resultado. Si es un "*Tocado*", verifica si alcanzas 5 Tocados. Entonces ganaste: ¡muestra una sonrisa!

Instrucción: Si recibes un "*Tocado*", enciende el LED izquierdo de la fila superior (el LED en la posición (0,0)).

Actualiza el recuento de tus éxitos, y si llegaste a 5, ¡muestra una sonrisa! Si el resultado fue "*Agua*", enciende el LED derecho de la fila superior (el LED en la posición (4,0)).

Pon a prueba tu programa(s) con tu oponente. Para empezar, será más fácil si puedes ver las pantallas de los demás. Es posible que te resulte útil ingresar algún código de prueba, como sucedió en la tarea anterior. Por ejemplo, puedes enviar "*Tocado*" o "*Agua*" cuando recibes y decodificas un disparo. Puede que incluso te resulte útil imprimir las coordenadas del disparo cuando recibas el paquete.

10.10 Actividad extensiva

El juego Batalla Naval tiene muchas variaciones. Consulte el sitio de Wikipedia en Recursos para leer sobre las variaciones.

Ejercicio 1.
Una variación del juego permite a los jugadores mantener en secreto que un barco ha sido hundido. Por lo tanto, tu oponente tiene que tomar más tiros para confirmar que toda el área está despejada. ¡Esto es un poco como perder un paquete! Recuerda cómo lidiaste con las pérdidas de paquetes en los Capítulos 8 y 9. ¿Cómo aplicarías esos conceptos a este caso? Discute las posibles soluciones con tu amigo. Luego, programa y prueba tu nueva solución.

Ejercicio 2.
Imagina una variante cuando se necesitan 3 golpes para hundir un barco en lugar de 1 golpe. ¿Cómo cambiaría tu programa? ¿Necesitas hacer cambios en el lado del emisor o del receptor? ¿Qué tan similar es esto a usar retransmisiones predeterminadas en el Capítulo 8?

10.11 Problemas

Problema 10.1 La figura 10.6 muestra naves colocadas al azar en un área de batalla. ¿Qué coordenadas necesitas enviar para golpear a todos los barcos?

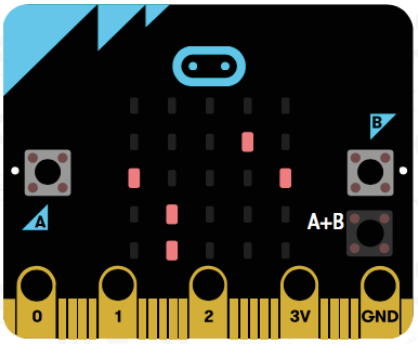


Figura 10.6: Juego Batalla Naval: un área de batalla aleatoria

Problema 10.2 La figura 10.7 muestra naves colocadas al azar en las áreas de batalla de dos micro:bits.

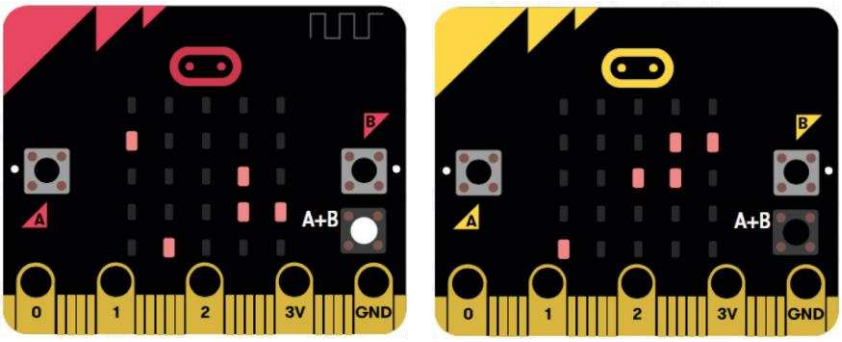


Figura 10.7: Juego Batalla Naval: Dos jugadores

La Tabla 10.1 enumera todos los disparos que se han realizado desde la micro:bit 1 (izquierda / micro:bit rojo) y la micro:bit 2 (derecha / micro:bit amarillo). ¿Quién gana?

Partida	Micro:bit 1	Micro:bit 2	Resultado
1	(3,1)	(2,1)	
2	(0,3)	(0,1)	
3	(1,1)	(3,2)	
4	(4,1)	(3,3)	
5	(0,3)	(4,3)	
6	(2,2)	(0,3)	
7	(3,2)	(1,4)	

Tabla 10.1: Disparos en cada partida

10.13 Recursos

- **Batalla Naval en Wikipedia - https://en.wikipedia.org/wiki/Battleship_(game)**
- **Batalla Naval Juego 1 Online - https://battleship-game.org**
- **Batalla Naval Juego 2 Online - http://www.mathplayground.com/battleship.html**

INDEX

A

acuse de recibo (ACK)……11, 55

B

comunicación bidireccional..... 38
binario 11, 12
bit 11, 12
byte 12
broadcast 11, 18
dirección de broadcast 18

C

medio de comunicación.....11
confiabilidad 59

D

dirección 32
dirección de broadcast 18
dirección IP 33
dirección multicast 23, 24

E

radiación electromagnética..... 18
espectro electromagnético 18
encabezado 29
frecuencia 18
hertz.....18
ondas de radio.....18
velocidad de la luz 18
longitud de onda 18
editor de bloques 49

G

comunicación grupal. 11, 23, 25

I

interferencia.....50
protocolo de internet.....33
conexión inalámbrica.....17

M

multicast 23
dirección multicast 21, 22, 25

N

network 12

P

paquete.....33
lista de permitidos.....35
ping 11, 39, 40, 41
protocolo.....33
protocolo Parar y Esperar 56, 58

R

retransmisión.....12, 50, 51
número de secuencia 57
tiempo de ida y vuelta (round-trip-time).39
tiempo de ejecución.....40

S

señal 11, 12
Solicitud de Repetición Automática (ARQ). 55, 56

T

tiempo de espera.....56, 57

U

unicast 11, 32, 44

W

wifi.....17


¡Hasta luego!