PROYECTO INTEGRADO:
"Arquitectura de Sistemas Informáticos y Scripting de Control Maqueen"

Autor: TERESA ANTON

Nivel: 1º de ASIR

BLOQUE 1: Fundamentos de Hardware
(Análisis Comparativo)

Objetivo: Identificar los componentes de una placa base real en un sistema de control embebido.

1.1. Identificación de Componentes

En esta sección, el alumno debe comparar la arquitectura de una **Placa Base ATX** con la estructura de la **Micro:bit** y la placa de expansión **Maqueen**.

- **CPU / Microprocesador:** En nuestro sistema es la placa Micro:bit, encargada de procesar las instrucciones del código.

  Buses de Datos: Los pines de conexión y el bus de radio (Grupo 0-255) que comunica el mando con el chasis.

- **Periféricos de Entrada/Salida:** Sensores infrarrojos (Entrada) y Motores/LEDs (Salida).

BLOQUE 2: Scripting y Optimización de Código

Objetivo: Desarrollar scripts de control eficientes, declarando variables y optimizando la lógica de ejecución del procesador.

2.1. Configuración Inicial (Setup del Sistema)

Para iniciar la comunicación, se deben seguir estos pasos técnicos:

1. **Carga de Librerías:** Importar la extensión "maqueen" en MakeCode.

2. **Inicialización del Bus:** Establecer el mismo grupo de radio en ambos dispositivos (Mando y Receptor) para sincronizar la frecuencia de datos.

2.2. Script de Diagnóstico POST (Power-On Self-Test)

Tarea: Modificar el inicio para que el sistema realice una comprobación visual.

- **Variable:** Crear la variable POST_status para gestionar el arranque.
- **Salida (Echo):** El sistema debe mostrar el mensaje "ASIR" **letra por letra** mediante un bucle de salida secuencial.
- **Modificación de Luces:** Tras el "echo" de las letras, encender los LEDs RGB frontales en color **Azul** para confirmar que el hardware está operativo.

2.3. Lógica de Control Eficiente (Anidamiento)

Siguiendo el principio de eficiencia en programación, se deben usar **condicionales anidados**.

- **Optimización:** Al meter los condicionales uno dentro de otro ("si no, si"), el procesador deja de leer el resto de líneas una vez que encuentra la instrucción correcta, ganando velocidad de respuesta.

2.4. Modificación de Funciones (Desafío del Alumno)

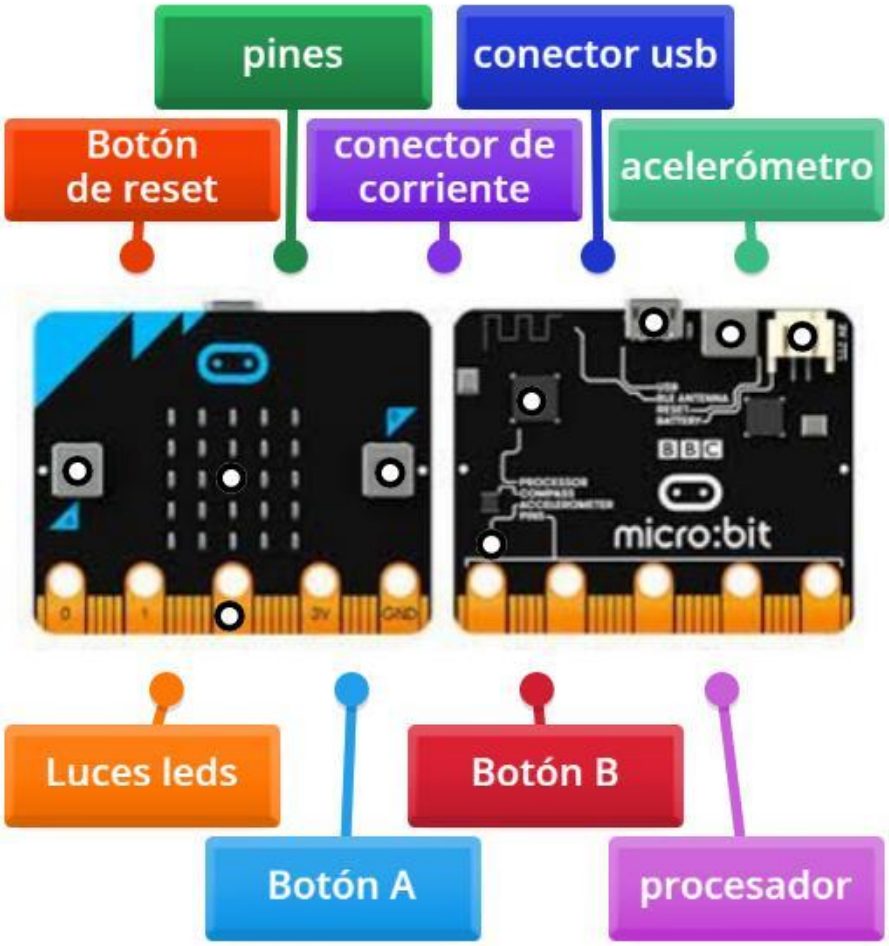
Añadir las siguientes funcionalidades al código base del Maqueen:

- **Avanzar (Instrucción 1):** Motores adelante + Encender LEDs frontales en **Rojo**.
- **Giro (Instrucción 2/3):** Parpadeo del LED correspondiente (intermitente) mientras gira.
- **Parada (Instrucción 4):** Motores stop + Mostrar imagen de **"Calavera" o "X"** en la matriz de LEDs.
- **Retroceder (Instrucción 5):** Motores atrás + Emisión de señal sonora de advertencia.

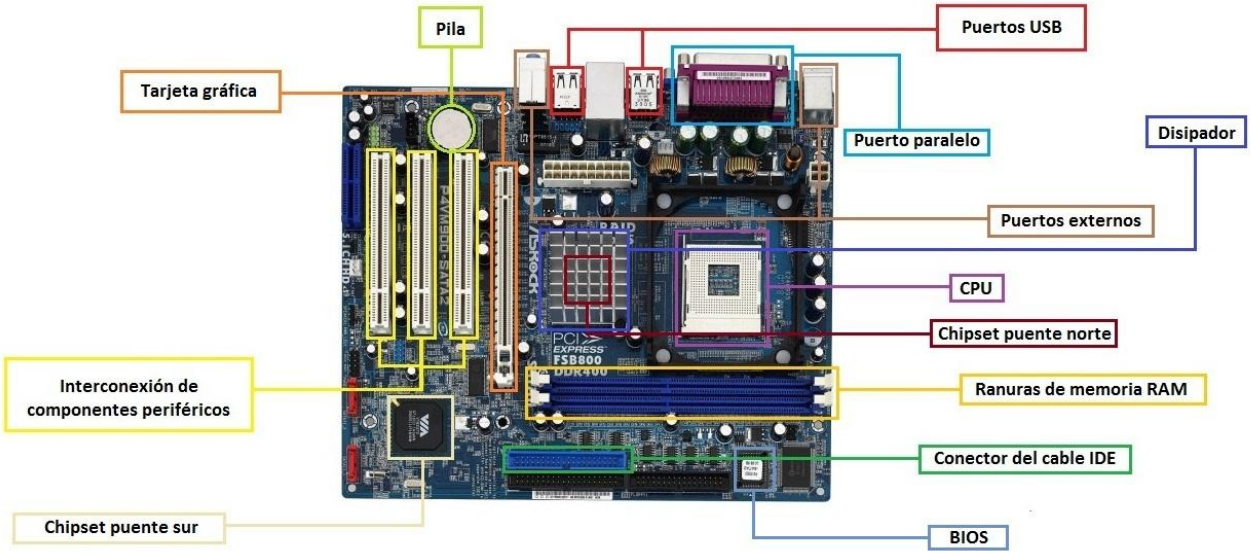
3.- Rúbrica de Evaluación (Escala 1 a 10)

Bloque	Criterio de Evaluación	Puntuación
HARDWARE	Identificación correcta de CPU, Buses e I/O en la placa base y Micro:bit.	/ 10
VARIABLES	Uso de variables para controlar el diagnóstico POST (Letra por Letra).	/ 10
LÓGICA	Implementación de condicionales anidados para optimizar el ciclo del procesador.	/ 10
SCRIPTING	Funcionalidad completa de los 5 movimientos con feedback de LEDs e imágenes.	/ 10

4.- Imágenes para realizar la actividad de Fundamentos de hardware:



COMPONENTES DE LA PLACA BASE



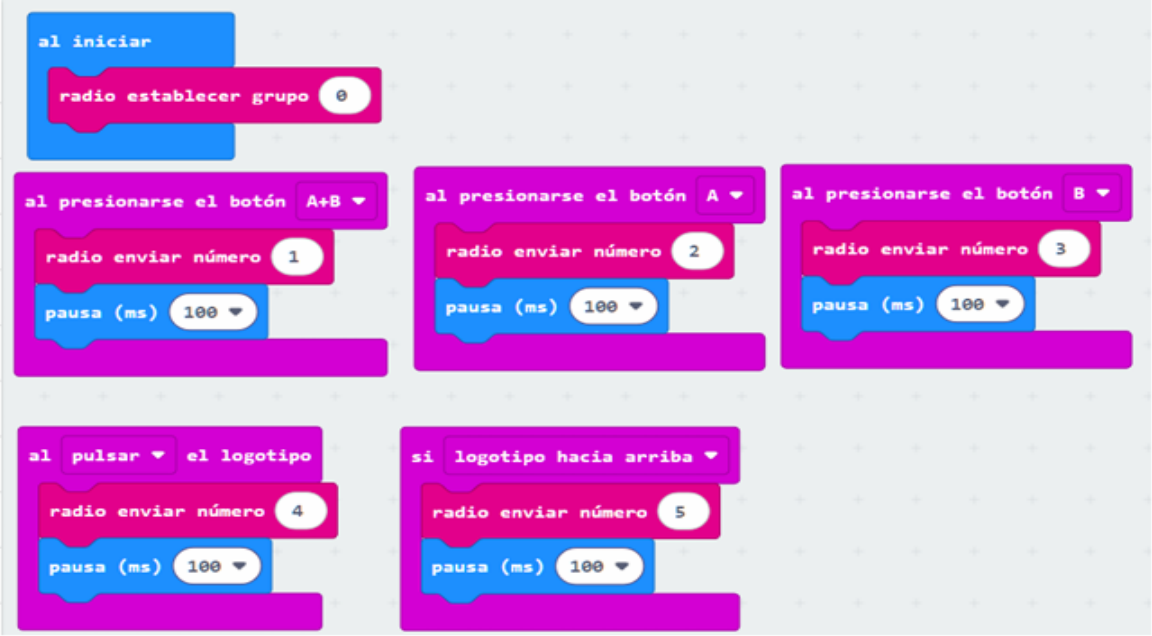
5. Ejemplo de programación de micro bit se adjunta material para poder programar, partimos del código original y se hace demostración:

1. RETO: MAQUEEN CONTROLANDO SUS MOVIMIENTOS CON OTRA MICROBIT COMO MANDO CON 5 BOTONES ACTIVOS.

Programa básico para dar movimiento a Maqueen con otra microbit como mando.
Todos los bloques relacionados con la radio se encuentran en Radio.
Al iniciar establezco a la radio y al mando el mismo grupo, por ejemplo el 0, (pero hay del 0 a 255 para elegi
adjudicar a cada alumno)
Por otro lado adjudicar un **número** a cada botón de la microbit (la emisora de radio solo emite números) y lo que queremos que haga al presionar ese botón o botones:

- **1** (botón A+B) Maqueen avanza
- **2** (botón A) Maqueen giro izquierda
- **3** (botón B) Maqueen giro derecha
- **4** (botón logo) Maqueen para
- **5** (botón logotipo hacia arriba) Maqueen retrocede o va hacia atrás.

PROGRAMA MANDO: Los bloques "al presionar el botón ....." se encuentran en Entrada
Nota: aunque es un mando con pulsadores, **no hará falta mantener presionado los mismos** para que Maqueen siga haciendo esa acción, no cambiará esta última hasta que se presione otro pulsador programad
del mando.

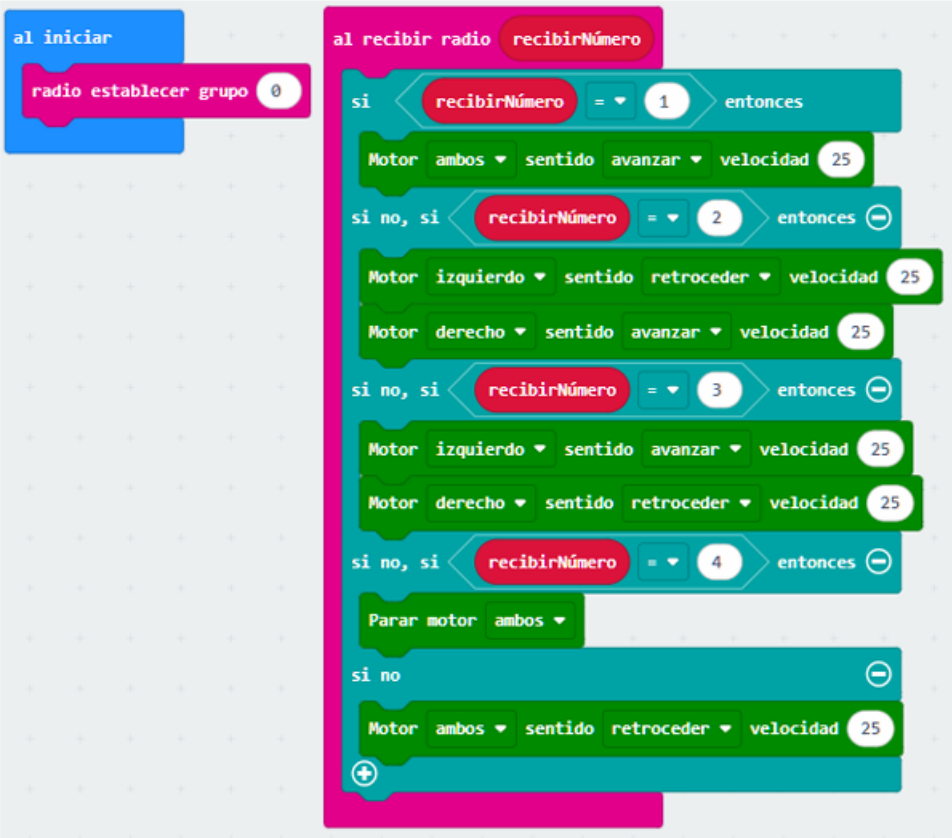


Nota: aunque el programa funciona igual, con o sin la pausa, se puede poner una corta (100 ms) para no saturar la comunicación por radio si vemos que Maqueen no hace caso a nuestras instrucciones.

6. Ejemplo de programación de micro bit se adjunta material para poder programar, partimos del código original y se hace demostración:

PROGRAMA ROBOT: con los bloques de lógica, "si" recibo un 1, el robot avanza, "si no, si" recibo un 2 el robot gira a la izquierda y así sucesivamente. Hasta que el robot no reciba un número diferente del mando, no cambiará su movimiento.

- **1** (botón A+B)
- **2** (botón A)
- **3** (botón B)
- **4** (botón logo)
- **5** (logotipo hacia arriba)



Nota: si va muy rápido puedes cambiar la velocidad a 20 o inferior para tener mayor control sobre el robot, también puedes modificar el retroceso según te convenga.

CONCLUSIÓN:

El Valor de la Tecnología Aplicada en el Aula de ASIR

La realización de esta actividad integrada no solo busca el manejo de un robot, sino la consolidación de competencias críticas en la administración de sistemas informáticos. Al trabajar con la Micro:bit y el ecosistema Maqueen, se consiguen los siguientes hitos tecnológicos:

1. Abstracción del Hardware Real

A diferencia de estudiar una placa base de forma estática, este proyecto permite al alumno ver el **flujo de datos en tiempo real**. Entender que la Micro:bit es la CPU y el chasis Maqueen es el bus de expansión ayuda a desmitificar el funcionamiento interno de un ordenador. Se comprende que el hardware es un conjunto de componentes interconectados que necesitan un firmware (nuestro código) para cobrar vida.

2. Desarrollo del Pensamiento Computacional y Lógica de Scripting

El uso de variables para procesos de diagnóstico y la implementación de **condicionales anidados** introduce al alumno en la **optimización de recursos**. En el mundo laboral de ASIR, un script mal estructurado puede consumir ciclos innecesarios de CPU en un servidor; aquí, el alumno experimenta esa eficiencia al ver una respuesta más rápida del robot cuando el código está correctamente "anidado" y optimizado.

3. Simulación de Entornos de Red y Comunicación

El trabajo con grupos de radio simula la configuración de redes inalámbricas y la gestión de canales. Los alumnos aprenden la importancia de la **segmentación y el direccionamiento**: si dos equipos usan el mismo canal sin permiso, hay colisiones de datos; si usan canales distintos, la comunicación es privada y eficiente.

4. Feedback Inmediato y Diagnóstico de Sistemas

La incorporación de LEDs frontales y mensajes en la matriz (el proceso POST) enseña una lección vital en sistemas: **la importancia de los registros (logs) y los estados visuales**. Un buen administrador de sistemas siempre programa salidas de información para saber qué está haciendo la máquina en cada momento.

En definitiva, esta actividad transforma conceptos abstractos de arquitectura de computadores y lógica de programación en una experiencia tangible. Desarrolla la capacidad de resolución de problemas, la atención al detalle técnico y prepara al estudiante para enfrentarse a entornos tecnológicos reales donde el hardware y el software deben trabajar en perfecta sincronía