

## Evento evaluativo 1

1)

Propósito específico: los sistemas embebidos se diseñan para cumplir funciones específicas a diferencias de los computacionales generales

EJEMPLO: un pacemaker, diseñado para mantener el ritmo cardiaco regular

Restricción de recursos: los sistemas embebidos operan con limitaciones de memoria, procesamiento y energía

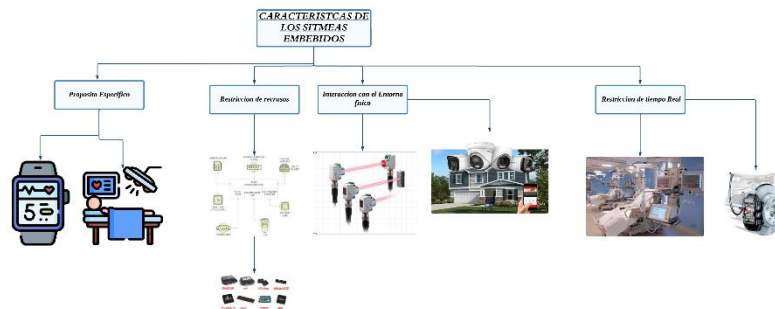
EJEMPLO: un sensor de temperatura IoT, que debe ser eficiente con la energía y capacidad de temperatura para que no sufra alteraciones en las mediciones

Interacción con el entorno físico: los sistemas embebidos interactúan directamente con un entorno a través de sensores y actuadores en tiempo real.

EJEMPLO: un sistema de control de dispensadores y control de cultivos que ajusta la frecuencia en tempo real según condiciones climáticas

Restricción de tiempo real: algunos sistemas deben operar en un marco de tiempo específico para ser efectivos

EJEMPLO: un sistema de frenado antibloqueo (ABS) en vehículos



2) Diferencias principales entre un sistema embebido con RTOS y uno sin RTOS:

Con RTOS: permite la multitarea y la administración de prioridades de manera mas eficiente ,crucial para aplicaciones que requieren respuestas en tiempo real

Sin RTOS: es mas sencillo y menos capaz de manejar múltiples operaciones o tareas simultaneas

EJEMPL BIOMEDICO:

Un ventilador mecánico en una unidad de cuidado intensivos donde el uso de RTOS es obligatorio para gestionar flujos de aire en tiempo real de manera precisa ,dado que a cualquier retraso o error puede ser mortal

3)

Gestión de la energía en sistemas embebidos:

Estrategias de hardware: uso de microcontroladores de bajo consumo, optimización del diseño de circuitos para reducir el desperdicio de energía

Estrategias de software: los algoritmos de gestión de software que afectan el rendimiento según la necesidad, como técnicas de suspensión o hibernación,

EJEMPLO CONCRETO: un dispositivo de monitoreo de salud que utiliza sensores para rastrear signos vitales y utiliza algoritmos para entrar en modo de bajo

4)

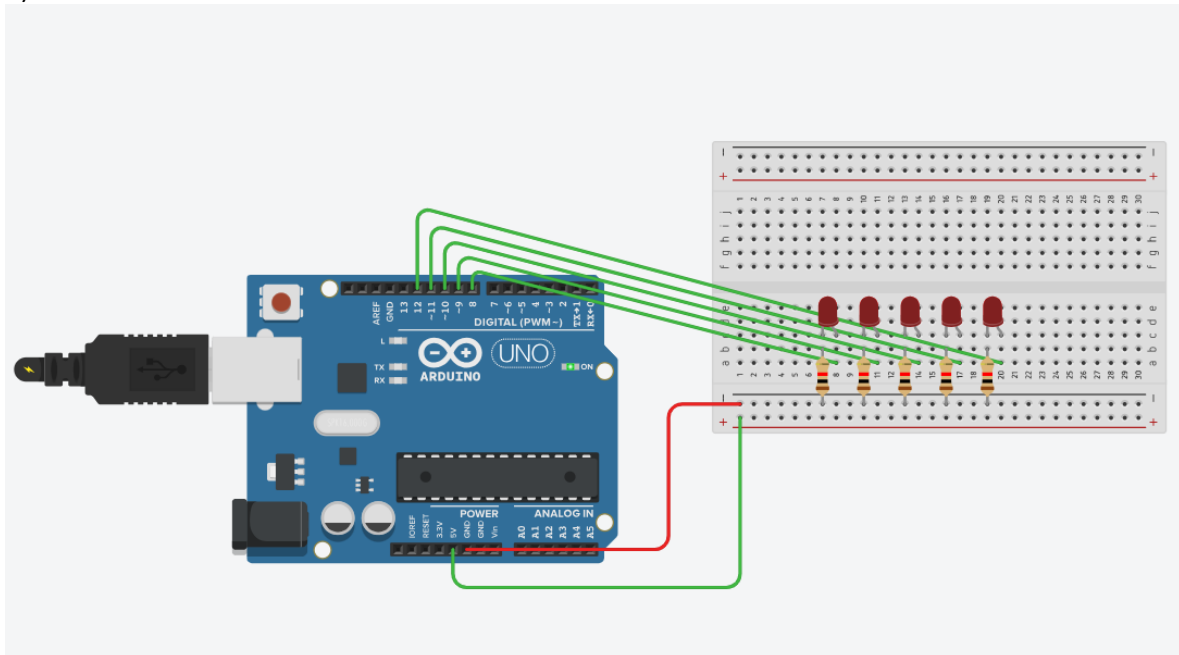
Sistemas embebidos críticos vs no críticos:

Críticos su fallo puede resultar en una grave consecuencia como pérdida de vidas o daño significativo como ejemplo un sistema de control aeronáutico, dispositivos de apoyo médico implantables


No críticos: su fallo no conlleva consecuencias graves como lo sería un reproductor de música y juguetes electrónicos

IMPORTANCIA EN EL DISEÑO: los síntomas críticos requieren mayores niveles de redundancia, pruebas más rigurosas y cumplimiento de normativa


5) modelo en tinkercad



Vista del menú:

 Monitor en serie

--- MENU DE LEDS ---  
1 - Encender de Izq a Der  
2 - Encender de Der a Izq  
3 - Parpadear todos  
4 - Encender LEDs en pares  
5 - Efecto onda (ida y vuelta)  
Ingrese un numero por favor:

Env.  

Código de varios tipos de encendido de los leds:

```
const int leds[] = {8, 9, 10, 11, 12};
```

```
const int numLeds = 5;
```

```
char comando;
```

```
void setup() {
```

```
    Serial.begin(9600);
```

```
    for (int i = 0; i < numLeds; i++) {
```

```
        pinMode(leds[i], OUTPUT);
```

```
        digitalWrite(leds[i], LOW);
```

```
    }
```

```
    mostrarMenu();
```

```
}
```

```
void loop() {
```

```
    if (Serial.available() > 0) {
```

```
        comando = Serial.read();
```

```
        ejecutarComando(comando);
```

```
    }
```

```
}
```

```
void mostrarMenu() {  
    Serial.println("\n--- MENU DE LEDS ---");  
    Serial.println("1 - Encender de Izq a Der");  
    Serial.println("2 - Encender de Der a Izq");  
    Serial.println("3 - Parpadear todos");  
    Serial.println("4 - Encender LEDs en pares");  
    Serial.println("5 - Efecto onda (ida y vuelta)");  
    Serial.println("Ingrese un numero por favor:");  
}
```

```
void ejecutarComando(char comando) {  
    Serial.print("\nEjecutando secuencia: ");  
    switch (comando) {  
        case '1':  
            Serial.println("Izquierda a Derecha");  
            secuencialzqDer();  
            break;  
        case '2':  
            Serial.println("Derecha a Izquierda");  
            secuenciaDerIzq();  
            break;  
        case '3':  
            Serial.println("Parpadeo General");  
            parpadeoTotal();  
            break;  
        case '4':  
            Serial.println("Encendido en pares");
```

```

        encenderPares();

        break;
case '5':
    Serial.println("Efecto Onda");

    efectoOnda();

    break;
default:
    Serial.println("Comando no válido");

    break;
}
mostrarMenu();
}

```

```

void secuencialzqDer() {
    for (int i = 0; i < numLeds; i++) {
        digitalWrite(leds[i], HIGH);

        delay(200);

        digitalWrite(leds[i], LOW);
    }
}

```

```

void secuenciaDerlzq() {
    for (int i = numLeds - 1; i >= 0; i--) {
        digitalWrite(leds[i], HIGH);

        delay(200);

        digitalWrite(leds[i], LOW);
    }
}

```

```
void parpadeoTotal() {  
    for (int i = 0; i < 5; i++) {  
        for (int j = 0; j < numLeds; j++) {  
            digitalWrite(leds[j], HIGH);  
        }  
        delay(300);  
        for (int j = 0; j < numLeds; j++) {  
            digitalWrite(leds[j], LOW);  
        }  
        delay(300);  
    }  
}
```

```
void encenderPares() {  
    for (int i = 0; i < numLeds; i += 2) {  
        digitalWrite(leds[i], HIGH);  
        if (i + 1 < numLeds) digitalWrite(leds[i + 1], HIGH);  
        delay(300);  
        digitalWrite(leds[i], LOW);  
        if (i + 1 < numLeds) digitalWrite(leds[i + 1], LOW);  
    }  
}
```

```
void efectoOnda() {  
    for (int i = 0; i < numLeds; i++) {  
        digitalWrite(leds[i], HIGH);  
        delay(200);  
        digitalWrite(leds[i], LOW);  
    }  
}
```

```

for (int i = numLeds - 2; i > 0; i--) {

    digitalWrite(leds[i], HIGH);

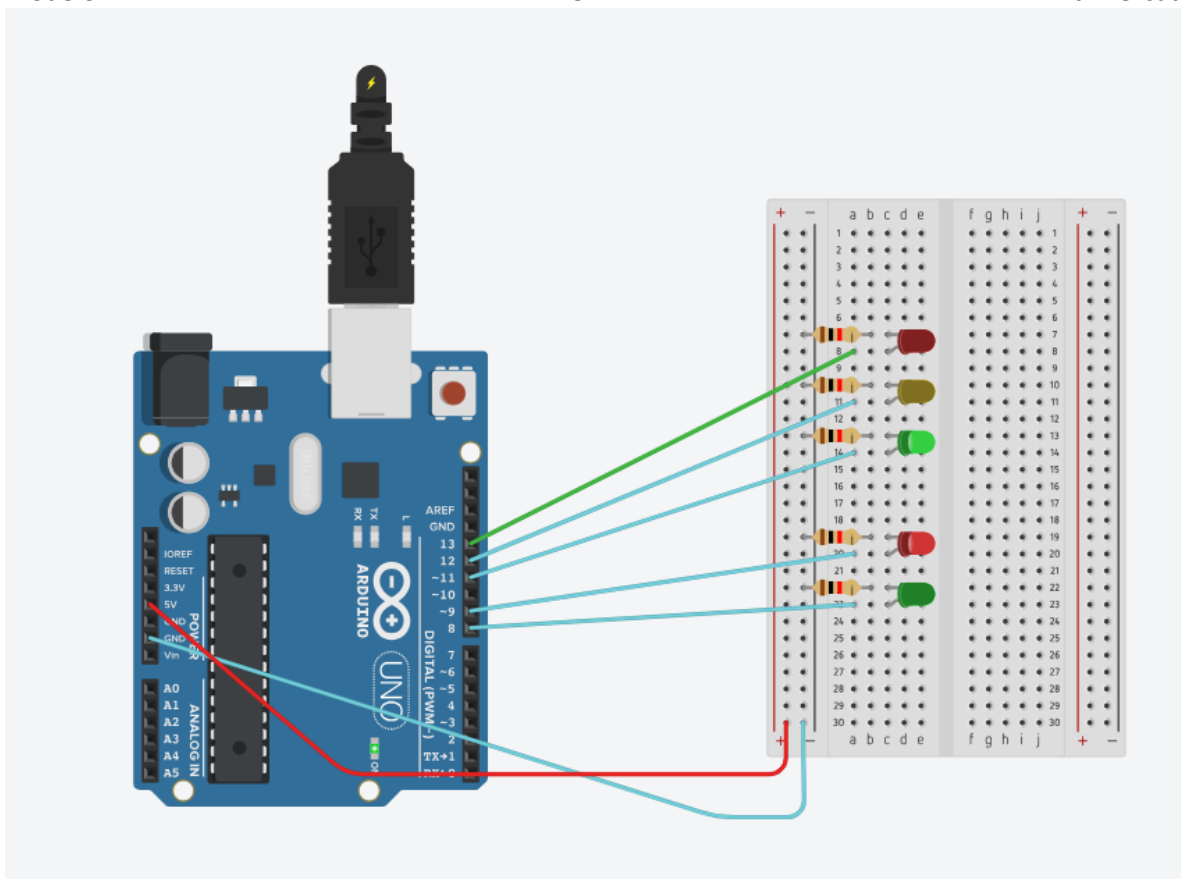
    delay(200);

    digitalWrite(leds[i], LOW);

}
}

```

6) modelo en tinkercad



**Este es el código del semáforo:**

```

const int ledRojoVehicular = 13; // Rojo vehicular
const int ledAmarilloVehicular = 12; // Amarillo vehicular
const int ledVerdeVehicular = 11; // Verde vehicular
const int ledRojoPeatonal = 9; // Rojo peatonal
const int ledVerdePeatonal = 8; // Verde peatonal

```

```

void setup() {
  pinMode(ledRojoVehicular, OUTPUT);
  pinMode(ledAmarilloVehicular, OUTPUT);
  pinMode(ledVerdeVehicular, OUTPUT);

  pinMode(ledRojoPeatonal, OUTPUT);
  pinMode(ledVerdePeatonal, OUTPUT);
}

void loop() {
  digitalWrite(ledVerdeVehicular, HIGH); // Verde v
  digitalWrite(ledRojoVehicular, LOW); // Apagar el rojo v
  digitalWrite(ledAmarilloVehicular, LOW); // Apagar el amarillo v
  digitalWrite(ledRojoPeatonal, HIGH); // Rojo peatonal
  digitalWrite(ledVerdePeatonal, LOW); // Apagar el verde peatonal
  delay(5000);

  digitalWrite(ledVerdeVehicular, LOW); // Apagar el verde v
  digitalWrite(ledAmarilloVehicular, HIGH); // Amarillo v
  delay(2000);

  digitalWrite(ledAmarilloVehicular, LOW); // Apagar el amarillo v
  digitalWrite(ledRojoVehicular, HIGH); // Rojo v
  digitalWrite(ledVerdePeatonal, HIGH); // Verde peatonal
  digitalWrite(ledRojoPeatonal, LOW); // Apagar el rojo peatonal
  delay(5000);

  digitalWrite(ledRojoVehicular, HIGH); // Rojo v
  digitalWrite(ledVerdePeatonal, LOW); // Apagar el verde peatonal
  digitalWrite(ledRojoPeatonal, HIGH); // Rojo peatonal

```



```
    delay(1000);  
}
```