



dSnake

Sistemas Empotrados Distribuidos

Máster en Ingeniería Informática



Jorge Casas Hernán
Mariano Hernández García

índice de contenidos



- Introducción
- Objetivos del proyecto
- Arquitectura del sistema
- Diseño
- Implementación
- Conclusiones
- Trabajo Futuro



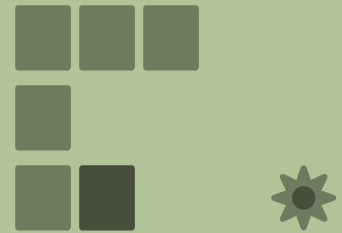
Introducción

Introducción

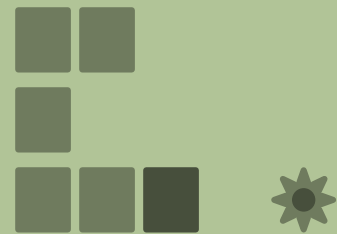


- dSnake (Distributed Snake) consiste en el clásico videojuego de los 70 *Snake* modificado para dos jugadores.
- Cada jugador controlará a una serpiente.
- Su objetivo será eliminar a la serpiente del otro jugador.
- Las serpientes morirán al colisionar con un muro, con la serpiente del otro jugador o con su propio cuerpo.

Introducción

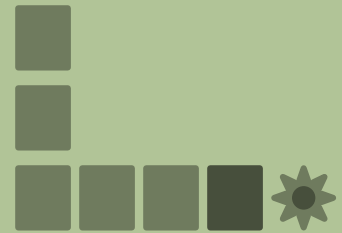


- Cada vez que un jugador gana, se le suma un punto. Máximo de 15 (F)
- Aparecerán frutas de forma aleatoria
- Las serpientes crecerán de tamaño cuando se coman una fruta

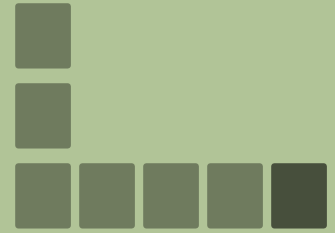


Objetivos del proyecto

Objetivos del proyecto

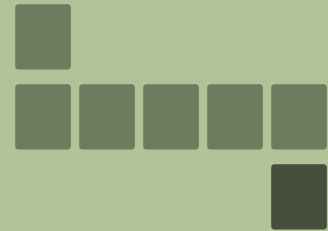


- Implementar el videojuego clásico *Snake*, adaptado para 2 jugadores
- Utilizar, al menos, dos placas de desarrollo diferentes
- Crear una memoria completa que documente el proyecto realizado



Arquitectura del sistema

Placas



→ Dos maletines con placa S3CEV40

- ◆ Cada jugador controlará a su serpiente y visualizará el juego

→ Raspberry Pi 2:

- ◆ Necesaria para las comunicaciones entre los maletines

Placas: Maletín



- Pulsadores: sirven para que el jugador 1 inicie la partida cuando ambos están listos.
- Teclado matricial: sirve para mover a la serpiente hacia el norte, sur, este y oeste.
- Timer 0: cuando expira, genera una fruta

Placas: Maletín



- Display LED 8 segmentos: sirve para mostrar la puntuación del jugador. Rango [0,F]
- LEDs: muestran el identificador de jugador, que puede ser '1' o '2'
- Uart1: canal de comunicación entre los maletines y la Raspberry Pi 2

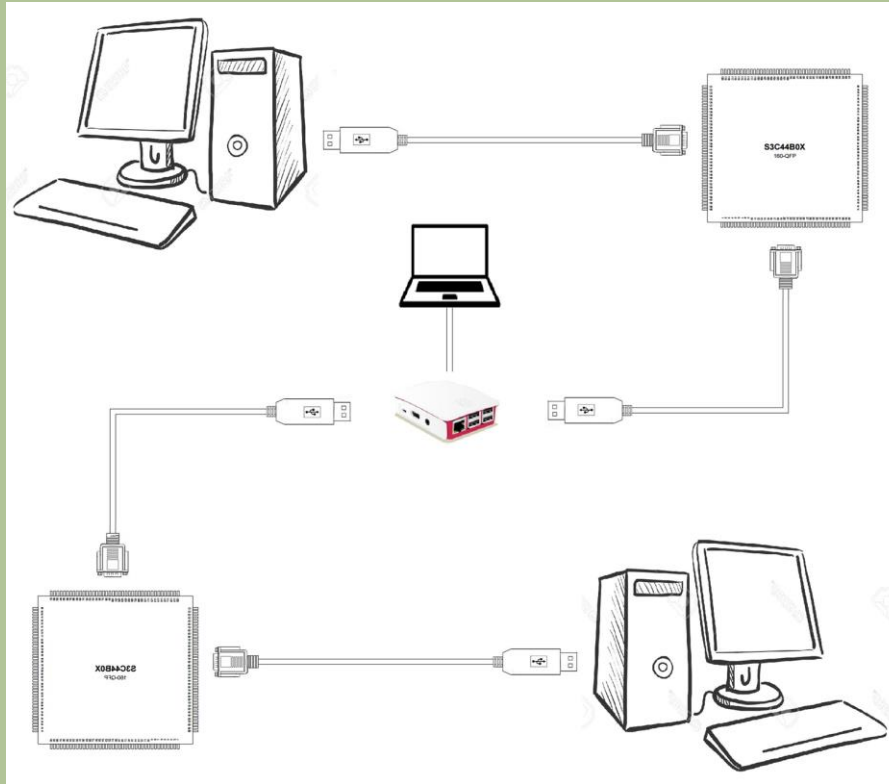
Placas: Raspberry Pi 2



→ Puerto Ethernet: canal de comunicación entre la rasp y el ordenador

→ Puertos USB(2): canal de comunicación entre la rasp y cada uno de los maletines

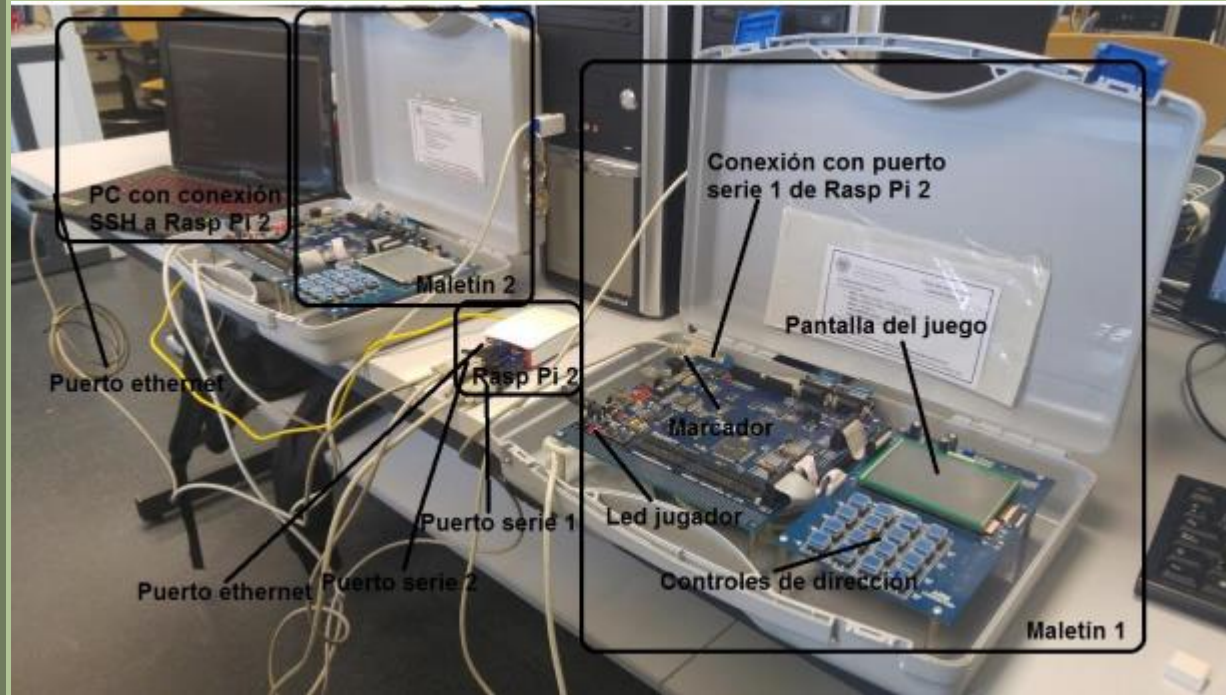
Interconexión entre componentes



Conexiones:

- 2x Adaptador SERIE - USB
- 2x Cable SERIE
- 1x Cable Ethernet

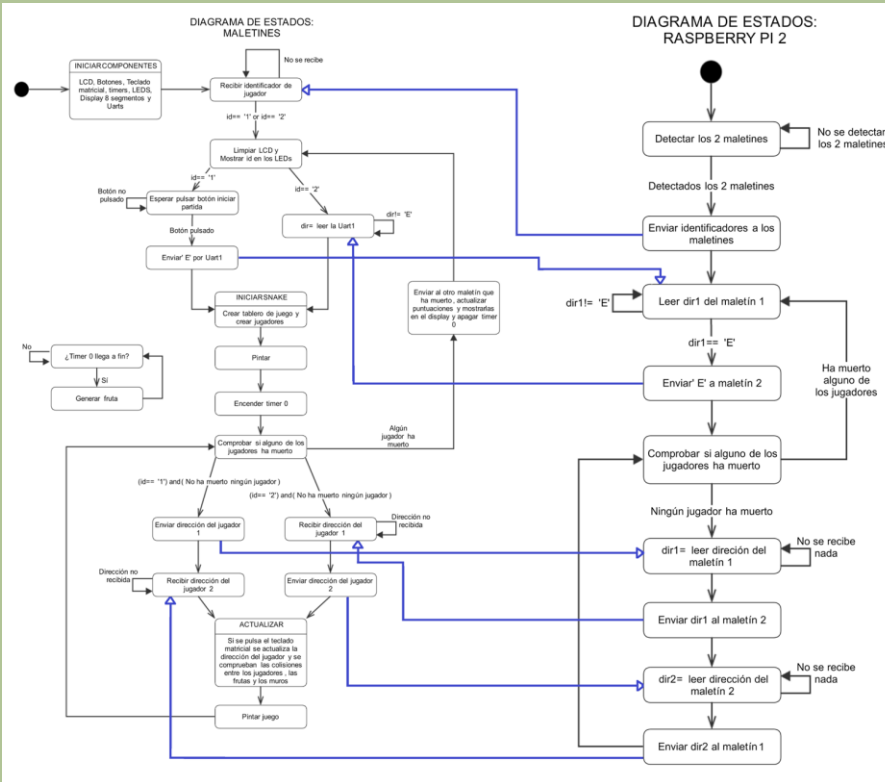
Interconexión entre componentes



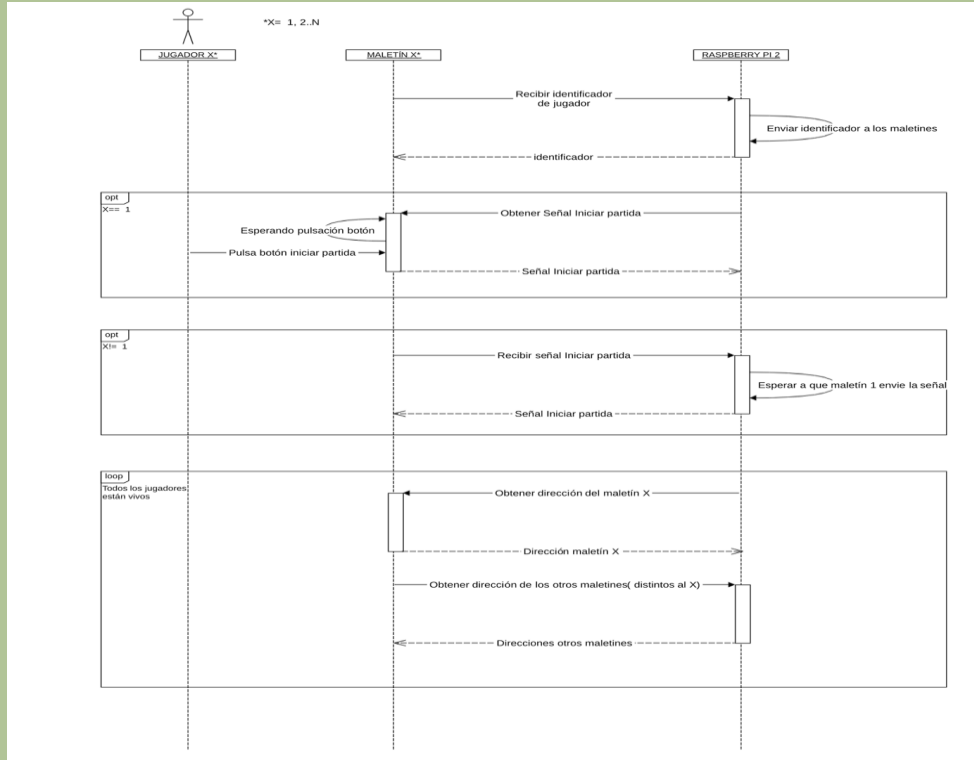


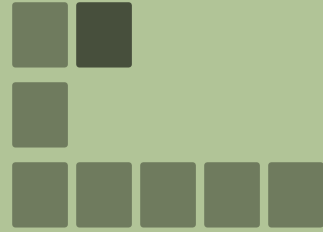
Diseño

Diseño: CFSM



Diseño: Diagrama de secuencia





Implementación

Primer prototipo



- Utiliza 2 maletines conectados por la Uart1
- El tiempo entre un fotograma y el siguiente es de aproximadamente 1 segundo:
 - ◆ 170ns → Comunicaciones
 - ◆ 800ms → Pintado de un fotograma en la LCD

Primer prototipo



→ Ejecutan el mismo código C (con directivas) con la misma máquina de estados:



- ◆ Iniciar tablero de juego
- ◆ Enviar posiciones del jugador al otro maletín
- ◆ Obtener posiciones del jugador del otro maletín
- ◆ Actualizar jugadores
- ◆ Generar fruta
- ◆ Calcular colisiones



Segundo prototipo



- Utiliza dos maletines conectados por la Uart1
- Se ha optimizado el código para disminuir el tiempo (segundos -> milisegundos):
 - ◆ Ya no se envían las posiciones de cada serpiente, sino la DIRECCIÓN que han tomado (1 byte).
 - ◆ En cada ciclo no se pinta toda la pantalla, sino que se actualizan solo las posiciones que HAN CAMBIADO

Segundo prototipo



→ Ejecutan el mismo código C (con directivas) con la misma máquina de estados:



- ◆ Iniciar tablero de juego

- ◆ Pintar juego

- ◆ Enviar DIRECCIÓN del jugador al otro maletín

- ◆ Obtener DIRECCIÓN del otro maletín

- ◆ Actualizar jugadores

- ◆ Generar fruta

- ◆ Calcular colisiones

- ◆ Pintar las posiciones que han cambiado




Tercer prototipo



- Utiliza dos maletines y una Raspberry Pi 2
- Cada maletín se conecta por serie por la Uart1 a uno de los puertos USB de la Raspberry.
- La latencia y los fallos de comunicación se reducen drásticamente.

Tercer prototipo



- La Raspberry ejecuta un programa en *Python* y actúa como un switch simplificado: recibe los paquetes de un maletín y los reenvía al otro.
- Ahora los maletines no usan directivas de compilador, la Raspberry gestiona a los jugadores 
- Este prototipo facilita la futura escalabilidad del sistema → N maletines (N jugadores)

Tercer prototipo

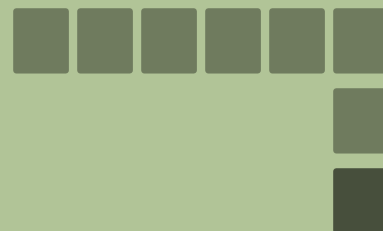


→ La misma máquina de estados queda así:

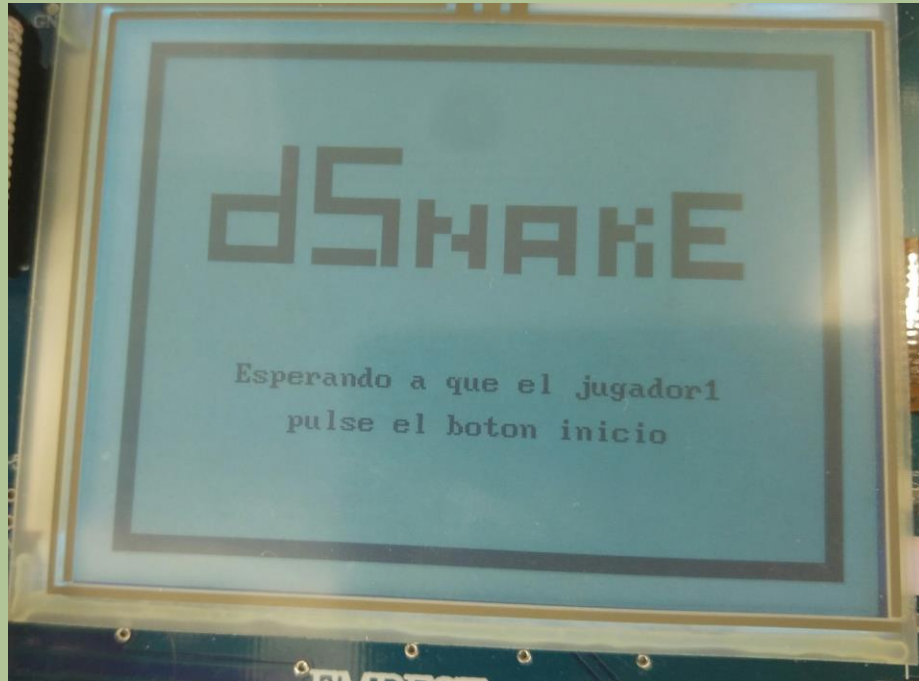
- ◆ Obtener identificador de jugador desde la Rasp
- ◆ Iniciar tablero de juego
- ◆ Pintar juego
- ◆ Enviar DIRECCIÓN del jugador al otro maletín
- ◆ Obtener DIRECCIÓN del otro maletín
- ◆ Actualizar jugadores
- ◆ Generar fruta
- ◆ Calcular colisiones
- ◆ Pintar las posiciones que han cambiado



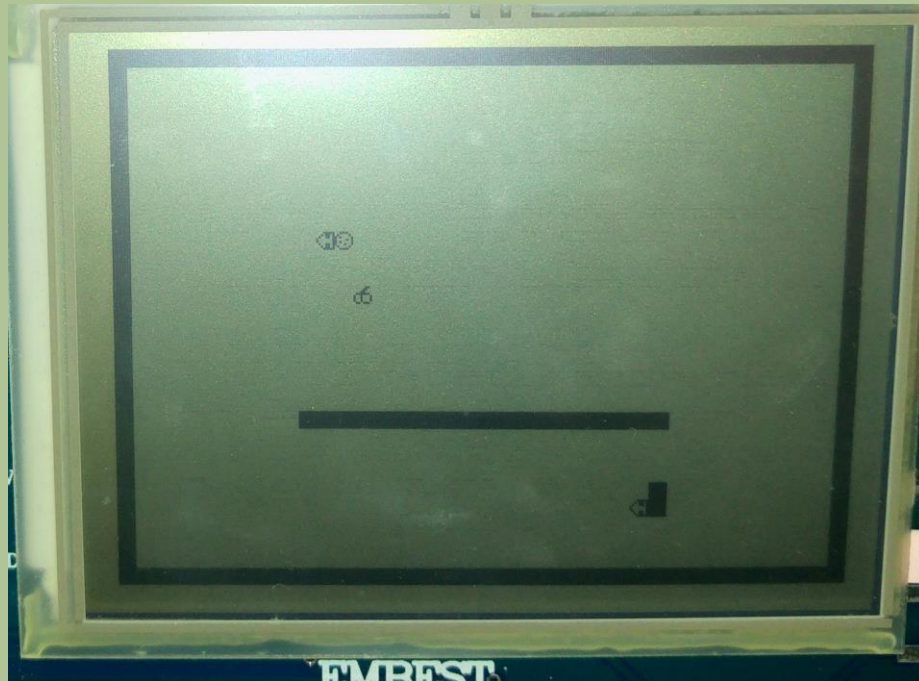
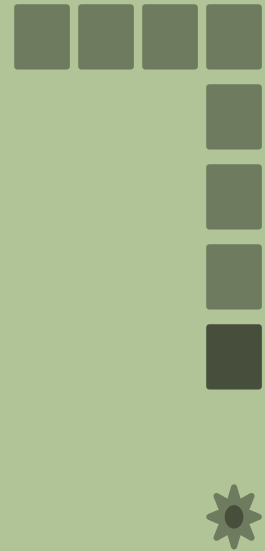
Capturas



Capturas



Capturas



Conclusiones



Conclusiones

- Implementar un videojuego en tiempo real que se ejecute de forma eficiente sobre un hardware de bajo rendimiento requiere dedicar gran parte del esfuerzo a la optimización.
- La comunicación directa entre maletines a través de la UART es inestable.
- Es muy importante optimizar las comunicaciones. Son el mayor cuello de botella.

Conclusiones

- El uso de la Raspberry Pi 2 nos ha hecho ver que resulta mucho más fácil implementar un sistema empotrado en hardware actual.
- Los 3 prototipos que hemos desarrollado han sido optimizados todo lo que hemos podido. Hay que exprimir el hardware al máximo

Trabajo futuro



Trabajo futuro

→ Aumentar la escalabilidad del sistema para que puedan jugar N jugadores:

- ◆ Modificar el código *Python* de la Raspi
- ◆ Modificar el código C de los maletines

Muchas gracias!!

