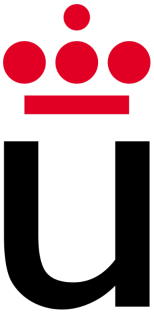

Universidad
Rey Juan Carlos

UNIVERSIDAD REY JUAN CARLOS
GRADO EN INGENIERÍA INFORMÁTICA
INTERACCIÓN PERSONA-ORDENADOR

PROYECTO - MINI CONSOLA INALÁMBRICA

GRUPO 16

Daniel Rodríguez Fernández
Víctor Fernández Montaño
Víctor Montoya Montalvo

1. Introducción y motivación

Para esta práctica se ha buscado aplicar de forma práctica los conocimientos adquiridos en el desarrollo de sistemas empotrados, comunicaciones inalámbricas y programación de bajo nivel, integrándolos en un proyecto interactivo y funcional.

De esta idea nace una mini consola inalámbrica capaz de ejecutar el clásico juego Pong, permitiendo la interacción entre jugadores sin necesidad de conexiones físicas. El proyecto no solo persigue la implementación básica del juego, sino también la exploración de las capacidades de comunicación entre dispositivos, la optimización de recursos limitados y el diseño de una experiencia de usuario sencilla. El objetivo final ha sido desarrollar un sistema compacto, portátil y funcional que refleje tanto los conocimientos previos como los adquiridos durante la asignatura.

2. Descripción y funcionamiento

La mini consola inalámbrica es un sistema portátil que está desarrollado en Arduino el cual integra una pantalla para ver el juego, mostrando ambas paletas, la bola, y las puntuaciones. También incluye dos mandos con acelerómetros conectados a la red wifi para la transmisión de información. La consola al encenderla se pone en la pantalla de inicio del Pong lista para empezar. El sistema es el encargado de la gestión de la lógica de cada juego, leyendo las entradas (acelerómetro a través de la red wifi implementada).

2.1 Pasos realizados

1. **Brainstorming de ideas:** se barajaron distintas ideas de proyecto hasta que acordamos hacer esta práctica, se la presentamos al profesor y la aceptó.

2. **Montamos un prototipo de la comunicación wifi (software):** probamos el funcionamiento de la red creada.

3. **Añadida la pantalla e implementación del juego:** se desarrolló el código para el funcionamiento del juego y se conectó la pantalla al proyecto.

4. **Montamos un prototipo del mando:** probamos que el mando funcionara e hiciera las funciones básicas.

5. **Terminamos el mando:** soldamos el mando y acabamos de montar el hardware.

6. **Playtesting y ajuste de sensibilidad y dificultad:** se terminó de calibrar la comodidad del gameplay y los distintos parámetros que afectan a la experiencia jugable.

2.2 Reparto de tareas

Daniel Rodríguez Fernández: Brainstorming, parte del ensamblaje, parte de las pruebas, escritura de la memoria y blog.

Víctor Montoya Montalvo: Brainstorming, parte del ensamblaje, parte de las pruebas, presentación en PowerPoint, escritura de la memoria y blog.

Víctor Fernández Montaño: Brainstorming, parte del ensamblaje, desarrollo del software, parte de las pruebas, compra del material.

3. Materiales Requeridos y coste

Para la realización de esta práctica, además de los materiales con los que ya contábamos, hemos requerido comprar algunos extra que poseen su precio adjunto:

2x ESP32-S3-Zero: CPU dentro de los mandos que se conecta a la conexión wifi que levanta la pantalla. (7,14€)

Módulo GY-521/MPU-6050: sensor de movimiento con acelerómetro y giroscopio dentro de los mandos para mover la pala. (4,42€)

Placa de desarrollo ESP32 Arduino LVGL WIFI y Bluetooth 2,8 "240*320 pantalla inteligente módulo TFT LCD de 2,8 pulgadas con WROOM táctil. Pantalla que contiene la lógica del juego, muestra los gráficos, y levanta la red WiFi (12,66€)

Batería de tipo IHR-18650: Da energía a los mandos. (Ya lo teníamos)

Pulsador de panel normalmente abierto: Botón para ciertas actividades. (Ya lo teníamos)

4. Hardware

- **Mandos:** Son unos botes que contienen el Módulo GY-521 MPU-6050 MPU6050 (acelerómetro y giroscopio) que están conectados al botón de la tapa y a la ESP32-S3-Zero (CPU del mando). Esta CPU recibe los datos del acelerómetro, los convierte al tipo de datos requerido, y lo envía a la consola mediante WiFi. Ambos componentes están conectados a una batería de tipo IHR-18650.

- **Pantalla:** Está implementada usando una placa de desarrollo ESP32 Arduino LVGL WIFI y Bluetooth 2,8 "240*320 pantalla inteligente módulo TFT LCD de 2,8 pulgadas con WROOM táctil dentro de la caja de un disco duro viejo. Esta placa contiene el programa del juego, la red WiFi, y la pantalla en la que se juega. Para que funcione hay que enchufarla con un USB a una fuente de energía.

5. Software

5.1 Consola

- **Inicialización:**
  Prepara la pantalla para dibujar el juego.

  tft.init();
  tft.setRotation(1);
  tft.fillScreen(TFT_BLACK);

  Crea la red WiFi para que el mando se conecte

  WiFi.softAP(ssid, password);

  Inicia la comunicación UDP

  Udp.begin(localPort);

- **Espera:**
  Antes de empezar el juego espera a que se conecten los mandos, les asigna jugador, y espera a que pulsen el botón de listo.

```
case PACKET_INTRO_TYPE:
    S2C_AssignPacket p;
    p.type = PACKET_ASSIGN_TYPE;
    p.playerId = connectedPlayers++;
    sendPacketTo(Udp.remoteIP().toString().c_str(), &p,
sizeof(S2C_AssignPacket));
    sprintf(s, "Player %d", p.playerId+1);
    tft.drawString(s, TFT_HEIGHT/4*(p.playerId == 0 ? 1 : 2),
TFT_WIDTH/2+FONT_SIZE_PX*4, FONT);
    tft.drawString("connected!", TFT_HEIGHT/4*(p.playerId == 0 ? 1 :
2), TFT_WIDTH/2+FONT_SIZE_PX*5, FONT);
    break;
```

```
if(paddles[0].isReady) {
    gameStart = true;
```

- **Juego:**
  Recibe el input del mando para mover la pala

  ```
  currentTilt = r.data.input.tilt;
  ```

  Mueve la pala del jugador.

  ```
  paddles[playerId].y = map(tilt, ...);
  ```

  Mueve la pala del jugador 2.

  ```
  updateAI();
  ```

  Mueve la pelota.

  ```
  ball->x += ball->vx;
  ball->y += ball->vy;
  ```

  Calcula las colisiones con las paredes y las palas.

  ```
  ball->vy *= -1;
  reflectBallOffPaddle();
  ```

  Aumenta la dificultad

  ```
  if(bounceCounter >= 10)
  ```

  Calcula la puntuación

  ```
  if(ball->x <= -ball->r) {
     score2++;
     initGameState();
   }else if(ball->x >= TFT_HEIGHT+ball->r) {
     score1++;
     initGameState();
   }
  ```

- **Dibujado en pantalla:**
  Para dibujar las palas, pelota, y mensajes (puntuaciones, mensaje inicial, etc) se usan las siguientes funciones.

```
tft.fillRect(...)
tft.fillCircle(...)
tft.drawString(score)
```

5.2 Mandos

- **Inicialización:**
  Trata de conectarse al WiFi

  ```
  WiFi.begin(ssid, password);
  ```

  Si no puede conectarse comienza el modo OTA, pero si tiene éxito al conectarse inicializa los sensores, calibra el giroscopio, inicia la comunicación UDP y se presenta al servidor

  ```
  mpu.begin()
  gyroBiasZ += g.gyro.z;
  Udp.begin(localPort);
  sendIntro();
  ```

- **Modo OTA:**
  Si no puede conectarse a la consola crea su propia red WiFi y permite desde el ordenador abrirlo en el navegador y subirle código actualizado para que se reinicie y lo use.

  ```
  WebServer server(80);
  const char* uploadPage = R"rawliteral(
  <form method='POST' action='/update'
  enctype='multipart/form-data'>
    <input type='file' name='update'>
    <input type='submit' value='Update'>
  </form>
  )rawliteral";

  void loop() {
    if(enabledOTAmode) {
      server.handleClient();
      (...)
    }
    (...)
  }
  ```

- **Loop:**
  Lee las entradas del acelerómetro y del botón.

mpu.getEvent(&a, &g, &temp);
digitalRead(BTN_PIN)

Calcula la inclinación usando el algoritmo del filtro complementario

*currentAngle = ALPHA * (gyro) + (1-ALPHA) * accel;*

Crea un paquete con los datos y se los envía a la consola.

C2S_InputPacket p;
sendPacket(&p, sizeof(...));

6. Interacciones

- **Inicio:** al iniciar el juego sale la pantalla de inicio y una cuenta atrás para el comienzo de la partida.

- **Juego:** la pala del jugador se mueve al mover el mando, y cada 10 rebotes la velocidad aumenta para aumentar la dificultad.

- **Apagado:** para apagarlo hay que desenchufar la pantalla y los mandos.

7. Problemas encontrados y soluciones

- **Se nos ha quemado un acelerómetro a dos días de la entrega.** Solución: se ha desarrollado una IA para el jugador 2, volviendolo un juego individual en caso de que no consiguiéramos encontrar otro acelerómetro a tiempo. Al final se ha encontrado, así que se ha aprovechado el código anterior para dar la opción de juego de 1 jugador o 2.

- **La pantalla no tiene potencia suficiente para reescribirse en cada frame.** Solución: hemos desarrollado el código para que se borre solo la pala y su posición anterior, consiguiendo optimizar el proyecto.

- **Meter los componentes del mando en el envase, que por tamaño era difícil de encajar.** Solución: reducir el tamaño del cableado y adaptar la forma del diseño a la del mando.

- **El acelerómetro mide la aceleración y el giroscopio la variación del ángulo, pero no te da el ángulo en el que estás, solo la variación.** Solución: se ha usado el algoritmo del filtro complementario para sacar el ángulo, que es una fórmula matemática que permite esto.

8. Bibliografía

Explicación sobre el filtro complementario
https://www.hibit.dev/posts/92/complementary-filter-and-relative-orientation-with-mpu6050

Otra explicación sobre el filtro complementario
https://vanhunteradams.com/Pico/ReactionWheel/Complementary_Filters.html

Ejemplos de la librería utilizada para manejar los gráficos de la pantalla
https://github.com/witnessmenow/ESP32-Cheap-Yellow-Display/tree/main/Examples/Basics