Periféricos y Dispositivos de Interfaz Humana

UNIVERSIDAD DE GRANADA

CURSO 2024 - 2025

Proyecto: PS Move como dispositivo de entrada para un ordenador

GABRIEL VICO ARBOLEDAS
RAÚL RODRÍGUEZ RODRÍGUEZ

Índice

1. Introducción

1.1 Objetivo del proyecto

El objetivo principal de este proyecto es transformar un mando de movimiento (como el PlayStation Move) en un dispositivo de entrada que simule el comportamiento de un ratón convencional en un sistema Windows. Aprovechando sus sensores inerciales (IMU), botones y conectividad Bluetooth, el sistema convierte el movimiento físico del mando en desplazamiento del cursor y gestiona los clics mediante la pulsación de botones específicos del dispositivo.

Además, el proyecto busca ser modular, fácilmente configurable y ampliable, permitiendo personalizar el comportamiento del cursor, los mapeos de botones, y la sensibilidad, así como soportar múltiples controladores si se desea.

1.2 Descripción general del sistema

Este sistema combina diferentes piezas de hardware y software para emular el comportamiento de un ratón utilizando un mando PlayStation Move. El flujo de trabajo y arquitectura general se describen a continuación:

1. Conexión y emparejamiento

Se utiliza PSMoveConfigTool para emparejar el mando con el PC a través de Bluetooth. Durante este proceso, también se realiza la calibración del giroscopio, acelerómetro y magnetómetro.

2. Transmisión de datos

Una vez emparejado, PSMoveService comienza a transmitir datos sensoriales (rotación, aceleración, etc.) y de botones a través de un servidor local.

3. Procesamiento con FreePIE

FreePIE, con un script personalizado en Python, interpreta estos datos. Se calcula el movimiento relativo del mando y se traduce a desplazamiento del cursor en pantalla. Los botones se asocian a clic izquierdo o derecho.

4. Simulación de ratón

Utilizando módulos como pywin32, el script de FreePIE puede enviar comandos al sistema operativo para mover el cursor y realizar acciones de clic.

1.3 Arquitectura del sistema

Se ha usado una arquitectura modular, que permite depurar cada componente por separado, y sustituir o extender funcionalidades sin modificar el sistema completo. También deja abierta la posibilidad de adaptar el sistema a otros mandos con sensores similares. A continuación, se muestra un esquema simplificado de la arquitectura del sistema:

```
[Mando PS Move] --> [Bluetooth] --> [PSMoveService] --> [FreePIEIO Plugin] --> [FreePIE Script (Python)] --> [Windows Input Simulation]
```

2. Requisitos para el proyecto

El correcto funcionamiento del sistema depende de una serie de requisitos tanto de software como de hardware, así como de la compatibilidad entre versiones. A continuación, se detallan todos los elementos necesarios para la implementación y ejecución exitosa del proyecto.

2.1 Software necesario

Para establecer la comunicación entre el mando de movimiento y el sistema operativo Windows, así como para interpretar los datos sensoriales y convertirlos en movimientos del ratón, se requieren las siguientes herramientas y bibliotecas:

- **PSMoveService**
  - Software que actúa como servidor para la conexión con mandos PS Move vía Bluetooth.
  - Permite la lectura en tiempo real de los sensores y botones del mando.
  - Se ejecuta como un servicio y es fundamental para el resto del sistema.

- **PSMoveConfigTool**
  - Interfaz gráfica incluida con PSMoveService.
  - Se utiliza para el emparejamiento, calibración y prueba de conexión del mando.
  - Facilita visualizar datos de sensores y comprobar que el dispositivo está correctamente emparejado.

- **FreePIE (Free Programmable Input Emulator)**
  - Entorno de scripting en Python orientado a la emulación de entradas (ratón, teclado, joystick, etc.).
  - Permite recibir datos del PSMoveService y convertirlos en eventos de entrada del sistema.
  - Requiere el plugin FreePIEIO.dll para la comunicación con el servicio.

- **Python** (versión 3.12 o compatible)
  - Necesario para el uso de scripts personalizados en FreePIE.
  - Se emplea con módulos como pywin32 para emular eventos de ratón a nivel de sistema.

- **Módulos adicionales de Python**
  - pywin32: para interactuar con la API de Windows (mover el cursor, simular clics, etc.).
  - Se instala vía pip (pip install pywin32), y puede requerir privilegios administrativos.

2.2 Hardware necesario

El sistema requiere los siguientes componentes de hardware:

- **Mando PlayStation Move**
    - Controlador principal del sistema.
    - Debe estar funcional, cargado, y tener conectividad Bluetooth.

- **Adaptador Bluetooth**
    - Compatible con HID y perfiles Bluetooth estándar.
    - Puede ser interno (portátiles) o USB externo (PCs de escritorio).
    - Recomendable que soporte múltiples conexiones simultáneas y tenga buen alcance.

- **PC con sistema operativo Windows**
    - Recomendado: Windows 7 o superior, con permisos de administrador para instalación de drivers y servicios.

2.3 Versiones compatibles

La compatibilidad de versiones es clave para evitar errores durante la instalación y ejecución. A continuación, se enumeran las versiones recomendadas y probadas:

1. **Windows**: Windows 10 (versión 1809 o superior)
2. **PSMoveService**: Versión: 1.0.0.0 Beta 9 o superior. Requiere .NET Framework 4.6 o superior.
3. **FreePIE**: Versión estable disponible en GitHub.
4. **Python**: Se ha probado con Python 3.10 y Python 3.12. Asegurarse de que se instalan los paquetes de 64 bits para compatibilidad con pywin32.
5. **pywin32**: Versión recomendada: pywin32-310 para Python 3.10 o correspondiente a tu versión instalada.

3. Instalación del proyecto

3.1 Instalación de PSMoveService

PSMoveService es el núcleo del sistema. Actúa como servidor para recibir y transmitir datos desde el mando PS Move a otros programas como FreePIE.

Pasos para su instalación:

1. **Descargar el instalador**:
    - Accede al repositorio oficial: https://github.com/psmoveservice/PSMoveService/releases .
    - Descarga la última versión estable (release ZIP o instalador .exe).

2. **Extraer o instalar**:
    - Si es un .zip, descomprime el contenido en una carpeta de fácil acceso.
    - Si es un instalador, ejecuta con permisos de administrador.

3. **Ejecutar el servicio**:
    - Inicia el archivo PSMoveService.exe.
    - Asegúrate de que la consola muestra mensajes como "Listening for connections…" para verificar que está funcionando.

4. **Configurar con PSMoveConfigTool**:
    - Ejecuta PSMoveConfigTool.exe para gestionar mandos, calibraciones y pruebas.
    - Este programa detectará automáticamente el servicio en ejecución.

3.2 Instalación de FreePIE

FreePIE es el motor que ejecuta scripts personalizados (en Python) que interpretan los datos del mando y simulan entradas del ratón.

Pasos para su instalación:

1. **Descargar FreePIE**:
    - Desde el repositorio oficial de GitHub: https://github.com/AndersMalmgren/FreePIE
    - Puedes descargar la última versión release.

2. **Instalación**:
    - Si has descargado un .zip, simplemente descomprime en cualquier directorio.
    - Asegúrate de que el ejecutable FreePIE.exe esté accesible.

3. **Instalar el plugin FreePIEIO (si no viene incluido)**:
    - Algunos scripts requieren el plugin FreePIEIO.dll para recibir datos desde PSMoveService. Copia este archivo en la carpeta plugins/ de FreePIE si no está.

4. **Ejecutar FreePIE**:
    - Ejecuta FreePIE.exe como administrador.
    - Carga el script Python que controlará el ratón (que se mostrará a continuación).

3.3 Conexión del mando por Bluetooth

La conexión Bluetooth es un paso crítico. El mando debe estar emparejado correctamente con el PC para que pueda enviar datos a PSMoveService.

Pasos para la conexión:

1. **Conectar por USB**:
    - Conecta el mando PS Move al PC con su cable USB.
    - Asegúrate de que se enciende la luz y Windows lo reconoce.

2. **Abrir PSMoveConfigTool**:
    - Selecciona la opción Pair Controller to Host.
    - El sistema emparejará el mando con tu adaptador Bluetooth. Si se desconecta y parpadea, es señal de éxito.

3. **Desconectar el cable USB**:
    - El mando debe permanecer encendido y visible en la herramienta.

4. **Verificar conexión**:
    - Dentro de PSMoveConfigTool, selecciona “Test Controller” y mueve el mando.
    - Debes ver los datos en vivo del giroscopio, acelerómetro y magnetómetro.

3.3.1 Calibración del giroscopio, acelerómetro y magnetómetro

Una calibración correcta es esencial para que el ratón se mueva con precisión y recorra toda la pantalla.

Calibrar el giroscopio:

1. En PSMoveConfigTool, selecciona el mando conectado.
2. Elige la opción Calibrate Gyroscope.
3. Espera a que el proceso termine (debe mostrar “Completed”).

Calibrar el acelerómetro:

1. Selecciona Calibrate Accelerometer.
2. Sigue las instrucciones para colocar el mando en varias posiciones.
3. Cada posición debe mantenerse quieta hasta que el sistema la registre.

Calibrar el magnetómetro (opcional pero recomendado):

1. Selecciona Calibrate Magnetometer.
2. Gira lentamente el mando en todas direcciones.
3. El objetivo es llenar una esfera virtual con puntos (representa la lectura del campo magnético).
4. Finaliza cuando el sistema indique una cobertura satisfactoria.

3.4 Script de Python

El siguiente script desarrollado en Python tiene como objetivo interpretar en tiempo real los datos de movimiento provenientes del mando PlayStation Move, a través del servicio PSMoveService y el middleware FreePIE, para simular el comportamiento de un ratón convencional en un sistema operativo Windows.

```
import ctypes
from ctypes import wintypes # Librería para acceder al mouse

def move_mouse(dx, dy): # Permite mover el mouse a coordenadas relativas
    pt = wintypes.POINT()
    ctypes.windll.user32.GetCursorPos(ctypes.byref(pt))
    ctypes.windll.user32.SetCursorPos(pt.x + dx, pt.y + dy)

def apply_deadzone(value, threshold=0.1): # Zona muerta: evitar movimientos
involuntarios como temblores
    return 0 if abs(value) < threshold else value

def update():
    global cursorScale

    # Botones
    mouse.leftButton   = freePieIO[3].yaw == 1
    mouse.rightButton  = int(freePieIO[3].x) & 0b00010000 > 0
    mouse.middleButton = int(freePieIO[3].x) & 0b00000001 > 0

    # Movimiento con zona muerta
    yaw = apply_deadzone(freePieIO[1].yaw)
    pitch = apply_deadzone(freePieIO[1].pitch)

    # Calcular las nuevas coordenadas
    dx = int(-yaw * cursorScale * 4)
    dy = int(-pitch * cursorScale * 4)

    move_mouse(dx, dy)

if starting:
    cursorScale = 5  # Sensibilidad
    freePieIO[0].update += update
```

4. Usos de PS Move en PC

4.1 Juegos y entretenimiento

- Juegos y simuladores de manos o herramientas VR
- Gamepad alternativo
- Control de juegos emulados de plataformas como Wii

4.2 Proyectos de desarrollo

- En entornos de diseño para controlar pinceles virtuales
- Desarrollo de interfaces naturales (sin teclado y ratón)
- Desarrollo de software para personas con movilidad reducida

4.3 Creación artística

- Pintura o escultura 3D en entornos virtuales
- Control de música gestual
- Manipulación en entornos virtuales

4.4 Sustituto de periféricos

- Control remoto multimedia
- Control de drones o robots