Instituto Tecnológico de Costa Rica

Algoritmos y Estructuras de Datos I

Leonardo Araya Martínez

Proyecto #1

Connect Dots

Documento De Diseño

Emanuel Rojas Fernandez

Gabriel Fernandez Vargas

Jimmy Feng Feng

II Semestre 2023

Índice

1. Listado de requerimientos del sistema:

- Como usuario, me gustaría que la interfaz contenga una descripción general del juego con sus respectivas instrucciones
- Como jugador, quiero tener un control físico para controlar el juego
- Como usuario me gustaría que el control sea fácil de utilizar y sencillo de entender
- Como administrador me gustaría que las acciones realizadas en el control se envíen de manera rápida al servidor y con poco delay
- Como administrador me gustaría que el videojuego contenga un servidor lo suficientemente estable y rápido para la conexión efectiva de los jugadores
- Como usuario, quiero que el juego contenga una interfaz ordenada y fácil de entender
- Como sistema, quiero que los datos que se envía en el servidor sea por formato JSON
- Como usuario me gustaría que al alcanzar cierta cantidad de cuadros completados el juego se acabe
- Como jugador me gustaría que haya forma de diferenciar los cuadros completados por los diferentes usuarios dentro del juego
- Como usuario me gustaría que haya puntos acumulables por cada cuadro creado

2. Elaboración de opciones de solución al problema

Opción 1: Uso de JSerialComm para la conexión serial y manejo de acciones

JSerialComm es una biblioteca de Java que proporciona acceso a puertos seriales de una computadora sin ninguna dependencia externa. Para implementar esta solución, se debe desarrollar un control físico que permita enviar un comando específico a la aplicación a través de un puerto serial, que luego se traduce en una acción en el juego. Un ejemplo de la implementación de esta opción se puede visualizar en el siguiente diagrama:

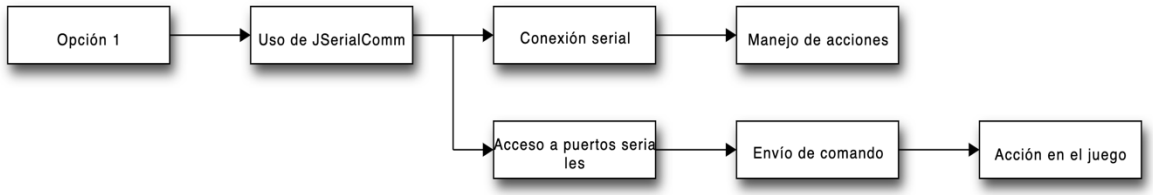

Figura 1. Diagrama de bloques de la opción 1

Opción 2: Control físico con Python usando PyAutoGUI

En esta opción, también implica desarrollar un control físico que, cuando interactúe el usuario, envía comandos a una aplicación Python. Esta aplicación de Python, al recibir los comandos del control, usa la librería PyAutoGUI para emular acciones de teclado y mouse para controlar el juego. Un ejemplo de la implementación de esta opción se puede visualizar en el siguiente diagrama:

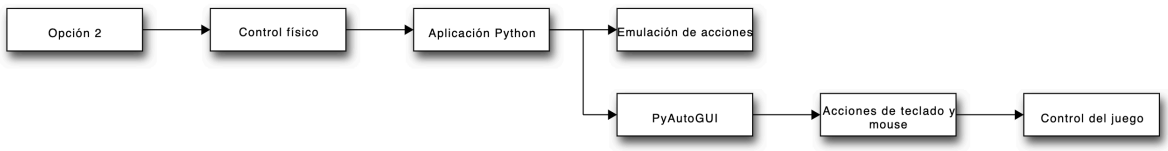

Figura 2. Diagrama de bloques de la opción 2

3. Valoración de opciones de solución

Opción 1: JSerialComm

Ventajas:

- Rendimiento: Ofrece una comunicación serial eficiente con menor sobrecarga, ya que está integrando directamente en la aplicación Java.
- Integración directa: Al ser una biblioteca Java, se integra directamente, evitando intermediarios.
- Independencia del SO: Funciona en diferentes sistemas operativos

Desventajas:

- Requiere conocimiento avanzado de Java: Necesario para la implementación y el manejo de la comunicación serial.
- Dependencia de la biblioteca: Si la biblioteca deja de recibir soporte, puede ocasionar problemas de funcionamiento.

Opción 2: Control físico con Python usando PyAutoGUI

Ventajas:

- Flexibilidad: Ofrece una amplia gama de opciones para simular entradas.
- Fácil de implementar: PyAutoGui es una biblioteca muy sencilla y no requiere mucho conocimiento superior en el lenguaje para implementarlo.
- Adaptable: Se pueden hacer cambios rápidamente y adaptar para otros juegos o aplicaciones.

Desventajas:

- Múltiples componentes: Se requiere manejar tanto el código de Python como el hardware del control.
- Tiempo de inicio: Se debe iniciar la aplicación Python para establecer comunicación con el control físico, este proceso puede llevar consigo un tiempo de espera adicional antes de que el jugador pueda comenzar a jugar.

3.1. Valoración basada en criterios adicionales:

- Salud y seguridad pública: Ambas opciones tienen un impacto mínimo en este criterio, ya que no genera ningún tipo perjuicio hacia la salud y seguridad pública.
- Costo total de vida: La opción 2, al ser más adaptable, podría resultar menos costosa si se requiere cambios frecuentes.
- Carbono neto cero: Ambas opciones generan un impacto mínimo en este criterio.
- Aspectos relaciones con recursos, culturales, sociales y ambientales: La opción 2 es más reutilizable y adaptable para diferentes culturas o necesidades sociales. Con respecto a los recursos, ambas opciones solamente se necesitaría elaborar un control físico y establecer la conexión.

4. Selección de la propuesta final

Después de analizar ambas opciones en detalle, se ha seleccionado la opción 2, control físico con Python usando PyAutoGUI como propuesta final.

Justificación:

- Una de las mayores ventajas de la opción 2 es su flexibilidad. Con Python y PyAutoGUI, se puede adaptar fácilmente nuestra solución para satisfacer nuevas necesidades o cambios en el juego.

- Con las bibliotecas y herramientas que dispone Python, el desarrollo y la implementación puede ser más rápido en comparación con soluciones nativas.
- Python tiene una comunidad activa y en constante crecimiento, lo cual asegura una amplia gama de recursos, tutoriales y ayuda en caso de problemas o dudas.

4.1. Consideraciones para la implementación:

Es esencial tener en cuenta las desventajas y retos que se presentaron anteriormente. Es vital trabajar en los siguientes puntos:

- Optimización del rendimiento: Se debe asegurar que la aplicación Python esté bien optimizada para no consumir recursos innecesarios.
- Pruebas extensivas: Se debe realizar pruebas extensivas para asegurar la compatibilidad y correcto funcionamiento del control para tener una buena experiencia de usuario.

5. Diseño de la alternativa seleccionada

Protocolo de Comunicación Arduino-Python usando Serial:

5.1. Protocolo

5.1.1. Objetivo del Protocolo:

Establecer una comunicación eficiente entre un control físico basado en Arduino y una aplicación Python para controlar acciones de teclado y ratón en un juego o aplicación.

5.1.2. Capas del Protocolo:

- Capa de Comunicación Serial: Encargada de transmitir los datos entre Arduino y la computadora que ejecuta la aplicación Python.
- Capa de Aplicación: Define cómo se interpretan los datos recibidos en la aplicación Python para realizar acciones específicas.

5.1.3. Formato de Datos:

- Los datos se transmiten como caracteres a través de la comunicación serial.
- Cada comando se envía como un solo carácter, seguido de un carácter de nueva línea ('\n') para separar cada comando.
- Ejemplos de comandos:
  - 'A': Presionar la tecla 'A' en la aplicación.
  - 'B': Realizar un clic del ratón.
  - 'I': Mover el cursor hacia arriba.
  - 'K': Mover el cursor hacia abajo.
  - 'J': Mover el cursor hacia la izquierda.
  - 'L': Mover el cursor hacia la derecha.

5.1.4. Proceso de Comunicación:

- Arduino monitorea el estado de los botones y el joystick y envía comandos a la aplicación Python cuando se detectan eventos (por ejemplo, presionar un botón o mover el joystick).
- La aplicación Python lee continuamente desde el puerto serial y ejecuta acciones en función de los comandos recibidos.
- Se utiliza un carácter de nueva línea ('\n') como delimitador para identificar el final de cada comando.

5.1.5. Manejo de Errores:

La aplicación Python debe manejar posibles errores de comunicación, como desconexiones inesperadas o comandos no reconocidos, para garantizar un funcionamiento confiable. El control al desconectarse muestra un error que se desconectó el control.

5.1.6. Seguridad:

Se deben tomar precauciones para garantizar que la comunicación entre Arduino y la aplicación Python sea segura y estable.

Este protocolo proporciona una base sólida para la comunicación entre el control físico basado en Arduino y la aplicación Python, lo que permite el control efectivo de acciones de teclado y ratón en el contexto de un juego o aplicación. Es importante que tanto Arduino como la aplicación Python estén sincronizados en cuanto a la interpretación de los comandos y el manejo de la comunicación para lograr un funcionamiento correcto.

5.2. Diagramas:

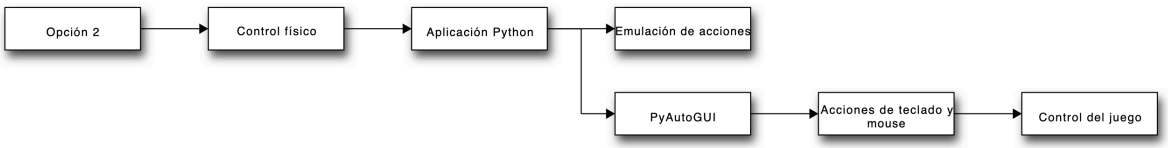

Figura 3. Diagrama de bloques

El diagrama de bloques describe el proceso que realiza el sistema de control. El control físico interactúa con la aplicación Python para mandar comandos. La aplicación de Python recibe comandos para presionar una tecla o mover el puntero del ratón gracias a la librería PyAutoGUI y así podrá interactuar con el juego “Connect Dots”.

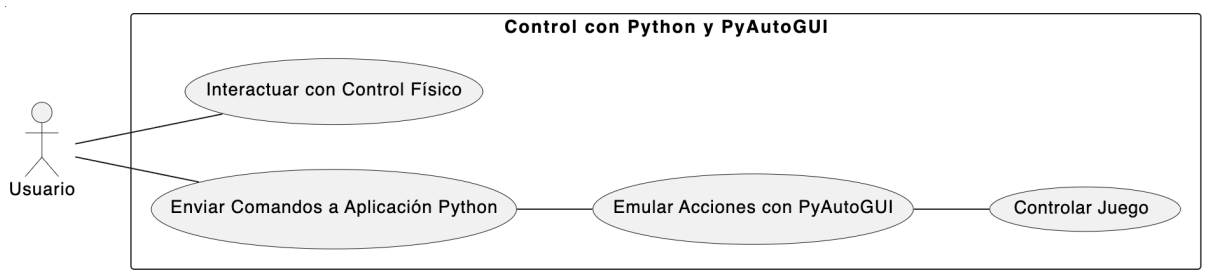

Figura 4. Diagrama UML de casos de uso

En este diagrama podemos observar como lo que pasa al utilizar el control y cómo funciona su conexión al juego por medio de Python y emulando las acciones con PyAutoGUI.

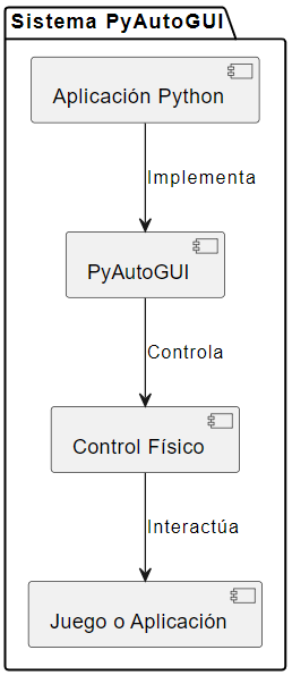

Figura 5. Diagrama de composición

El diagrama de composición describe como es la estructura de la implementación del control usando PyAutoGUI para interactuar con el juego.

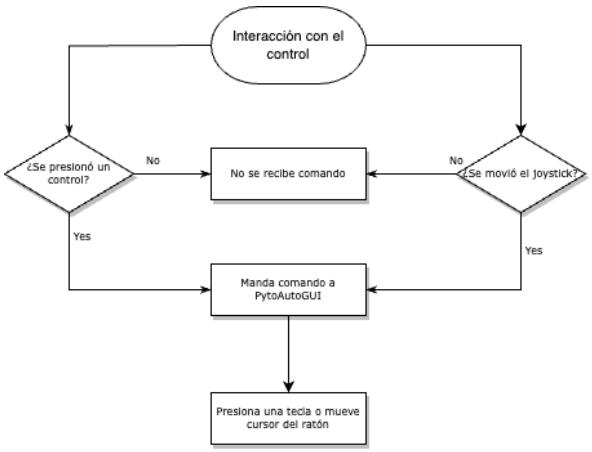

Figura 6. Diagrama de flujo

El diagrama de flujo describe las posibles acciones que puede ocurrir al interactuar con el control. Al ser una implementación sencilla, no presenta muchos condiciones, lo cual facilita la interacción con el juego.

6. Validación del diseño

6.1. Validación de Requerimientos

Se verificó que la opción cumple con los requerimientos del sistema que se mencionaron anteriormente. El diseño propuesto permite la comunicación efectiva entre el control físico basado en Arduino y la aplicación Python para controlar el juego de acuerdo con las instrucciones de los administradores.

6.2. Costo Total de la Vida

La evaluación de los costos debe llevarse a cabo considerando los componentes físicos (Arduino y hardware asociado), el desarrollo y mantenimiento del software (aplicación Python) y los costos operativos a largo plazo. Por lo tanto,

6.3. Carbono Neto Cero

Esta implementación no tiene un impacto ambiental muy fuerte, ya que gran parte del control es de software. En la parte de hardware, se pretende minimizar la huella de carbono por medio de la utilización de la tecnología impresión en 3D para la estructura del control y así poder controlar la cantidad de emisión de gases de efecto invernadero.

6.4. Aspectos Relacionados con Recursos, Culturales, Sociales y Ambientales:

- Recursos: Se promueve la reutilización de componentes y el reciclaje de materiales al final de la vida útil del producto.
- Culturales y Sociales: Se ha evitado la insensibilidad cultural y se fomenta la colaboración y competencia entre los jugadores para promover interacciones positivas.

- Ambientales: Se ha prestado atención a la eficiencia energética en hardware y software, la minimización de residuos y la sostenibilidad a largo plazo.