UNIVERSIDAD DE CONCEPCIÓN

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA MECÁNICA

ESTABLECIMIENTO DE PROTOCOLO DE COMUNICACIÓN ENTRE X-PLANE Y MICROCONTROLADORES EXTERNOS EN EL SIMULADOR DE VUELO DEL LABORATORIO DE TÉCNICAS AEROESPACIALES

POR

Germán Adolfo Quijada Arriagada

Profesor guía:

Bernardo Andrés Hernández Vicente

19 de mayo de 2023

Agradecimientos

Elif

Índice general

Índice de figuras

Capítulo 1

Definición del problema

1.1. Planteamiento del problema

1.1.1. Contexto y justificación

Los programas de ingeniería aeroespacial además de impartir conocimientos de las ciencias básicas, abren el camino a la especialización ingenieril en las áreas aeronáutica y espacial, siendo uno de los tantos temas abordados la introducción a los procedimientos para ejecutar un vuelo y los fenómenos físicos involucrados que lo hacen posible, lo que abre las puertas a posterior investigación relacionada donde contar con el equipo adecuado que permita realizar simulaciones de vuelo que sustenten estos estudios puede ser muy útil [30].

Las simulaciones permiten acceder o modificar arbitrariamente variables de la aeronave y el entorno, con la ayuda de interfaces que extraen de la simulación el estado de los sistemas involucrados y lo exportan a otros medios en los que sea más fácil hacer uso de los datos, ya sea para visualizarlos en tiempo real en un panel de instrumentos con indicadores, registrar información de la condición de la aeronave durante el vuelo para posterior evaluación de rendimiento, o la modificación directa de la simulación para estudiar condiciones específicas, entre otros. Todo sin dejar de lado el realismo y por lo tanto autenticidad que ofrece el software de simulación evitando lo costoso de realizar los mismos estudios en aeronaves de verdad.

Sin embargo, existen barreras tecnológicas que ralentizan el uso de un simulador de vuelo por lo compleja que puede ser su interfaz gráfica o lo exigentes que sean en términos del manejo de una aeronave, el usuario debe en primera instancia contar con el hardware adecuado que pueda ejecutar el simulador y familiarizarse con el funcionamiento interno del software para lograr sus fines. El objetivo de este proyecto de ingeniería abarca principalmente el caso de lograr comunicar el estado de la simulación a un hardware externo y que este pueda enviar comandos en respuesta, estableciendo una interfaz dedicada para ello que sea versátil y fácil de empezar a utilizar, junto con las instrucciones pertinentes.

En la simulación el uso de hardware externo que no sea la computadora principal es algo común por distintos motivos, como el querer emular la experiencia de un avión real lo que se traduce en la creación de controles e indicadores personalizados parecidos a los presentes en un panel de instrumentos, usualmente conectados a un

microcontrolador que debe interactuar con el software por medio de alguna interfaz. Además, y sobre todo en un contexto de investigación donde se deban realizar múltiples pruebas iterativas, el que se trabaje con equipos externos evita que se utilicen recursos que pudiese requerir la simulación y de manera más importante también se evita el que se hagan modificaciones al software o hardware del equipo principal en la instalación de los programas para la investigación y luego en la desinstalación una vez los análisis concluyan, así el simulador se mantiene íntegro asegurando su disponibilidad para ser utilizado en un futuro con otro propósito.

1.1.2. Software utilizado

Algunas soluciones existentes para comunicar simuladores de vuelo con microcontroladores son SimVimX [25] y MobiFlight [18], estos programas están diseñados para entusiastas que construyen paneles de instrumentos personalizados conectados a microcontroladores como Arduino o a otros computadores, habilitan la integración de los paneles proporcionando manuales e interfaces gráficas que ayudan a programar todo.

Debido al enfoque que tienen SimVimX y MobiFlight, las funciones que puede cumplir el microcontrolador están limitadas a hacer de interfaz para paneles de instrumentos, y no permiten mayores prestaciones sin modificar su código para su uso en otros fines.

Flight Simulator Universal Inter-Process Communication [3] o FSUIPC es un software que accede a la memoria interna de un simulador como Microsoft Flight Simulator [15] o Prepar3D [7], que permite a otros programas leer y modificar variables internas de la aeronave o el entorno, muchos plug-ins existentes dependen de esta interfaz para funcionar como por ejemplo MobiFlight. Además, el protocolo que establece FSUIPC puede ser considerado como un estándar, ya que existe otro software que lo reimplementa como XPUIPC [26] o XConnect [28] que habilita a los programas diseñados para funcionar con FSUIPC a que también lo hagan con X-Plane [23].

Programar software que aproveche FSUIPC otorga las ventajas de compatibilidad con cualquier simulador soportado por la librería o sus reimplementaciones como XPUIPC, y el poder concentrar el desarrollo del establecimiento del protocolo a la comunicación externa con microcontroladores, puesto que la manipulación interna de memoria para cada simulador ya esta cubierta.

1.1.3. Otras soluciones

X-Plane Connect [19] o XPC es otra interfaz diseñada por la NASA que cumple funciones similares a FSUIPC, pero específica para X-Plane sin posibilidad de adaptarla a otros simuladores. Lo interesante de esta alternativa es que incluye una librería para programar en MATLAB [9] y por consiguiente Simulink [11] lo que facilita el análisis de sistemas de control. SimConnect [16] desarrollado por Microsoft es una interfaz de bajo nivel como XPC o FSUIPC diseñada exclusivamente para Microsoft Flight Simulator 2020 también con características similares a las demás librerías.

1.2. Objetivos

1.2.1. Objetivo general

Reconfigurar simulador de vuelo del laboratorio de técnicas aeroespaciales, luego establecer un protocolo de comunicación entre el software del simulador y hardware externo, finalmente documentar la utilización del protocolo de comunicación y del simulador.

1.2.2. Objetivos específicos

1. Reconfigurar la disposición del hardware del simulador con el objetivo de facilitar su uso y agilizar en el futuro la instalación de nuevo software. Se registrará y comparará el hardware actual del simulador con el hardware nuevo disponible en el laboratorio.

2. Puesta en marcha del software de simulación. Instalación de sistema operativo Windows 10 actualizado y X-Plane, configurar el panel de instrumentos y controles.

3. Establecer protocolo de comunicación entre software del simulador y equipos externos, primero definiendo un protocolo de comunicación sobre IP y Serial diseñado para microcontroladores que soporten estas funciones, luego preparar con librerías existentes la extracción y manipulación de variables internas del simulador para finalmente combinar el protocolo de comunicación y la implementación de la librería en un programa personalizado.

4. Documentar uso del simulador preparando manuales instructivos referentes a la configuración de hardware y software del simulador, y el trabajo con el protocolo de comunicación con dispositivos externos.

1.3. Condiciones de diseño

1.3.1. Simulador

Actualmente el simulador en el laboratorio de técnicas aeroespaciales está limitado a X-Plane 9 debido a su disposición de 3 computadores de bajas prestaciones corriendo en paralelo, y cualquier mejora que se quiera hacer sobre el sistema debe cumplir con las condiciones que esa configuración implica, por esto se busca un diseño nuevo que permita más fácilmente trabajar en el simulador.

La nueva disposición de hardware consistirá en reducir la cantidad de computadores a solo uno, esto corta significativamente el tiempo que toma cualquier proceso que tuviese que realizarse en los tres computadores, elimina la necesidad de un router y cables que los conecten, y reduce la latencia entre las imágenes que muestran distintos monitores entre otras cosas.

Respecto al software, se va a realizar una instalación de Windows 10 compatible con gran parte si no todos los simuladores de vuelo en el mercado. Se instalarán X-Plane 9 y 10 disponibles por la universidad los cuales deberán configurarse de modo que trabajen con los múltiples monitores.

1.3.2. Protocolo de comunicación

El nuevo código que se vaya a escribir para el protocolo de comunicación utilizara estándares ya establecidos en simuladores, y la implementación en el lado del hardware externo será en un lenguaje que compartan los microcontroladores o microprocesadores como Arduino y Raspberry con mínima modificación.

Reducir la complejidad del protocolo de comunicación es clave, pues se está buscando que esta interfaz pueda ser aprovechada por usuarios no necesariamente familiarizados con la programación para que puedan cumplir sus objetivos, los cuales van más allá que solo crear paneles de instrumentos, como por ejemplo control y pruebas automatizadas.

1.3.3. Documentación

Respecto a la documentación esta será en formato tipo manual, abarcará la nueva configuración del simulador y el trabajo con el protocolo de comunicación, desde el encendido de los equipos hasta el uso del protocolo con ejemplos para comenzar. Incluirá la solución a problemas comunes que se vayan notando los que no necesariamente deban ser técnicos.

1.4. Metodología de trabajo

1.4.1. Simulador

Para reconfigurar el simulador de acuerdo con las necesidades y las condiciones de diseño establecidas en primera instancia se realiza un levantamiento de antecedentes “en terreno” de la configuración actual a una profundidad suficiente para volver a implementarla en caso de que se quiera volver ella. El nuevo hardware disponible en el laboratorio también se analizará con su documentación para comparar sus capacidades con el hardware actual.

Hecho esto, se investigarán las alternativas y posibilidades de habilitar la máxima cantidad de monitores por computador considerando el hardware disponible. Finalmente se ejecutará el nuevo diseño del simulador y se instalara y configurara el software.

1.4.2. Protocolo de comunicación

El comienzo del desarrollo del protocolo de comunicación consistirá en establecer el canal sistema operativo - microcontrolador y la librería para trabajar con el simulador.

1. Seleccionar canal de comunicación entre el sistema operativo y hardware externo

Para lograr un enlace entre un microcontrolador y el software de simulación primero es necesario establecer comunicación entre el microcontrolador y el sistema operativo, esto requiere caracterizar los canales disponibles en el hardware externo que se vaya a utilizar, los microcontroladores por lo general cuentan con una interfaz serial e inalámbrica por Bluetooth o wifi, también habrá que averiguar los lenguajes en los que programarlos comparando sus características de acuerdo a los objetivos.

2. Seleccionar software que permita la extracción y manipulación de variables del simulador

Primero se deben comparar las librerías que permiten trabajar sobre las variables del simulador según su lenguaje de programación, software de simulación que soportan y si sigue siendo activamente desarrollada.

Una vez tomadas las decisiones se procede a escribir el programa que se ejecutara en el sistema operativo, que actúe de interfaz entre el software de simulador y el microcontrolador, comunicados por medio de un protocolo personalizado que trabaje por sobre el de la librería y transmitido por el canal seleccionado.

1.4.3. Documentación

Con el simulador reconfigurado, se invitará a miembros de la carrera interesados a probar y trabajar con el equipo para obtener realimentación sobre su uso y verificar el cumplimiento de los objetivos que se buscaban, con posibilidad de realizar modificaciones menores en respuesta a lo que se vaya reportando, además de registrar problemas comunes que deban ser mencionados en la documentación del uso del simulador.

Finalmente, con base en la experiencia y los registros tomados durante el desarrollo se crearán manuales que describan el uso y procedimientos para realizar trabajos de interés con el protocolo de comunicación, que estarán disponibles de manera física y online en un repositorio dedicado al simulador.

Capítulo 2

Simulador

El simulador de vuelo del laboratorio de técnicas aeroespaciales es una herramienta muy útil para los alumnos de la universidad, otorgando una primera aproximación a los procedimientos necesarios para ejecutar un vuelo, y permite familiarizarse con los conceptos teóricos que se ven en clases de una manera práctica.

Durante los años varios alumnos han trabajado con el equipo haciendo mejoras o investigación y es la forma en la que se ha mantenido el simulador operativo hasta hoy. Al estar relacionado el principal objetivo de este proyecto de ingeniería con el simulador, se hace notar la oportunidad de trabajar sobre el simulador en sí y llevar a cabo el proceso de su actualización con mejores componentes que están disponibles en el laboratorio, pero que aún no están instalados.

En este capítulo se describe el proceso de actualización del equipo del simulador en hardware y software como está descrito en la metodología, los problemas que surgieron y el diseño final, comenzando por la caracterización del simulador en su condición inicial.

2.1. Configuración inicial

El equipo consiste en 3 computadores conectados a 7 pantallas con una disposición de 5 pantallas abarcando el campo de visión horizontal y 2 pantallas auxiliares para mostrar instrumentos del panel, periféricos de control similares a los encontrados a una avioneta Cessna 172 [31], sistema de sonido y red local con router (2.1). El computador principal cuenta con Windows 10, procesador Intel i7 de tercera generación y tarjeta gráfica NVIDIA GTX 1060 (2.2), los dos computadores secundarios tienen Windows 7 y prestaciones mucho más bajas, pero suficientes para ejecutar X-Plane 9.

Cada computador corre una instancia de X-Plane 9, el principal dirige la simulación, recibe la entrada de los controles y envía el estado de la simulación a los computadores secundarios por medio de la red local, los que replican el estado y actúan como extensión del campo de visión y despliegue del panel de instrumentos, todo configurado de acuerdo a las instrucciones del manual de usuario de X-Plane 9 [24].

Figura 2.1: Simulador de vuelo con configuración original

Figura 2.2: Computador principal con configuración original

2.2. Alternativas de diseño

Se solicita instalar dos componentes nuevos con el objetivo de ejecutar X-Plane 10 y posiblemente simuladores más recientes en el futuro, los cuales son una tarjeta gráfica NVIDIA GTX 1080 y una fuente de poder de 700 W. Considerando la configuración inicial del simulador y los objetivos, la instalación de los nuevos componentes no es un tema de solo reemplazar los antiguos, ya que los computadores secundarios seguirán obsoletos y limitados a X-Plane 9.

2.2.1. Dos computadores

Una solución plausible es actualizar y reemplazar los componentes en el computador principal, el mejor de los secundarios actualizarlo con los componentes sobrantes (GTX 1060 y fuente) y descartar el tercer computador. Es inmediatamente la solución con menos complicaciones y (sin mencionar algunos detalles) compatible con la configuración de X-Plane ya implementada y descrita en los manuales, sin embargo tiene muchos problemas que surgirán a futuro.

En rendimiento la GTX 1080 y 1060 soportan con creces X-Plane 10 (que no es mucho más exigente que X-Plane 9 gráficamente), pero el procesador de todos los computadores del simulador también requiere actualización, la tercera generación de Intel salió en 2012 y con nuevo software como X-Plane 12 esta empieza a mostrar sus limitaciones, el tener 2 computadores implica que en futuras actualizaciones se deban comprar el doble de

componentes, los que probablemente solo cumplan con los requisitos mínimos de los simuladores nuevos, en vez de actualizar solo un computador con todo el presupuesto y asegurarse que podrá ejecutar el software de sobra.

Dos o más computadores requieren múltiples instalaciones del software de simulación (que para el caso de X-Plane implica comprar más licencias [27]), configuración para la ejecución automática del software en cada computador si no se quiere tener que usar múltiples periféricos, o implementar soluciones para conectar periféricos a múltiples computadores, configurar una red local que asigne una IP estática distinta a cada computador para la posterior sincronización del software de simulación, si es que estos la soportan como X-Plane. Estos detalles y otros no mencionados significan mayor complejidad para trabajar en el simulador o poner en marcha nuevo software en el futuro.

2.2.2. Un computador

En vista de los problemas recién mencionados, se evalúa la opción de usar solo un computador asignándole los mejores componentes disponibles, lo que elimina gran parte de las complicaciones de tener dos o más, pero que aún cumpla con los objetivos inmediatos. Implementar la configuración existente del simulador es fácil, por ejemplo si antes se tenían 3 instancias de X-Plane en computadores distintos ahora se pueden tener abiertas 3 instancias, pero en solo un computador con múltiples pantallas, claro que hacer eso es posible una vez puesto el hardware en marcha donde debido a las circunstancias surgen complicaciones.

2.2.3. Múltiples pantallas

No se pueden conectar 7 pantallas a una sola tarjeta de video, la GTX 1080 y GTX 1060 cada una permite hasta 4 monitores, el procesador Intel permite hasta 3 salidas de video con sus gráficos integrados [6], pero la placa madre tiene solo dos salidas, una digital y una análoga. Existen adaptadores USB a HDMI que habilitan más salidas de video, pero la información que existe de ellos es muy variada, hay reportes de que estos adaptadores pueden reducir el rendimiento además de que cuestan dinero y no se puede asegurar que vayan a ser compatibles.

Usando solo el material disponible en el laboratorio se decide por conectar ambas tarjetas de video simultáneamente al computador habilitando hasta 10 salidas, 8 desde las GPU y 2 de los gráficos integrados del procesador. Esto puede suponer un riesgo de recalentamiento, pero se esperará a ver con el tiempo si esto resulta en un problema a largo plazo, en caso de altas temperaturas se confía en que las unidades de procesamiento incluyen sensores que apagan el sistema antes de cualquier situación peligrosa para evitar daños.

Ambas tarjetas de video además de la interfaz PCI con la placa madre requieren ser conectadas directamente a la fuente de poder con cables de 6 y 8 pines dedicados para funcionar. La nueva fuente de poder no cuenta con los suficientes cables para ambas GPU, se podría comprar o fabricar un cable splitter para lograr conectar todo, pero supone mucho riesgo de recalentamiento por la calidad del splitter y la cantidad de energía que va a transmitir, la nueva fuente no está diseñada para eso.

Usar dos fuentes de poder asegura que se cuenta con suficiente energía para alimentar ambas tarjetas de video sin riesgo a recalentamiento de cables, no se necesitan modificar internamente los componentes, puesto que aun en esta configuración están trabajando dentro de los parámetros establecidos por los fabricantes.

2.3. Configuración final

La nueva configuración consiste en reducir la cantidad de computadores de tres a solo uno, el cual en el futuro pueda ser actualizado con un buen procesador para ejecutar simuladores modernos como Flight Simulator 2020 o X-Plane 12. Además reduciendo la complejidad de tener que preparar múltiples computadores a solo tener que configurar en uno el software, donde en los casos que no exista soporte oficial para tantas pantallas (como en X-Plane 9 y 10) se pueden preparar scripts que abran múltiples instancias del simulador que repliquen la antigua configuración.

2.3.1. Hardware

Se instalan los nuevos componentes al computador principal y se prueba que funcionen correctamente, las unidades de procesamiento del equipo en este punto son su procesador Intel i7-3770k y una tarjeta gráfica NVIDIA GTX 1080. Con esta configuración se ve limitada la cantidad de salidas de video disponibles como fue descrito previamente, por lo que se procede a realizar pruebas del funcionamiento del equipo al agregarse la tarjeta GTX 1060 y una segunda fuente.

Con la segunda tarjeta gráfica conectada al puerto PCI disponible, se prepara la antigua fuente de poder para un encendido manual [20] y se conecta para alimentar la GTX 1060, por HDMI se conecta una quinta pantalla al equipo el cual se enciende para comprobar su funcionamiento. Las pruebas son un éxito (2.3) y se decide por instalar la segunda fuente dentro del gabinete (2.4).

Figura 2.3: Prueba de encendido con segunda fuente de poder

Al no estar conectada directamente a la placa madre, la segunda fuente de poder necesita ser encendida y apagada manualmente al mismo tiempo que el computador, para resolver este problema se compra un dispositivo que sincroniza el encendido de la fuente auxiliar por medio de un relé (2.5).

Finalmente se conectan todas las pantallas al equipo, terminando el proceso de actualización (2.6) y dando paso a comenzar la configuración del software de simulación. Los componentes principales del computador por el momento son el procesador Intel i7-3370k, dos tarjetas gráficas NVIDIA GTX 1080 y NVIDIA GTX 1060, y se espera que se introduzca en el futuro un mejor procesador que permita ejecutar software de simulación actual como Microsoft Flight Simulator 2020 y X-Plane 12.

Figura 2.4: Instalación de segunda fuente con amarras plásticas

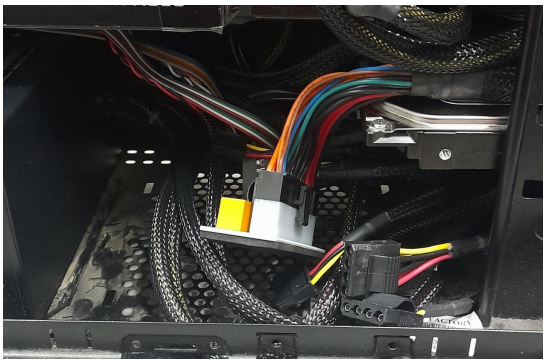

Figura 2.5: Dispositivo sincronizador de fuentes

2.3.2. Software

Preparar el software abarca desde formatear el dispositivo de almacenamiento hasta configurar los programas, de tal manera que desde el encendido del equipo se pueda rápidamente empezar una simulación y volar.

Desde los servidores de Microsoft se obtiene la imagen de disco más reciente de Windows 10 [13], se graba en un disco duro externo USB y se instala formateando el almacenamiento. Windows se encarga de descargar automáticamente todos los drivers necesarios, incluso el de los periféricos de control.

Posterior a la instalación, completada la información que se solicite y actualizado el equipo, se ordenan las pantallas en el panel de control de NVIDIA de manera que las 5 pantallas superiores formen una imagen continua y las 2 pantallas auxiliares se desplazan a la parte inferior, con la herramienta de administración de discos de Windows se divide la partición principal creando una nueva partición de 300 GB, con el propósito de almacenar los archivos de instalación del software como respaldo, como por ejemplo las imágenes de disco de X-Plane 9 y 10.

La universidad dispone de copias de X-Plane 9 y X-Plane 10, los archivos de X-Plane 9 se descargan desde una carpeta compartida en OneDrive proporcionada, estos archivos vienen en formato de imágenes de disco MDF y se instala el software WinCDEmu [29] para poder cargarlas uno a uno. De X-Plane 10 se cuenta con los discos físicos, el computador principal no tiene un lector de DVD por lo que temporalmente se conecta uno de los equipos secundarios que si tiene lector y se crean imágenes de disco con el programa ImgBurn [2], que luego

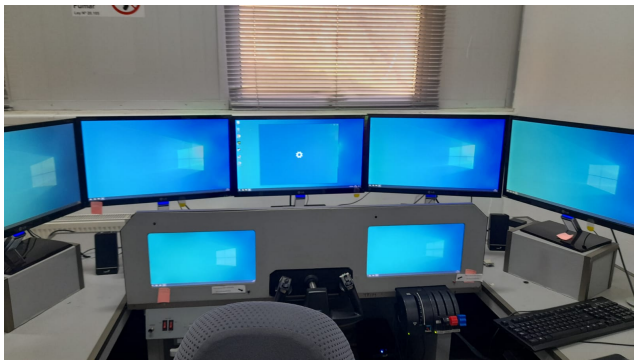

Figura 2.6: Simulador de vuelo con configuración final

se copian al principal para completar la instalación de la misma forma que como con X-Plane 9.

[Scripts para ordenar las multiples instancias de X-Plane aún en desarrollo]

2.4. Resultados

A los pocos días de haber terminado la configuración del hardware con múltiples pantallas, los integrantes del simulador de vuelo en el laboratorio de técnicas aeroespaciales realizaron pruebas con versiones de demostración de Flight Simulator 2020 y X-Plane 12 (2.7), a pesar de que el rendimiento era bajo los alumnos pudieron configurar los controles y realizar vuelos, incluso invitando a otros alumnos y docentes a pilotar y ver los resultados.

Figura 2.7: Pruebas con Microsoft Flight Simulator 2020

El software más reciente instalado por los integrantes del grupo de interés incluye soporte para configuración multi-pantalla sin necesidad de trucos como abrir más de una instancia del programa, por lo que fue una tarea fácil configurarlo, sin embargo los simuladores X-Plane antiguos proporcionados por la universidad requieren más trabajo porque la disposición multi-pantalla con un solo computador no está considerada en los manuales [24].

En conclusión se lograron los objetivos, se cumple la solicitud de actualizar el simulador con los nuevos componentes y se simplifica la disposición de tres computadores a solo uno, habilitando un camino claro de

actualización que es la compra de nuevas piezas como un procesador más reciente o almacenamiento de estado sólido, se reducen los tiempos necesarios para instalar nuevo software, configurarlo y se asegura un futuro próspero para el grupo de interés del simulador de vuelo.

Capítulo 3

Protocolo de comunicación

El protocolo de comunicación consiste en una selección de mensajes estructurados, diseñados para realizar ciertas acciones sobre una simulación de vuelo, los mensajes son enviados desde un equipo externo como un microcontrolador u otro computador convencional, a otro equipo que esté ejecutando software de simulación.

Este protocolo ofrece la capacidad de trabajar de manera programática con las variables que gobiernan una simulación, por ejemplo puede hacer todo lo que ya incluye X-Plane en su interfaz de usuario como guardar datos a un Excel o cargar un estado específico, pero además permite acceder a variables internas de la simulación y modificarlas arbitrariamente en tiempo real.

Ejemplos de uso pueden ser ejecutar acciones sencillas como automatizar una misión, o analizar como afectan diversas condiciones externas como el clima o comandos del piloto a una maniobra de manera iterativa, reiniciando repetidamente el estado de la aeronave a uno inicial pero alterando ciertas variables. Más interesante es la nueva habilidad de poder responder en tiempo real al estado de la simulación, que permite desarrollar sistemas de control desde simples PID en Python o C hasta complejos modelos en Simulink.

En el contexto del laboratorio de técnicas aeroespaciales, el diseño del protocolo de comunicación busca alcanzar un balance entre versatilidad y facilidad de uso, que permitirá a estudiantes interesados aplicar sus conocimientos de Python para manipular la simulación mediante código comprensible, pero que también sirva de herramienta potente para la investigación y no sea un factor limitante debido a que no posea cierta funcionalidad que se requiera.

En este capítulo se describe el proceso de desarrollo del protocolo de comunicación según la metodología presentada y las decisiones que fueron tomadas en el camino.

3.1. Microcontroladores

El objetivo principal de este proyecto de ingeniería es establecer el protocolo en microcontroladores, ventajas de trabajar con ellos es que permiten una fácil interacción con el mundo real desplegando información con luces LED, sonidos o pantallas y recibiendo entradas con botones o potenciómetros, además son muy rápidos y consistentes ejecutando las órdenes programadas.

Existen muchos tipos de microcontrolador con distintas funciones, la mayoría se debe programar en C y algunos soportan Python que es de especial interés al ser el lenguaje que se enseña en la universidad, tienen una variada cantidad de pines de uso general, distinto tamaño de las memorias, etc. Pero si hay algo que tienen en común es que soportan comunicación Serial usualmente por USB, y los que tienen el hardware necesario también comunicación inalámbrica por wifi y bluetooth.

El diseño del protocolo se centra en crear una estructura de mensajes que sea fácil implementar en cualquier microcontrolador, independiente del lenguaje o el canal. Esto se traduce en que no va a haber una librería para cada lenguaje tipo librerías Arduino o módulos Python que abstraen la comunicación, sino que primero se analizan los tipos de estructura de datos que los lenguajes de interés soportan por defecto y se diseña el protocolo alrededor de ellos aprovechando las librerías estándar ya establecidas.

Los microcontroladores seleccionados para comenzar el diseño son la familia de Arduino y el Raspberry Pi Pico, más importante que los modelos en específico es el lenguaje en el que se programan que por defecto es C, además algunos microcontroladores Arduino y Raspberry se pueden programar en Python con la implementación MicroPython [1].

3.1.1. Serialización de datos

Al no proporcionar librerías para cada lenguaje todo el proceso de comunicación debe programarlo manualmente el usuario, teniendo en cuenta esto, para que el protocolo aun así sea fácil de usar primero hay que analizar las formas que tienen estos lenguajes de empaquetar información para su transmisión.

Cadenas de caracteres

La manera más sencilla es concatenar información en cadenas de caracteres, que luego el compilador o interprete automáticamente transforma a bytes para su transmisión. Para el envío de mensajes puntuales y que no son formulados dinámicamente las cadenas de caracteres son ideales. Por ejemplo si se quiere enviar un mensaje que solicita monitorear las variables de velocidad y altitud de manera continua, se podría escribir una cadena así.

```
msg = "M;2B8:i;574:i;"
```

Los primeros dos caracteres `M;` indican que el mensaje es una solicitud de monitoreo de variables específicas, el segmento `2B8:i;` señala que se monitoree la variable en la memoria `2B8`, que contiene la velocidad, e `i` es el tipo de variable que en este caso es un número entero con signo de 32 bits.

Para recibir información o crear mensajes de manera dinámica las cadenas de caracteres ya no sirven, se complica la extracción de los datos al tener que emplear funciones como `string.split` que separen la cadena en subcadenas delimitadas por el carácter `:`, y la formulación de mensajes dinámicos depende de la concatenación de cadenas con el operador `+`.

Estructuras

Las estructuras son contenedores capaces de almacenar múltiples variables de distinto tipo en espacios continuos de memoria, una vez definida la estructura se puede recibir un mensaje en forma de secuencia de bytes

y convertirlo rápidamente a la disposición de la estructura con la que se puede trabajar fácilmente. Por ejemplo luego de enviar la previa orden de monitoreo, se recibirán continuamente los valores de velocidad y altitud en forma de estructura de números de punto flotante con doble precisión.

```
msg = sock_recv.recvfrom(1024)
tas, altitude = struct.unpack("<dd", msg)
print(tas, altitude)
```

La variable `msg` contiene el mensaje recibido con la información solicitada, la función `struct.unpack` lee el mensaje de acuerdo a la cadena de formato `"<dd"`, el carácter `<` indica que la información viene codificada en formato little-endian y los caracteres `dd` que el mensaje contiene dos números de punto flotante, uno por cada letra `d`.

SimuLink

Si bien se aleja un poco de los objetivos del proyecto, SimuLink es una herramienta muy útil para el control en tiempo real por lo que vale la pena revisar como enviar y recibir información en ella, además existe documentación para ejecutar modelos en microcontroladores [10].

El paquete “Instrument Control Toolbox” de SimuLink agrega bloques para enviar y recibir información por UDP/IP [8], los mensajes deben contener números en estructuras con la misma disposición que la descrita en el ejemplo anterior.

3.1.2. Canales de comunicación

Los microcontroladores Arduino y Raspberry Pi Pico permiten comunicación serial por USB, y los que tienen el hardware necesario pueden comunicarse por wifi y bluetooth.

Sobre la comunicación wifi se tiene el “Internet Protocol” o IP que asigna direcciones a cada equipo y habilita múltiples puertos por los que enviar o recibir información, y sobre IP puede implementarse el “Transmission Control Protocol” o TCP y “User Datagram Protocol” o UDP entre otros, que son protocolos que administran el envío de mensajes en paquetes con funcionalidad especifica, por ejemplo TCP asegura garantías en la comunicación con mecanismos de respaldo en caso de que los mensajes no sean recibidos o lleguen en un orden distinto al esperado, pero a costo de velocidad de procesamiento en la verificación y más pasos necesarios para establecer una conexión que como lo hace UDP.

User Datagram Protocol

Una vez establecida la conexión a una red wifi, enviar información en UDP por IP es posible conociendo la IP del equipo de destino y el puerto en el que este está esperando recibir mensajes, de la misma manera el microcontrolador puede abrir un puerto donde leer mensajes a medida que vayan llegando. Es un mecanismo muy simple que cumple con los parámetros de diseño.

```
import socket
import struct

# Inicializar estructuras de comunicacion
```

```
SERVER = ("192.168.1.10", "27015")
CLIENT = ("127.0.0.1", "27016")
sock_send = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock_recv = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock_recv.bind(CLIENT)

# Enviar mensaje de monitoreo
msg = "M;2B8:i;"
sock_send.sendto(bytes(msg, "ascii"), SERVER)

# Leer valores recibidos
for i in range(11):
  msg, addr = sock_recv.recvfrom(1024)
  tas = struct.unpack("<d", msg)
  print("TAS: " + str(tas[0]/128) + " knots")
```

El primer bloque define las estructuras necesarias para la comunicación, la variable `SERVER` almacena la dirección de destino y el puerto, y la variable `CLIENT` el puerto donde se recibirán mensajes.

El segundo bloque envía una cadena de caracteres codificada en "ASCII" al servidor con la orden de monitorear la variable en la memoria `2B8` con tipo de variable número entero con signo de 32 bits.

El tercer bloque recibe información del servidor y la lee desempaquetando la estructura que contiene solo un número de coma flotante con doble precisión.

UDP no tiene concepto de conexiones, solo permite enviar mensajes a cierta dirección sin otorgar retroalimentación de que estos hayan llegado o no, y al recibir mensajes la única garantía es que los datos al leerse se encuentran tal como fueron enviados, porque internamente el protocolo realiza una verificación de integridad con un checksum, de otra manera el mensaje se descarta. Estas limitaciones se pueden sobrellevar fácilmente para los casos en los que se requiera, ese trabajo tendrá que hacerlo el programa del servidor y no el cliente por lo que no agregara más complejidad al protocolo.

3.2. Interfaz a simulador

El programa que haga interfaz con el simulador tiene la responsabilidad de establecer comunicación con el microcontrolador y ejecutar acciones sobre la simulación de acuerdo a lo que se solicite, además de proporcionar reportes que ayuden al usuario a diagnosticar problemas de conexión o errores que se produzcan en los comandos que se reciban.

El usuario final no necesita conocer el funcionamiento interno del programa aparte del protocolo de comunicación con él, es aquí donde ocurre toda la abstracción del acceso a la memoria del simulador para lectura y escritura.

Considerando lo anterior y buscando facilitar el uso de este software, se decide que el programa interfaz estará escrito en C++ y se distribuirá como un archivo ejecutable para Windows, sin ninguna otra dependencia.

Acceder a la memoria interna del simulador es posible gracias a varias librerías como se mencionó en la sección de "Planteamiento del problema", de todas las existentes se opta por Flight Simulator Universal Inter-

Process Communication o FSUIPC, debido a que hasta hoy sigue en desarrollo y se ofrece soporte en los foros oficiales [21], existen reimplementaciones del protocolo para otros simuladores como XPUIPC [26], convirtiéndolo en un estándar y asegurando que funciona con X-Plane 9 y 10 que son los programas instalados en el simulador, y en el futuro con los que nuevos simuladores que vayan a salir [4].

3.2.1. Flight Simulator Universal Inter-Process Communication

Flight Simulator Universal Inter-Process Communication [3] es en primer lugar un plug-in para Microsoft Flight Simulator y Prepar3D que abre un canal de comunicación "entre-procesos" en el sistema operativo, y en segundo lugar es una librería que permite la comunicación con el plug-in para enviar y recibir datos internos de la simulación.

Su reimplementación para X-Plane, XPUIPC [26], reemplaza el plug-in original y ofrece uno que funciona con las versiones 9, 10 y 11 del simulador con la versión para X-Plane 12 aun en desarrollo [4], la librería sigue siendo la de FSUIPC, lo que implica que los programas escritos con la librería también funcionan en X-Plane y XPUIPC sin intervención del programador, formando un estándar.

La librería ofrece funciones muy básicas, pero efectivas, estas son `Open()` para conectar con el simulador y `Close()` para cerrar la comunicación, `Write(address, value)` para escribir en un lugar de la memoria un valor y `Read(address)` para leer un valor en la memoria. Las posiciones de la memoria están descritas en un documento incluido con FSUIPC [22] e indican que variable reside en que lugar y el tipo de variable, sea número entero o de coma flotante, si es con o sin signo y cuantos bytes ocupa.

3.3. Flight Simulator Simple Protocol

Considerando las decisiones de diseño descritas hasta ahora en el capítulo, se comienza la escritura del programa intermediario entre el microcontrolador y el simulador, primero estableciendo un entorno de desarrollo en C++ integrando la librería de FSUIPC.

Si bien el programa intermediario no será utilizado por el usuario final que solo trabajara con el protocolo, interesa que el entorno de desarrollo sea simple para facilitar el trabajo a un tercero que quisiera modificar el programa, por esto se decidió trabajar con Visual Studio Code [17] y las herramientas de compilación de C++ para Windows [12]. Básicamente solo un editor de texto, el compilador y el terminal, evitando entornos más complejos y pesados como Visual Studio completo.

El repositorio de trabajo se encuentra en la cuenta de GitHub del autor [5], la carpeta del proyecto se puede cargar en Visual Studio Code y, teniendo instalado el compilador de Microsoft para C++ [12] se debería tener todo lo necesario para replicar el entorno de desarrollo.

Primero se realizaron pruebas con la librería FSUIPC para comprobar que la versión para X-Plane funcionase correctamente, luego se estableció comunicación por UDP con la librería WinSock [14] entre el programa y Python, y luego con SimuLink, existiendo ejemplos en el mismo repositorio.

La primera implementación del protocolo incluye el monitoreo y control de variables de manera genérica, solicitando al usuario ingresar la ubicación de memoria y tipo de variable según la documentación de FSUIPC

[22], luego se van agregando comandos más específicos que sean más amigables como monitoreo de “Velocidad” y “Altitud” por sus nombres, no por sus ubicaciones en la memoria.

3.4. Protocolo

El protocolo habilita la comunicación entre un microcontrolador y un programa intermediario, el que accede a la memoria del simulador y la modifica o lee de acuerdo a los comandos que se reciben. La comunicación puede efectuarse por serial USB y por wifi en paquetes UDP por IP.

Los mensajes están formados por cadenas de caracteres o estructuras de números de coma flotante de simple o doble precisión dependiendo de la arquitectura del microcontrolador.

3.4.1. Monitoreo de variables genérico

Este comando activa el monitoreo de las variables definidas en una cadena de caracteres en un orden específico, en ese orden es en el que se empezara a recibir de manera continua el valor de las variables en una estructura con números de coma flotante.

Variables que se podrían querer monitorear son las que describen el estado del vuelo como la velocidad o la altitud.

```
# Monitoreo generico
# M;ubicacion1:tipo1;ubicacion2:tipo2;ubicacion3:tipo3;
#
# Ejemplo
# Enviar mensaje de monitoreo
msg = "M;2B8:i;"
sock_send.sendto(bytes(msg, "ascii"), SERVER)

# Leer valores recibidos
for i in range(11):
  msg, addr = sock_recv.recvfrom(1024)
  tas = struct.unpack("<d", msg)
  print("TAS: " + str(tas[0]/128) + " knots")
```

3.4.2. Escritura continua de variables genérica

Este comando define que variables se quieren escribir de manera continua y que tipos son, luego se deben enviar los valores en formato de número con coma flotante para que el programa los escriba y reemplace en la simulación.

Variables que se podrían querer escribir continuamente son los controles de la aeronave como el acelerador o el timón.

```
# Escritura generica
# C;ubicacion1:tipo1;ubicacion2:tipo2;ubicacion3:tipo3;
#
# Ejemplo
# Enviar mensaje de control
```

```
msg = "C;310A:c;089A:s;"
sock_send.sendto(bytes(msg, "ascii"), SERVER)

# Enviar valores del control
for i in range(11):
  # 4to bit habilita el control del acelerador
  # 00001000 = 8
  inputs = 8
  # El valor del acelerador va de -4096 a 16384
  throttle = 16384/10*i
  msg = struct.pack("<dd", inputs, throttle)
  sock_send.sendto(msg, SERVER)
  time.sleep(1);
```

3.5. Resultados

a

Bibliografía

[1] Damien George. *MicroPython*. URL: https://micropython.org/ (visitado 18-05-2023).

[2] Digital Digest. *ImgBurn*. URL: https://www.imgburn.com/ (visitado 17-05-2023).

[3] Peter Dowson y John Dowson. *Flight Simulator Universal Inter-Process Communication (FSUIPC)*. URL: http://www.fsuipc.com/ (visitado 15-05-2023).

[4] Enrico Schiratti. *X-Plane 12 Version of XPUIPC*. 26 de ene. de 2023. URL: https://www.projectmagenta.com/2023/01/x-plane-12-version-of-xpuipc/ (visitado 19-05-2023).

[5] Germán Quijada. *Repositorio Flight Simulator Simple Protocol*. URL: https://github.com/qgerman2/fssp (visitado 19-05-2023).

[6] Intel. *Especificaciones Intel Core i7-3770K*. URL: https://ark.intel.com/content/www/es/es/ark/products/65523/intel-core-i73770k-processor-8m-cache-up-to-3-90-ghz.html (visitado 16-05-2023).

[7] Lockheed Martin. *Prepar3D*. URL: https://www.prepar3d.com/ (visitado 15-05-2023).

[8] MathWorks. *Basic UDP Communication*. URL: https://la.mathworks.com/help/instrument/basic-udp-communication.html (visitado 19-05-2023).

[9] MathWorks. *MATLAB*. URL: https://mathworks.com/ (visitado 15-05-2023).

[10] MathWorks. *Run Model on Arduino Hardware*. URL: https://la.mathworks.com/help/supportpkg/arduino/ug/run-model-on-arduino-hardware.html (visitado 18-05-2023).

[11] MathWorks. *Simulink*. URL: https://mathworks.com/products/simulink.html (visitado 15-05-2023).

[12] Microsoft. *Configure VS Code for Microsoft C++*. URL: https://code.visualstudio.com/docs/cpp/config-msvc (visitado 19-05-2023).

[13] Microsoft. *Descarga Windows 10*. URL: https://www.microsoft.com/es-mx/software-download/windows10 (visitado 17-05-2023).

[14] Microsoft. *Getting started with Winsock*. 9 de feb. de 2023. URL: https://learn.microsoft.com/en-us/windows/win32/winsock/getting-started-with-winsock (visitado 19-05-2023).

[15] Microsoft. *Microsoft Flight Simulator*. URL: https://www.flightsimulator.com/ (visitado 15-05-2023).

[16] Microsoft. *SimConnect*. URL: https://docs.flightsimulator.com/html/Programming_Tools/SimConnect/SimConnect_SDK.htm (visitado 15-05-2023).

[17] Microsoft. *Visual Studio Code*. URL: https://code.visualstudio.com/ (visitado 19-05-2023).

[18] *MobiFlight*. URL: https://www.mobiflight.com/ (visitado 15-05-2023).

[19] NASA. *X-Plane Connect*. URL: https://github.com/nasa/XPlaneConnect (visitado 15-05-2023).

[20] Overclockers Club. *How To Turn On An ATX Power Supply Without A Motherboard*. URL: https://www.overclockersclub.com/guides/atx_psu_startup/ (visitado 17-05-2023).

[21] Peter Dowson. *FSUIPC Support*. URL: https://forum.simflight.com/forum/30-fsuipc-support-pete-dowson-modules/ (visitado 19-05-2023).

[22] Project Magenta. *FSUIPC Offsets Quick Reference*. URL: https://www.projectmagenta.com/download/8733/ (visitado 19-05-2023).

[23] Laminar Research. *X-Plane*. URL: https://www.x-plane.com/ (visitado 15-05-2023).

[24] Laminar Research. *X-Plane Operation Manual*. Ver. 9.61. URL: https://www.x-plane.com/files/manuals/X-Plane_Desktop_manual.pdf (visitado 16-05-2023).

[25] Roman Sychev. *SimVimX*. URL: https://simvim.com/ (visitado 15-05-2023).

[26] Torsten Spiering. *X-Plane Universal Inter-Process Communication (XPUIPC)*. URL: https://www.schiratti.com/xpuipc.html (visitado 15-05-2023).

[27] X-Plane Support. *Do I need to purchase another copy of X-Plane 11 for WED?* URL: https://questions.x-plane.com/10417/do-i-need-to-purchase-another-copy-of-x-plane-11-for-wed (visitado 16-05-2023).

[28] Mat Sutcliffe. *XConnect*. URL: https://xplaneconnect.sourceforge.net/ (visitado 15-05-2023).

[29] Sysprogs. *WinCDEmu*. URL: https://wincdemu.sysprogs.org/ (visitado 17-05-2023).

[30] Mark D White y Gareth D Padfield. «The Use of Flight Simulation for Research and Teaching in Academia». En: 2006.

[31] FlightGear Wiki. *Hardware Review: Saitek Pro Flight Cessna controls*. URL: https://wiki.flightgear.org/Hardware_Review:_Saitek_Pro_Flight_Cessna_controls (visitado 16-05-2023).