UNIVERSIDAD DE CONCEPCIÓN

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA MECÁNICA

IMPLEMENTACIÓN DE UNA ARQUITECTURA PARA VALIDACIÓN DE CONTROLADORES DE AERONAVES DE ALA FIJA EN X-PLANE

POR

Germán Adolfo Quijada Arriagada

Profesor guía:

Bernardo Andrés Hernández Vicente

21 de octubre de 2023

Índice general

Capítulo 1

Definición del problema

1.1. Contexto

discusión sobre la utilidad de RPAs autónomos y luego hablar sobre los tipos de controladores

1.2. Planteamiento del problema

Los software controlador de RPAS mantenidos actualmente como ArduPilot o Pixhawk incluyen sistemas de autopiloto y vuelo asistido generalmente a base de algoritmos PID. En algunos casos como con Pixhawk los detalles de implementación del controlador están documentados [1], permitiendo que un usuario experimentado pueda ajustar los parámetros en busca de un mejor desempeño. *Quizás se quiera lograr un vuelo más estable si se está pilotando manualmente [2], o en misiones autónomas reducir el consumo de las baterías, maximizar la distancia a recorrer o reducir el tiempo que una misión pueda tomar*. Otra manera de lograr estos objetivos puede ser reemplazando por completo el algoritmo PID por controladores basados en modelos dinámicos como lo son LGR y LQR. Si bien estos tipos de controlador han sido probados [3] no existe una arquitectura accesible que permita el prototipado rápido y la validación de estos sistemas.

Los PID no son optimos citar, implementar un nuevo algoritmo es peligroso en vuelo no citar

1.3. Objetivo

Implementación de una arquitectura para validación de nuevos algoritmos de control en microcontroladores para aeronaves de ala fija por medio de simulación hardware-in-the-loop con X-Plane.

1.3.1. Objetivos específicos

1. Estudio de algoritmos de control y opciones de personalización disponibles para estos en los software de autopiloto actuales y selección de software autopiloto base.

2. Desarrollo de extensión de protocolo de comunicación de X-Plane para controladores autopiloto.

3. Implementación de distintos algoritmos de control a los ya implementados en el software y validación en simulador X-Plane como prototipo de arquitectura.

4. Establecer y documentar proceso sistemático para la implementación de algoritmos de control en la nueva arquitectura.

1.4. Condiciones de diseño

Al tratarse de un trabajo para el laboratorio de técnicas aeroespaciales, es de especial interés que la arquitectura sea accesible para alumnos y docentes en la facultad y el actual informe debe poder servir de documentación. Para estructurar la presentación del proyecto este se puede dividir en dos partes, el entorno de desarrollo que servirá como base para la investigación en X-Plane con un autopiloto, y el estudio de diseño de algoritmos de control.

1.4.1. Entorno de desarrollo

El entorno de desarrollo debe ser construido como una extensión del protocolo de comunicación con X-Plane establecido en el proyecto de ingeniería [CITA], habilitando una interfaz a microcontroladores con software autopiloto donde X-Plane emule las funciones que haría una aeronave real. X-Plane debe proporcionar lecturas de sensores simulados y responder apropiadamente a entradas de control de radio lo cual es una técnica conocida como “Hardware-in-the-loop”. Esta técnica facilita que el trabajo realizado en simulación pueda ser aplicado en aeronaves reales, puesto que para el controlador autopiloto no existirá diferencia entre estar conectado a X-Plane o funcionando en una aeronave real. Los autopilotos además se conectan a un software de estación en tierra que habilita una interfaz gráfica que permite monitorear el vuelo y crear una ruta para la misión [4], entre otras cosas. El diseñar la arquitectura alrededor de software como ArduPilot o Pixhawk permite aprovechar todas estas funciones existentes. Los sistemas de autopiloto cuentan con interfaces estandarizadas para conectar sensores y entradas de radiocontrol. Es imperativo seleccionar una base que cuente con estos estándares definidos para la posterior implementación del hardware. El autopiloto base debe ser uno de los que ya se tenga experiencia en el laboratorio como Pixhawk o ArduPilot.

1.4.2. Diseño de algoritmos de control

[ARREGLAR] matlab y simulink

Un algoritmo podrá ser validado en el software de simulación X-Plane como primera iteración, en este se podrá probar la interfaz entre el controlador de vuelo y X-Plane, la capacidad de sintonizar el algoritmo a aeronaves específicas y la realización de misiones autónomas. Esto será parte de un

proceso sistemático con instrucciones con el objetivo final de que los algoritmos puedan ser probados en aeronaves reales. El procedimiento de diseño de controladores fomentará la experimentación con base en herramientas ya utilizadas en el laboratorio como lo son Simulink o Python. Con especial enfoque en aeronaves de ala fija.

1.5. Metodología de trabajo

1.5.1. Selección de autopiloto base

A partir de las condiciones de diseño establecidas se realizará una comparación entre las soluciones de autopiloto existentes. Además se investigarán trabajos ya realizados que hayan tenido objetivos similares a los de este proyecto, en especial el de modificar e implementar algoritmos de control puesto que en mi opinión es la parte más difícil.

1.5.2. Extensión de protocolo

Con el autopiloto base seleccionado, se debe proceder a replicar las interfaces de hardware estandarizadas para simular la experiencia de una aeronave real, donde X-Plane junto al protocolo de comunicación la emularan. Los datasheets de los sensores disponibles serán utilizados para replicar las señales que estos entregan al controlador de acuerdo a su capacidad o precisión. Se estudiaran los factores adversos que pudiesen ocurrir afectando la radiofrecuencia de control debido al clima o distancias para también hacer una estimación de ellos. Todo esto se implementará en el software inoFS [5] en su programación. Para la interfaz de hardware se conectará a la Raspberry Pi Pico un bus de comunicación de acuerdo al estándar de autopiloto [6], sea uno proporcionado por el laboratorio o replicado con los insumos electrónicos que se deban comprar. Para que este objetivo pueda ser logrado no hace falta modificar el código del autopiloto, si los algoritmos PID logran funcionar con simulaciones de X-Plane es suficiente.

1.5.3. Prototipo de implementación de nuevos algoritmos

Ya sea a partir de esfuerzos previos realizados por la comunidad o el estudio del código fuente del autopiloto base se implementara otro sistema de control que se ejecute en el microcontrolador con el software de autopiloto. Al existir la extensión de protocolo probada con PID el funcionamiento del nuevo prototipo de controlador podrá ser verificado en X-Plane de la misma manera. El desarrollo del nuevo algoritmo será a partir de sistemas creados en Simulink y Python, donde el código final será exportado a C o C++ para su integración en el microcontrolador de autopiloto. El objetivo se considerará listo cuando se demuestre que el nuevo algoritmo logra cumplir tareas básicas como estabilización o el ajuste autónomo de las condiciones de vuelo a otras condiciones preestablecidas.

1.5.4. Formulación de metodología para implementación de algoritmos

Durante el desarrollo del prototipo probablemente se habrán intentado varias alternativas para lograr el funcionamiento adecuado, el último objetivo es preparar una metodología junto a documentación

que defina el camino seguido para la implementación de nuevos algoritmos en microcontroladores con software de autopiloto. Se reducirá el procedimiento a los pasos mínimos esenciales en documentación web del repositorio de inoFS, además de lo reportado en el informe de memoria de título. La metodología abarcará desde la creación de un algoritmo sencillo hasta su verificación en X-Plane.

Capítulo 2

Entorno de desarrollo

Este capítulo describe el proceso para establecer el entorno de desarrollo. Este entorno servirá de base para el trabajo posterior en los siguientes capítulos que, como está descrito en la sección de condiciones de diseño, tiene como función principal habilitar la validación del trabajo realizado por medio de la técnica “Hardware-in-the-loop”.

2.1. Selección de autopiloto base

y será provechoso si ya existen esfuerzos por agregar la capacidad de algoritmos personalizados a alguno de estos lo que reduciría trabajo. Una solución que se acerca a esta visión es la de recompilar ArduPilot junto con código exportado de Simulink para la implementación de nuevos controladores [7].

2.2. Extensión de protocolo de comunicación

Capítulo 3

Diseño de algoritmos de control

Por lo general los software de autopiloto cuentan con un solo tipo de algoritmo de control para cada lazo y aunque ofrezcan opciones para activar o desactivar ciertas asistencias [8], o el ajuste de parámetros internos [9] no tienen forma de cambiar por completo el algoritmo de control. Esto por lo general es una decisión de diseño, ya que los lazos PID han demostrado ser suficientes para los requerimientos de la plataforma y dar soporte para más alternativas es trabajo extra. Los controladores de vuelo realizan varias funciones internas de menor nivel para poder habilitar las asistencias y hacer de autopiloto, por ejemplo la recopilación de información de los sensores y posterior filtrado no es un problema trivial y en ArduPilot ha pasado por varias iteraciones [10].

3.1. Modificación de algoritmo existente

3.2. Implementación de nuevo algoritmo

Capítulo 4

Documentación

a

Capítulo 5

Conclusión

aaaa

Bibliografía

[1] Pixhawk. *Controller Diagrams*. URL: https://docs.px4.io/main/en/flight_stack/controller_diagrams.html (visitado 15-09-2023).

[2] Betaflight. *PID Tuning Guide*. URL: https://betaflight.com/docs/wiki/archive/PID-Tuning-Guide (visitado 15-09-2023).

[3] UZH Robotics y Perception Group. *Onboard State Dependent LQR for Agile Quadrotors (ICRA 2018)*. URL: https://www.youtube.com/watch?v=8OVsJNgNfa0 (visitado 15-09-2023).

[4] ArduPilot. *Choosing a Ground Station*. URL: https://ardupilot.org/copter/docs/common-choosing-a-ground-station.html (visitado 15-09-2023).

[5] Germán Quijada. *Repositorio inoFS*. URL: https://github.com/qgerman2/inoFS (visitado 15-09-2023).

[6] Pixhawk. *Autopilot Bus & Carriers*. URL: https://docs.px4.io/main/en/flight_controller/pixhawk_autopilot_bus.html (visitado 15-09-2023).

[7] ArduPilot. *Adding Custom Attitude Controller to Copter*. URL: https://ardupilot.org/dev/docs/copter-adding-custom-controller.html (visitado 15-09-2023).

[8] ArduPilot. *Plane Flight Modes*. URL: https://ardupilot.org/plane/docs/flight-modes.html (visitado 15-09-2023).

[9] ArduPilot. *Roll, Pitch and Yaw Controller Tuning*. URL: https://ardupilot.org/plane/docs/new-roll-and-pitch-tuning.html (visitado 15-09-2023).

[10] ArduPilot. *Extended Kalman Filter (EKF)*. URL: https://ardupilot.org/copter/docs/common-apm-navigation-extended-kalman-filter-overview.html (visitado 15-09-2023).

Apéndice A

Carta Gantt