Universidad Tecnológica Nacional

Facultad Regional Córdoba

Práctica Profesional Supervisada

Informe Final

Software en Tiempo Real para UAV

:

Alumno:
Malerba, Martin Iñaki Leg 63495

5 de septiembre de 2018

Índice

1. Introducción **2**

2. Planificación **2**
2.1. Parámetros de Diseño 2
2.2. Zephyr 3

3. Componentes **3**
3.1. Procesamiento 3
3.2. Sensores 4
3.3. Comunicación 4
3.4. Telemetría 4
3.5. Actuadores 5

4. Conclusiones **5**

1. Introducción

La realización del trabajo se basa en el desarrollo de una plataforma de arquitectura abierta, para la implementación en investigación y desarrollo de algoritmos de control y estimación de posición mediante el uso de sensores de bajo costo.

Enmarcado sobre un contexto de investigación académica, realizamos en conjunto con el Centro de Investigación en Informática para la Ingeniería (CIII) el proyecto de un cuadricóptero de dimensiones reducidas, enfocado en la disminución de los costos empleados en la fabricación de los cuadricópteros convencionales.

El rol que este trabajo cubre fue la elección, implementación y desarrollo del software que controlará al vehículo y donde se implementarán las susodichas investigaciones.

2. Planificación

Para el correcto control de un vehículo autónomo es necesario disponer de un sistema capás de obtener información del entorno, predecir cambios y actuar acorde al estado en el que se encuentre. El conjunto debe ser capás de medir y procesar una gran cantidad de variables del ambiente a una frecuencia conocida, que garantice un bucle de realimentación correcto.

Para garantizar el comportamiento adecuado en el tiempo requerido se necesita que el sistema sea predecible. Por este motivo, se utiliza un sistema operativo en tiempo real. Los sistemas en tiempo real, o RTOS, se caracterizan por presentar requisitos especiales en cinco áreas generales:

- Determinismo
- Sensibilidad
- Control del usuario
- Fiabilidad
- Tolerancia a los fallos

La mayoría de los RTOS son utilizados cuando existen requerimientos de tiempo muy rígidos en las operaciones o en el flujo de datos, generalmente son requeridos como sistemas de control en una aplicación dedicada.

La eficiencia de los RTOS no solo depende de la exactitud de los resultados de cómputo, sino también del momento en que los entrega. La predictibilidad es su característica principal de este tipo de sistemas.

Este tipo de sistemas se caracterizan por tener que producir una salida, como respuesta a una entrada, en un tiempo determinado. El intervalo de tiempo que se presenta entre la entrada y la salida debe ser muy pequeño para que la respuesta temporal del sistema sea aceptable.

2.1. Parámetros de Diseño

Para el diseño de un RTOS es necesario definir las metas del sistema para luego tener una medición objetiva de los resultados obtenidos

> 3Se identifican todas las tareas que se tienen que realizar y también se identifican las restricciones temporales que se pretenden cumplir. Posteriormente se codifican los programas que ejecutarán las tareas. Por último se pasa a medir el tiempo de cómputo de cada tarea y se realiza un análisis de efectividad.

Este análisis consiste en aplicar unas pruebas al conjunto de tareas de tal forma que si éstas pasan el test entonces se puede garantizar que ninguna tarea perderá su plazo de ejecución. De lo contrario si no pasan el test se tiene que volver a comenzar desde el principio, es decir, comenzar de nuevo, utilizando otro procesador más potente o utilizando otros algoritmos para implementar las tareas.

2.2. Zephyr

Hay una gran cantidad de RTOS disponibles para el uso. Todos presentan diferentes complejidades y ventajas, pero para la realización de este proyecto se eligió el sistema llamado Zephyr.

Zephyr es un proyecto creado por la Linux Foundation, en un esfuerzo de unir fuerzas con los mayores líderes de la industria para la generación de un kernel con un bajo consumo de recursos, orientado a la escalabilidad y a dispostivos de recursos acotados, a través de múltiples arquitecturas.

El kernel de Zephyr deriva del Microkernel VxWorks de Wind River, utilizado por más de 20 años en aplicaciones de procesamiento de señales en la industria aeroespacial, militar, de comunicaciones y procesamiento de imágenes, siendo uno de sus mayores y mas recientes éxitos el aterrizaje de Philae en el cometa Churyumov–Gerasimenko y su acompañante Rosetta.

3. Componentes

El sistema de manejo de un vehículo aéreo está compuesto por múltiple módulos:

3.1. Procesamiento

Como unidad principal de procesamiento se utilizó un microcontrolador de STMicroelectronics denominado bajo el código STM32F401. Este SoC ofrece el mejor balance entre consumo de energía y capacidad de procesamiento, conteniendo un Cortex M con unidad de punto flotante a 84 MHz.

A 84 MHz, el STM32F401 logra alcanzar 105 DMIPS ejecutando código desde la memoria flash, permitiendo el uso de la unidad de punto flotante incrementando el rango de aplicaciones. En este estado, mediante su procesador de 90 nm, el sistema de escalado de frecuencias permite un consumo de 128 μA/MHz.

A pesar del tamaño reducido, el STM32F401 provee las siguientes características:

- 96 KBytes de SRAM
- 3 USARTs a 10.5 Mbit/s
- 4 SPI a 42 Mbit/s
- 3 I^2C
- 1 USB 2.0 OTG
- 2 I^2S de 32-bit/192KHz
- 12-bit ADC hasta 2.4 MSPS
- 10 timers, 16- y 32-bit, a 84 MHz

3.2. Sensores

Debido a que el vehículo no posee visión del entorno, es necesario fusionar las medidas obtenidas por distintas clases de sensores en un esfuerzo de calcular y predecir la actitud del vehículo. Para este fin, el UAV posee sensores capaces de medir variables tales como:

- Aceleración
- Giro
- Campo Magnético
- Presión Atmosférica
- Distancia al Suelo

Estos sensores deben ser consultados a la mayor frecuencia posible, ya que al ser un sistema discreto esto nos permite mejor resolución y respuesta al control.

3.3. Comunicación

Como el sistema debe ser capás de recibir órdenes y comunicar su estado, es necesario que sea capás de comunicarse con una estación terrena. Para ésto, utilizamos la prestación incorporada en el microprocesador de comunicación vía IEEE 802.11, comercialmente denominada WiFi. Sobre ésta capa física, se implementó una comunicación mediante el protocolo TCP, el cual nos permite una fiabilidad en la conexión y nos garantiza que los comandos lleguen al vehículo.

3.4. Telemetría

Para conocer el estado del vehículo es necesario determinar un protocolo de telemetría, en donde todas las métricas del sistema y sus sensores son enviados a través de un protocolo inalámbrico. Esto permite el conocimiento del estado en todo momento, como también el análisis de las métricas posteriormente, ya que son almacenadas en un medio persistente en la estación de tierra.

La disposición de telemetría en tiempo real permite al operador la toma de decisiones y envío de órdenes al vehículo mientras se lo opera, sea esta operación realizada en forma manual o mediante automatismos implementados en el vehículo.

Figura 1: Hexacóptero

Figura 2: Octacóptero

3.5. Actuadores

Como actuador para la modificación de la actitud del vehículo se utilizan 4 motores de corriente continua, ubicados en las esquinas del vehículo. Es común la utilización de una mayor cantidad de motores, como se puede ver en los hexacópteros, con 6 motores, o los octacópteros, con 8. La principal ventaja de 4 motores por sobre 6 u 8 se basa en el costo de manufactura y en la facilidad de maniobra, ya que una mayor cantidad de motores proporciona empuje éxtra pero complica y alentiza la reacción del mismo.

4. Conclusiones

El utilizar un sistema operativo que se encuentra en desarrollo trae consigo problemas como son los cambios repentinos de la arquitectura, o problemas de compatibilidad con sensores o protocolos. Esto me permitió aprender mucho sobre las formas de desarrollo en grupo y de contribución a proyectos de grandes dimensiones.

Los problemas que encontré al implementar el sistema en mi plataforma me permitió realizar mi primer contribución al Software Libre, mediante la correción de un driver de i2c para el chip utilizado.

Por otro lado, el desarrollo de un vehículo volador le agrega un grado de complejidad muy interesante al proyecto, ya que todas las pruebas deben ser pensadas y desarrolladas con mayor cuidado y planificación, ya que un error puede generar la destrucción de los prototipos.