Universidad Zaragoza
1542

Trabajo Fin de Grado

Navegación de un quadcopter basada en información de un sensor de rango láser

Autor

Lorenzo Montano Oliván

Director

José Luis Villarroel Salcedo

Escuela de Ingeniería y Arquitectura
Zaragoza,Febrero de 2016

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Lorenzo Montano Oliván,

con nº de DNI 77133640-N en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Navegación de un quadcopter basada en información de un sensor de rango láser,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 5 de Febrero de 2016

Fdo: Lorenzo Montano Oliván

Resumen

Los drones o UAV son una tecnología en auge en los últimos años. Esto se debe al gran número de aplicaciones que se están desarrollando, sobre todo en escenarios exteriores, donde se dispone información GPS para su localización. En escenarios interiores, al no disponer de información GPS, la localización y la navegación tiene que hacer uso de otros sensores para dicha navegación y localización.

En este trabajo fin de grado se va a diseñar e implementar un sistema de navegación en espacios cerrados (tipo túnel,galería...) para un dron F450. Este sistema debe ser capaz de modificar la orientación del dron de forma autónoma, para desplazarse sin colisión por ese tipo de escenarios. La información del entorno se adquiere a partir un escáner de rango láser. Se trata de un primer paso en el control autónomo del dron en interior, para lo cual únicamente se controlará el ángulo Yaw (orientación en la dirección del movimiento). Los otros grados de libertad se controlarán manualmente desde el mando disponible. Con ello se desacopla el problema complejo de control de este tipo de vehículos, centrando el estudio del control en este grado de libertad. De esta forma se consigue el objetivo fundamental de seguimiento semiautónomo de pasillos o tuberías, paso por puertas que comunican varias estancias, e intersecciones entre pasillos.

Para conseguir estos objetivos se han realizado simulaciones haciendo uso de la herramienta Matlab para poder comprobar el correcto funcionamiento de los métodos de navegación implementados antes de probarlos en el modelo real.

Se ha diseñado la arquitectura hardware y tiempo-real del sistema, integrando los sensores, el sistema de comunicaciones, y el procesador F28335. Este procesador es un microcontrolador de gama media de Texas Instruments (150MHz,32 bits, CPU en coma flotante). Las técnivas de percepción y navegación se han realizado como un sistema en tiempo real basado en tareas, implementado en el sistema operativo SYS/BIOS.

Se han desarrollado dos métodos de navegación. Tras su evaluación e implementación se ha optado por uno de ellos para realizar las pruebas de vuelo finales. Se ha conseguido realizar vuelos en espacios cerrados con obstáculos que el quadcopter ha esquivado satisfactoriamente, alcanzando los objetivos planteados.

Índice general

Índice de figuras

1. Introducción

1.1. Contexto y estado del arte

Un dron es un vehículo aéreo no tripulado reutilizable(UAV)[1]. Este es capaz de mantener un nivel de vuelo controlado mediante la fuerza ejercida normalmente por hélices propulsadas por motores de explosión o reacción.

Los primeros drones surgieron durante la primera guerra mundial y se utilizaron en la segunda para entrenar a los operarios que manejaban los cañones antiaéreos. La única característica que diferencia a los drones de los misiles con control remoto, es que los primeros son reutilizables. Es por esta razón por la que los misiles no son considerados UAV.

Existe una amplia variedad de formas, tamaños, configuraciones y características en el diseño de los drones. Se pueden encontrar desde helicópteros, drones con forma de avión o multirotor.

Históricamente los drones han sido únicamente de uso militar, utilizados tanto para misiones de ataque como de reconocimiento. En la actualidad se están desarrollando numerosas aplicaciones de uso civil, tanto de tipo profesional tales como búsqueda de personas desaparecidas, cartografía aérea como por ejemplo reconstrucción 3D de un terreno 1.1(c), prevención de incendios, revisión de tuberías 1.1(b), vigilancia,agricultura1.1(a)... o usos más lúdicos como realizar filmaciones o uso recreativo de los drones comerciales.

(a) Dron con aplicación agricola (b) Dron revisando una tubería (c) Reconstrucción 3D de un edificio con un dron

Figura 1.1: Diferentes aplicaciones de drones

[1]UAV:Unmanned Aerial Vehicle

Para realizar el control autónomo de los drones es necesario desarrollar algoritmos de planificación y navegación reactiva, que utilizan información del entorno obtenida mediante diferentes sensores externos conectados a una unidad de control capaz de procesar toda la información captada del entorno. En la actualidad los drones cuentan con sensores tales como: GPS, IMU, altímetro, y sensores de rango, para obtener la ubicación en la que estos se encuentran. Para realizar una localización en interior donde no hay información GPS, es necesario un cierto reconocimiento del entorno para lo cual se utilizan e integran sensores tales como escáner láser, RGBD (Kinect), y cámaras.

En la actualidad la navegación en interiores, sin información GPS, se está desarrollando principalmente en laboratorios de investigación. Para una navegación autónoma completa es necesario aplicar y desarrollar complejos algoritmos de construcción de mapas y localización a partir de la información sensorial. En este proyecto se da un paso más hacia ese objetivo en esta línea de investigación del Grupo de Robótica, desarrollando un algoritmo de navegación implementable en el procesador de bajo coste embarcado en el dron. Con esta técnica se controlará autónomamente uno de los grados de libertad del quadrotor, la orientación o ángulo Yaw, a partir únicamente de la información de un escáner láser embarcado. Ello permitirá guiarlo en interiores, manteniendo un nivel de seguridad frente a colisiones con obstáculos y una orientación en la dirección principal del espacio libre en el escenario de navegación. Como aplicaciones inmediatas cabe mencionar la navegación en pasillos dentro de edificios, o en túneles o tuberías para inspección.

Este proyecto se lleva a cabo dentro del grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza, uno de los grupos de investigación del Instituto Universitario de Investigación en Ingeniería de Aragón (I3A) considerado Grupo de Investigación por el Gobierno de Aragón. Dicho grupo tiene las siguientes lineas de trabajo:

- Localización y Mapeado Simultaneo.
- Visión por Computador y Percepción.
- Comunicaciones y redes ad-hoc.
- Exoesqueletos y procesamiento de bioseñales.
- Aprendizaje: en robótica, optimización Bayesiana, interfaces cerebro-ordenador...
- Robótica Móvil. Planificación y navegación.

Este proyecto se centra en la línea de “Robótica móvil, planificación y navegación”. Se van a desarrollar diferentes proyectos en los que la intervención de drones autónomos es una pieza central de la investigación. En particular se va a trabajar en la inspección de túneles y tuberías, en la detección de sustancias tóxicas y cálculo de parámetros medioambientales, por poner algunos ejemplos. En todos ellos es fundamental dotar al dron de la capacidad de navegar orientándose adecuadamente y evitando los obstáculos que aparezca en su camino, bien para

la navegación completamente autónoma o para dotar de un mecanismo de autoprotección si es manipulado remotamente.

Se han realizado varios TFG en esta linea de investigación como son: La creación de un autopiloto para poder tener el control de todas las variables del dron o un generador de trayectorias, para poder enviar una serie de consignas de posición que el dron deberá alcanzar. Este proyecto continúa los anteriores, añadiendo nuevas capacidades autónomas.

1.2. Objetivos

El objetivo principal de este proyecto es conseguir implementar un sistema de navegación para espacios cerrados(tipo túnel, galería,...), capaz de controlar un quadcopter FlameWheel F450 a partir de la información tomada por un sensor láser y un autopiloto Naza-M Lite.

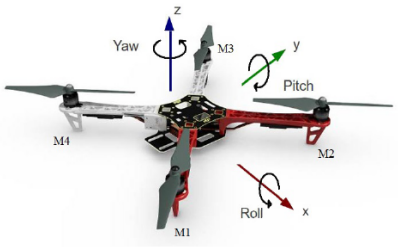


Figura 1.2: FlameWheel F450

Se busca crear un sistema de navegación que otorgue al quadcopter [1] la capacidad de modificar su orientación de forma autónoma. El proyecto se va a centrar en el cálculo de la orientación del quadcopter a partir de la información de los obstáculos del entorno proporcionada por el escáner láser. De los 6 grados de libertad del dron (x,y,z,roll,pitch,yaw), los 5 primeros serán controlados remotamente por el piloto o autopiloto comercial de abordo, y se actuará sobre "yaw"que es el que proporciona la orientación del vehículo. De esta manera se simplifica y desacopla el problema de control del movimiento, ya de por sí complejo.

Para desarrollar estos objetivos se cuenta con:

- Un Sensor de rango láser HOKUYO-04LX
- Un microcontrolador F28335
- Software de desarrollo integrado de aplicaciones Code Composer
- Sistema operativo de tiempo real SYS/BIOS
- Matlab R2015a

1.3. Organización de la memoria

El trabajo se ha dividido en 5 capítulos:

- **Hardware y Software:** Se ha utilizado el hardware y software proporcionado por el grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza. En este capítulo se van a explicar las características más relevantes del hardware que se ha integrado en el quadcopter y el software utilizado para la programación de este.
- **Arquitectura del Hardware:** En este capítulo se va a dar una visión global de la estructura hardware integrada en el sistema y se van a explicar las características básicas de el hardware que ha sido necesario añadir al diseño.
- **Navegación con evitación de obstáculos:** En este capítulo, se van a analizar los diferentes métodos de navegación que se han probado en el quadcopter. También se hará una pequeña comparación entre dos de los métodos propuestos y se explicará como se ha llevado a acabo el cálculo de el ángulo de referencia que debe llevar el quadcopter.
- **Implementación software del sistema:** Aquí es donde se explica como se ha llevado a acabo la integración del hardware en el procesador haciendo uso de las herramientas que nos ofrece el sistema de tiempo real que se ha utilizado.
- **Pruebas de vuelo:** Se han realizado numerosas pruebas para poner a punto el sensor para que sea capaz de orientarse correctamente. En este capítulo se explican algunas de las más importantes.
- **Conclusiones:** En este capítulo se recogen las conclusiones obtenidas con este TFG.

2. Descripción del hardware y software

2.1. Hardware

Para llevar a cabo el control del quadcopter se va a hacer uso de los siguientes componentes: Un sensor láser para calcular algoritmos que permitan navegar al quadcopter en espacio libre, un Autopiloto en el que van integradas las ecuaciones de movimiento para controlar el quadcopter y un microcontrolador en el cual se implementará el software para llevar a cabo el control. A continuación se va a dar una pequeña explicación de cada uno de estos dispositivos.

2.1.1. Sensor Láser HOKUYO-04LX

El HOKUYO-04LX [2] es un sensor láser de tamaño reducido y preciso, adecuado para aplicaciones robóticas. Tiene un rango de medida de 20mm a 4m, en un arco de 240°. El rango de medida es suficiente para navegar por un entorno cerrado, que es por el cual se va a tener que mover el quadcopter.

Figura 2.1: Sensor Láser

2.1.1.1. Especificaciones técnicas del sensor láser

En esta sección se presenta la información básica del sensor láser. Estas especificaciones son importantes para la decodificación de los datos enviados por el mismo.

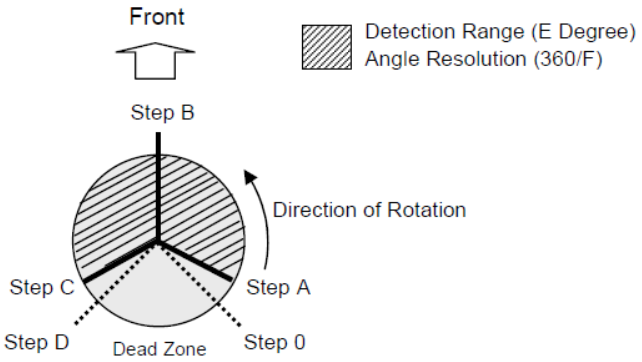


Figura 2.2: Rango de detección

En la Figura 2.2 podemos ver que el sensor va captando las medidas en sentido antihorario visto desde arriba. En el caso del Hokuyo URG-04LX el "Step "que vemos en la imagen 2.2, es el angulo máximo al que el sensor llega a tomar medidas, y el "Step A"es el punto inicial del escaneo. La resolución angular del láser se obtiene dividiendo 240°(Rango de escaneo del sensor) entre 682(Número de medidas tomadas). Realizando esta operación, se obtiene una precisión de 0.36 $\frac{\text{o}}{medida}$

2.1.1.2. Codificación de los datos

Los datos enviados por el sensor, vienen codificados para reducir el tiempo de transmisión entre el procesador y este. Estos datos deben ser decodificados para poder procesarlos. Hay tres técnicas de codificación distintas para enviar los datos que el usuario puede seleccionar en función del volumen de datos que se vayan a tratar. Estas son: codificación de dos, tres, y cuatro caracteres. En nuestro caso utilizaremos la codificación de dos caracteres, ya que al ser una aplicación que debe funcionar en tiempo real y la cantidad de operaciones que el procesador debe realizar es relativamente elevada, interesa que el tiempo de respuesta sea el menor posible, para un funcionamiento óptimo de la aplicación.

2.1.2. Autopiloto Naza-M Lite

Este dispositivo es un controlador en el que están integradas las ecuaciones de movimiento del quadcopter, a partir de las cuales se calculan las señales PWM para que cada motor funcione

con la potencia necesaria para alcanzar las consignas de ángulo impuestas por el usuario. Esto último puede hacerse a través de un mando que controla la potencia y los ángulos Roll, Pitch, Yaw, o en el caso de la aplicación que se va a ha implementar para la navegación por espacio cerrado, únicamente se actuará sobre el ángulo Yaw.

2.1.3. Microcontrolador TMS320F28335

El TMS320F28335 [3] es un microcontrolador de Texas Instruments que trabaja a una frecuencia de 150 MHz con una CPU de 32 bit de coma flotante. Este procesador es idóneo para aplicaciones en tiempo real. Tiene una frecuencia de trabajo suficiente para realizar las operaciones que se quieren llevar a cabo en esta aplicación. Ademas incluye 3 módulos para conexión SCI, que son los necesarios para conectar el sensor láser, el IMU y los XBee.

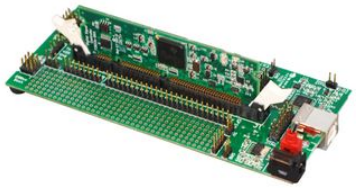


Figura 2.3: Microcontrolador F28335

2.1.4. Estructura mecánica y motores del quadcopter

El quadcopter está formado por una estructura de plástico sobre la que se sitúa todo el hardware descrito anteriormente así como los cuatro motores con sus respectivos ESC(Electronic Speed Controller). Esta estructura está formada por dos barras rojas y dos blancas. Las rojas indican la parte delantera del quadcopter.[4]

Los motores son trifásicos síncronos de corriente continua capaces de desarrollar una potencia máxima de 370W. Para ello es necesario el uso de los ESC, cuya función es convertir la señal PWM enviada por el microcontrolador en tres fases que controlan la velocidad del motor. 1.2

2.1.5. Batería y regulador de tensión

La batería utilizada es el modelo Turnigy 5.0. Dicha batería es de 4 celdas y tiene una carga mínima de 14,8 voltios. Su capacidad es de 5000 mAh, lo que conlleva un vuelo medio entre 10 y 12 minutos. Pesa 552 g y tiene unas dimensiones de 149 x 49 x 33 mm. También se utiliza un regulador de tensión de 5V para alimentar el microcontrolador y después este alimenta los sensores a 3,3 V y 5V.

2.1.6. Multiplexor

El multiplexor 4-Channel RC Servo Multiplexer, esta diseñado para este tipo de aplicaciones. Es capaz de multiplexar dos entradas de 4 canales a través de un 5º canal y obtener la salida que se seleccione con este último.

Este dispositivo ofrece la posibilidad de realizar pruebas de vuelo con el código implementado en el microcontrolador, con la seguridad de que si el sistema falla, se puede conmutar rápidamente a modo autopiloto con una palanca situada en el mando.

2.2. Software

La implementación del software se va ha realizar en el entorno de programación Code Composer de Texas Instruments, mediante el uso de SYS/BIOS, un sistema operativo para controlar las distintas tareas que el procesador debe realizar en tiempo real. Previamente todo el software ha sido simulado en MATLAB, para comprobar de una forma más visual el funcionamiento de los métodos de navegación utilizados. El codigo ha sido implementado en coma flotante en C++.

2.2.1. Code Composer Studio

Code Composer [5] es un entorno de desarrollo integrado de aplicaciones creado por Texas Instruments. En el se puede encontrar una serie de herramientas para analizar paso a paso el código implementado. Tiene editor, *debugger*, compilador y *linker*

Se puede programar en el microprocesadores tales como C2000, TMS570, Sitara, C55x o C28x. Se ha utilizado la versión 5.5.

2.2.2. SYS/BIOS

SYS/BIOS es un núcleo de tiempo real de Texas Instruments para DSP's,ARM y microcontroladores. Permite una planificación de tareas en el procesador basdada en prioridades fijas y herencia de prioridad. Este sistema operativo contiene herramientas para facilitar la gestión de las tareas y analizar de una forma sencilla la conmutación de las distintas tareas en el procesador. Con ella se pueden visualizar: tiempos de cómputo, carga del procesador, graficos de ejecución...Los elementos que forman SYS/BIOS son:

- Interrupciones hardware(Hwi)
- Interrupciones software(Swi)

- Tareas(Task)
  - Periódicas
  - Esporádicas
- Semáforos

2.2.3. MATLAB

Se ha utilizado Matlab [6] R2010a y R2015a para simular todos los métodos matemáticos utilizados para calcular la trayectoria que debe seguir el quadcopter. Matlab es una herramienta muy completa y útil que permite simular de forma visual los métodos utilizados para la navegación del quadcopter y comprobar si éstos funcionan correctamente antes de implementarlos en C++ en el procesador.

3. Arquitectura del Sistema

Además del hardware del que se partía inicialmente, ha sido necesario añadir otros dispositivos e implementar *drivers* software para poder cumplir los objetivos de este TFG.

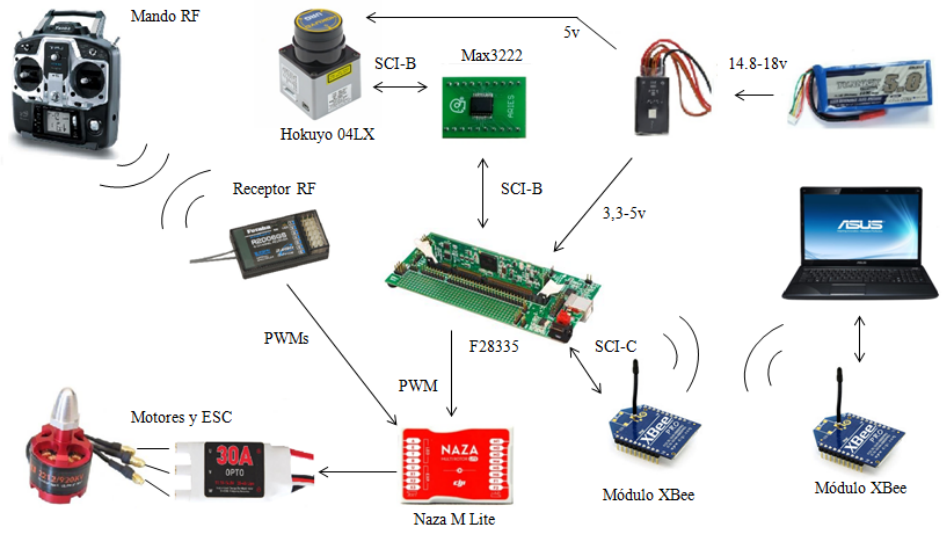


Figura 3.1: Diagrama de bloques del hardware utilizado

Para poder conectar el sensor láser al microcontrolador F28335, ha sido necesario hacer uso de un conversor de tensión. Esto es debido a que el láser transmite las tramas mediante el protocolo RS-232 [1], y el microncontrolador mediante SCI [2]. La RS-232, funciona con señales bipolares de +/-6V y la SCI del microcontrolador funciona mediante señales de entre 0 y 3,3v. Por esta razón es necesario hacer un dispositivo capaz de adaptar ambas señales para que ambos dispositivos puedan comunicarse entre ellos. Ademas de esto, es necesario crear un driver software para procesar la información enviada por el sensor láser.

Debido a que el quadcopter no puede estar conectado con cables a la estación de control,

[1] Recommended Standard 232, es un interfaz que designa una norma para el intercambio de una serie de datos binarios

[2] Serial Comunication Interface

es necesario integrar un módulo de comunicaciones inalámbrica para poder visualizar en un dispositivo externo información de importancia que permita comprobar el correcto funcionamiento de los métodos de navegación. En este caso, se va a utilizar un módulo de comunicación XBee. Uno de los dispositivos se conecta directamente al microcontrolador F28335 a través de uno de los puertos SCI integrados en este y el otro, se conecta directamente al dispositivo externo en el que se visualizará la información mediante un USB. El dispositivo va a enviar la telemetría del quadcopter para poder visualizar en todo momento los obstáculos que se están detectando, y la dirección de movimiento a seguir. Con el fin de facilitar el envío de tramas desde el ordenador al microcontrolador y la visualización de las tramas recibidas desde el microcontrolador, se ha hecho uso de la herramienta "GUIDE"de Matlab para crear una interfaz gráfica y poder visualizar de forma clara la telemetría del quadcopter.

Tambíen será necesario generar un PWM para controlar el ángulo Yaw, que es el que define la orientación del quadcopter. Para ello será necesario utilizar el módulo *EPWM* del F28335. Esta señal PWM será enviada al autopiloto Naza M, el cuál se encargará de enviarla a los ESC para darles la velocidad necesaria a los motores para alcanzar la consigna impuesta por el PWM.

El resto de variables de estado del sistema serán controladas con un mando de radiofrecuencia, el cual transmite señales a un receptor integrado en el quadcopter. Este receptor está conectado directamente al autopiloto cuya función es procesar las señales enviadas por el receptor.

El software creado para el desarrollo de los métodos de navegación ha sido creado en Matlab para poder realizar simulaciones antes de probarlo en el modelo real. Una vez realizadas las simulaciones, ha sido necesario implementar todo el código en C++ para poder integrarlo en el microcontrolador F28335 que será el encargado de gestionar todos los dispositivos hardware integrados en el sistema, y de realizar los cálculos necesarios para obtener la dirección de movimiento del quadcopter con los métodos de navegación que se explicarán en el siguiente capítulo.

3.1. Descripción del hardware añadido

En esta sección se da información básica de los dispositivos hardware que ha sido necesario añadir durante la realización de este TFG para conseguir que el sistema funcione correctamente.

3.2. Modulos de comunicación XBEE

El módulo de comunicación XBee [7] ha sido proporcionado por el grupo de Robótica, Percepción y Tiempo Real de la Universidad de Zaragoza.

Este dispositivo tiene una distancia máxima de transmisión de datos de 550m en interiores

o núcleos urbanos y de 40km en campo abierto. Su velocidad de transmisión es de 24Kbits por segundo, lo que limita mucho la separación mínima entre las tramas de telemetría que se quieren enviar debido a la extensión de estas.

3.3. Conversor de tensión MAX3222

La señal de salida del puerto SCI del microcontrolador que es enviada al láser es de 3,3 V, así que es necesario utilizar un conversor de tensión para obtener una señal bipolar entre +/- 6V, que es la requerida por el estándar RS-232 , haciendo de esta forma posible la comunicación entre microcontrolador y sensor láser.

Este dispositivo [8] cuenta con dos canales de recepción y dos de envío, que da la posibilidad de conectar otra RS-232 a los puertos SCI.

4. Navegación con evitación de obstáculos

Para llevar a cabo el cálculo de la dirección de movimiento del quadcopter, es decir, el control del ángulo Yaw, se han estudiado tres métodos. El primero de todos es el método del eje principal de inercia, con el se va a calcular la dirección principal de movimiento del quadcopter, en campo abierto (sin obstáculos), o cuando hay obstáculos laterales (por ejemplo paredes) que permiten centrar el movimiento en el eje principal. Este método se ha elegido por su simplicidad de cálculo, importante para una implementación en tiempo real. Este método no es suficiente para navegar de forma autónoma, ya que es necesario dotar al quadcopter de técnicas para esquivar obstáculos que se interpongan en la dirección calculada con este método. Para ello se han implementado los métodos de campos de potencial, y una simplificación del método ND [9].

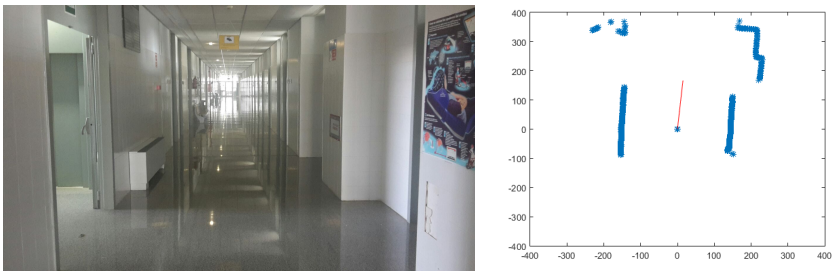
Existen muchos métodos de navegación para la evitación de obstáculos; se han seleccionado inicialmente éstos por se de los más utilizados en este tipo de navegación. Sin embargo, como se explica más adelante, su aplicación completa exige una serie de cálculos que no es posible implementar en el procesador embarcado, debido a las restricciones de capacidad de memoria y de cálculo del mismo. Por ello se ha simplificado su implementación para cumplir estos requisitos y adaptar los métodos a las restricciones hardware. Evidentemente ello tiene el inconveniente de no contemplar todas las posibilidades de navegación segura que proporcionan los métodos completos, pero se ha conseguido un funcionamiento razonable para el tipo de entornos no muy densos considerados en este proyecto.

4.1. Dirección principal de movimiento

En esta sección se va a explicar como se ha llevado a cabo el calculo de la dirección principal que debe llevar el quadcopter. Para ello se ha utilizado el eje principal de inercia calculado a partir de las medidas obtenidas por el sensor láser.

4.1.1. Ejes principales de inercia

Los ejes principales de inercia o ejes de simetría, son las direcciones representadas por los vectores propios del tensor de inercia. Estos vectores, definen los ejes en torno a los cuales



(a) Entorno real en el que debe volar

(b) Obstáculos detectados por el escáner láser (azul) y dirección principal calculada (rojo).

Figura 4.1: Entorno y dirección principal de movimiento

se genera el máximo y mínimo momento de inercia. Es decir estos ejes definen la recta que minimiza y maximiza la distancia a todos los puntos obtenidos con el sensor. En nuestro caso, elegiremos el eje de menor inercia, ya que el quadcopter debe estar siempre lo más alejado posible de los obstáculos detectados por el sensor. La Figura 4.1 representa un escenario real y la dirección principal calculada.

4.1.2. Cálculo del momento de inercia

El momento de inercia se define como la suma del cuadrado de las distancias tomadas por el sensor láser al eje de inercia. La masa es tomada como la unidad, ya que no se trata de un cuerpo físico con masa, sino que se trata de una figura virtual formada por los puntos del láser.

$$J = \sum_{i=0}^{n} m r_i^2 \tag{4.1}$$

Para llevar a cabo el calculo de los ejes principales de inercia, se ha utilizado la matriz tensor de inercia. Para ello es necesario calcular cada uno de los términos que componen la matriz.

$$I = \begin{pmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{pmatrix} \tag{4.2}$$

Los elementos I_{ii}, $i = 1, 2$ de la diagonal, reciben el nombre de momento de inercia respecto al eje.

$$I_{xx} = \sum_{i=0}^{n} y_i^2 \tag{4.3}$$

$$I_{yy} = \sum_{i=0}^{n} x_i^2 \tag{4.4}$$

Los otros dos términos son los productos de inercia según los mismos ejes.

$$I_{xy} = I_{yx} = \sum_{i=0}^{n} -x_i y_i \tag{4.5}$$

Una vez calculado el tensor de inercia de los puntos obtenidos con el sensor láser, calculando los vectores propios de esta matriz, se obtienen los ejes de simetría o ejes de inercia.

4.1.3. Vectores propios

Los vectores propios se obtienen a partir de los valores propios de la matriz de inercia. Como se ha explicado anteriormente, el objetivo de estos vectores es obtener el eje de menor inercia, el cual marca la dirección principal de movimiento. Este eje viene asociado al menor valor propio.

4.1.3.1. Cálculo de valores propios y vectores propios

La herramienta utilizada para encontrar valores propios de matrices cuadradas es el polinomio característico: decir que λ es un valor propio de A es equivalente a decir que el sistema de ecuaciones lineales A v = λ v $\rightarrow$ Av - λv = 0 (factorizando por v queda) (A - λI) v = 0 (donde I es la matriz identidad) tiene una solución no nula v (un vector propio), y de esta forma es equivalente al determinante:

$$\det(A - \lambda I) = 0 \tag{4.6}$$

Una vez que se conocen los valores propios λ, los vectores propios se pueden hallar resolviendo el sistema de ecuaciones homogéneo:

$$(A - \lambda I)\mathbf{v} = 0 \tag{4.7}$$

4.2. Detección de obstáculos

Una vez obtenida la dirección principal que determinará la consigna de orientación a alcanzar por el quadrotor, es necesario definir métodos para esquivar obstáculos que se interpongan

en la dirección de movimiento calculada con el método inercial. Para llevar a cabo dicha función han sido probados los dos siguientes métodos: Campos de potencial y ND.[10]

4.2.1. Campos de potencial

El método de campos de potencial, consiste en suponer que se está navegando por un campo de fuerzas virtual en el que los obstáculos generan una fuerza repulsiva y el objetivo genera una fuerza atractiva.[11]

4.2.1.1. Cálculo de potenciales

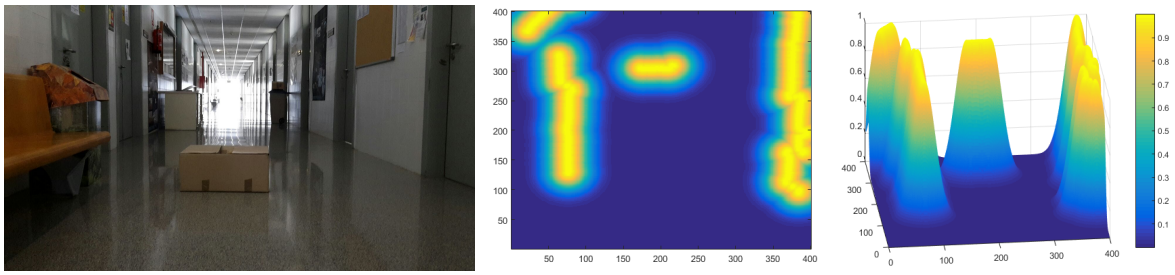
Para crear el campo de fuerzas nombrado anteriormente, es necesario ponderar de alguna forma la influencia que van a tener en el mapa los obstáculos detectados por el sensor. Para ello se ha utilizado la función gaussiana. Intuitivamente con esta función representamos que en el obstáculo y en sus proximidades existe una fuerza virtual repulsiva, que va decreciendo conforme se aleja el punto considerado del obstáculo. Se pueden utilizar diferentes funciones para esta representación, la gaussiana es una de ellas.

$$f(x,y) = A \exp\left(-\left(\frac{(x-x_o)^2}{2\sigma_x^2} + \frac{(y-y_o)^2}{2\sigma_y^2}\right)\right). \tag{4.8}$$

- x,y son las variables de la función, los puntos de evaluación de la función.
- x_o,y_o son las coordenadas del centro de la gaussiana, es decir, las coordenadas de los obstáculos detectados por el láser.
- σ_x,σ_y son las desviaciones típicas en cada uno de los ejes.
- A es un coeficiente para ajustar la amplitud de la función gaussiana.

Potencial repulsivo

Vamos a realizar una representación discretizada en vez de continua del escenario para simplificar los cálculos. Por ello trabajamos con una retícula o "grid"en la que cada celda tendrá asignado un potencial. Cada uno de los puntos detectados por el sensor, tiene asignada una función gaussiana para saber de esta forma la influencia que tiene en el campo de potenciales. El campo de potenciales final, se genera sumando cada una de estas funciones nombradas anteriormente en una única función, que es la que va a definir la influencia de todos los obstáculos detectados. Esta función está proyectada en la retícula, como se representa en la Figura 4.2

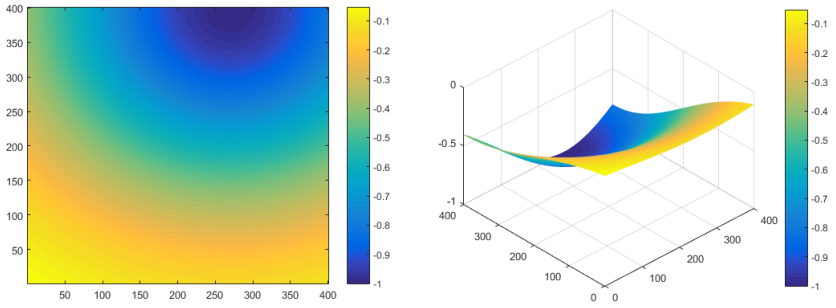


(a) Entorno real (b) Mapa de potencial repulsivo 2D (c) Mapa de potencial repulsivo 3D

Figura 4.2: Mapas de potencial repulsivo

Potencial atractivo

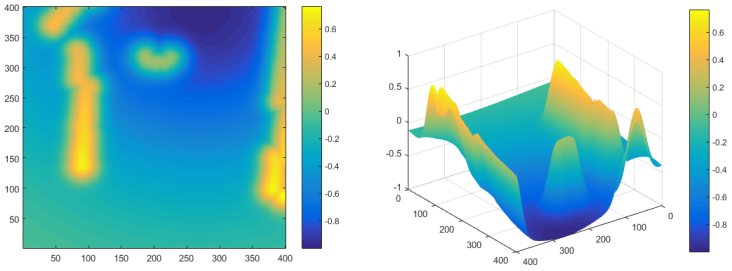
El objetivo a alcanzar, definido como una posición en el escenario sobre la dirección principal de movimiento calculada anteriormente, define un potencial atractivo virtual, que "atrae"al quadrotor hacia ese punto. Esta función va a generar una pendiente negativa hacia el objetivo, hacia el que se "deslizará"el dron.



(a) Mapa de potencial atractivo 2D (b) Mapa de potencial atractivo 3D

Figura 4.3: Mapas de potencial atractivo

Una vez calculado los potenciales atractivo y repulsivo, uniendo ambos mapas, se obtiene el mapa de potencial total, a partir de el cual se puede obtener la dirección instantánea en cada celda de la retícula por la que debe navegar el quadcopter siguiendo el camino de mayor gradiente negativo.



(a) Mapa de potencial 2D (b) Mapa de potencial 3D

Figura 4.4: Mapas de potencial atractivo y repulsivo

4.2.1.2. Gradiente del campo de potencial

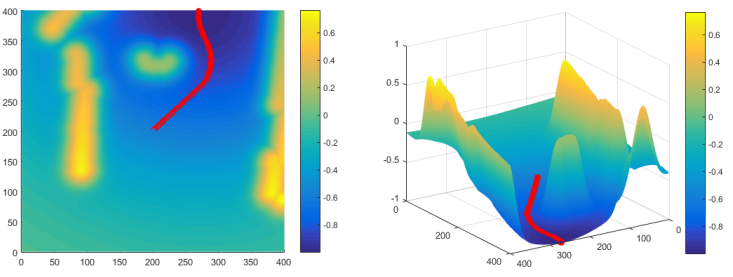
Para determinar la dirección de navegación por el campo de potencial en cada punto de la retícula, es necesario calcular el gradiente. Esta función determina el camino de mayor pendiente, el cual acaba en el punto objetivo marcado anteriormente, ya que es el punto del mapa de menor potencial.

El gradiente es el campo vectorial obtenido al derivar la función gaussiana, cuyas componentes son las derivadas parciales de esta función.

$$\frac{\partial f(x,y)}{\partial x} = -A\frac{(x-x_o)}{\sigma_x^2}\exp\left(-\left(\frac{(x-x_o)^2}{2\sigma_x^2}+\frac{(y-y_o)^2}{2\sigma_y^2}\right)\right). \tag{4.9}$$

$$\frac{\partial f(x,y)}{\partial y} = -A\frac{(y-y_o)}{\sigma_y^2}\exp\left(-\left(\frac{(x-x_o)^2}{2\sigma_x^2}+\frac{(y-y_o)^2}{2\sigma_y^2}\right)\right). \tag{4.10}$$

Siguiendo la dirección del campo vectorial obtenido con esta función, se llega al punto objetivo, y como se puede observar en las figuras (a) y (b), siguiendo el camino de menor gradiente, se esquivan los obstáculos que se interponen en la dirección de movimiento.



(a) Camino de menor gradiente 2D (b) Camino de menor gradiente 3D

Figura 4.5: Gradiente del campo de potencial (linea roja)

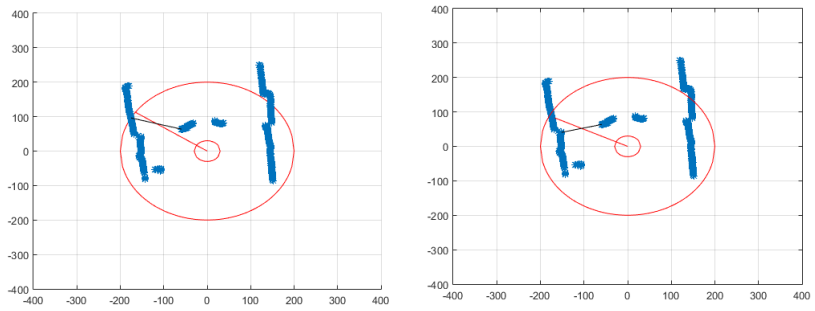
4.2.2. ND (Nearness Diagram)

El objetivo de este método de la misma forma que el de campos de potencial, es esquivar los obstáculos que se interpongan en la dirección principal de movimiento del quadcopter. Como se ha mencionado antes, la implementación del método completo supone una cantidad de cálculos elevada, que resulta difícil de implementar en el procesador embarcado, con las restricciones de tiempo real necesarias para una rápida corrección del movimiento. Por ello se implementa una versión simplificada del método.

El algoritmo ND, busca los posibles huecos de paso, físicamente atravesables por el vehículo, a partir de la información del escáner láser. Seleccionado el mejor hueco (con criterios de seguridad y proximidad angular al objetivo), el método calcula una dirección instantánea de movimiento en cada periodo de muestreo, que dirige al vehículo evitando los obstáculos circundantes. Se establece un radio de seguridad para el paso del vehículo, en el cual se identifican los huecos de paso. Se explica a continuación la implementación simplificada del método ND.

4.2.2.1. Funcionamiento del algoritmo

El objetivo es encontrar el hueco de mayor tamaño más cercano a la dirección principal de movimiento. Para ello se realiza un barrido en el sentido de escaneo del láser, y se localizan los obstáculos que se encuentran dentro del radio de seguridad. Una vez obtenidos todos estos obstáculos, se identifican los huecos formados entre ellos y se calcula la distancia mínima entre dichos obstáculos. Para ello se fija el punto del obstáculo más cercano y se busca el punto perteneciente al otro obstáculo que defina la menor distancia entre ellos, de esta forma se asegura que el quadcopter cabe por el hueco. La dirección de movimiento que se le envía al quadcopter, está definida por la recta que pasa por el punto medio del hueco.



(a) corrección de orientación errónea (b) corrección de orientación correcta

Figura 4.6: Dos posibles cálculos del hueco y sus correspondiente dirección de corrección de movimiento. La (a) conduce a colisión, la (b) es la correcta.

Se plantea un problema a tener que decidir los dos "extremos"del hueco, ya que hay muchos puntos candidatos posibles para serlo (ver 4.6). En la Figura 4.6 se pueden apreciar dos posible

elecciones del punto "izquierdo"del hueco. Una incorrecta elección conduce a un cálculo de la dirección de corrección fallida, que puede llevar a colisión lateral. En la figura 4.6(a) se puede apreciar como si no se realiza la corrección necesaria a la hora de elegir los dos puntos que definen el hueco, el quadcopter tendrá problemas para esquivar el obstáculo, ya que la dirección de movimiento, definida en la imagen con una linea roja, pasa muy cercana al obstáculo. Esto es debido a que la distancia tomada como tamaño del hueco es errónea. Para ello se ha implementado la búsqueda de la distancia mínima entre los dos obstáculos que es la que se muestra en la Figura 4.6(b), que conduce a una dirección correcta para esquivar con seguridad el obstáculo.

4.2.3. Cálculo de la orientación

Una vez calculada la dirección instantánea a la que hay que conducir al quadcopter, hay que calcular la acción de corrección de orientación (ángulo Yaw) a ejecutar en el mismo. Para ello se utiliza una combinación de el método inercial y el de campos de potencial o el ND. Se calcula primero la dirección principal de inercia, y cuando no se detecte obstáculo dentro del radio de seguridad, la dirección calculada por el método de potencial o el ND coincide con dicha dirección principal. En el caso de detectar un obstáculo en este radio, será necesario usar uno de los dos métodos nombrados anteriormente para corregir la dirección de movimiento. Al trabajar en una referencia robocéntrica, y considerar que en dicha referencia el quadcopter avanza siempre en la dirección del eje X (Figura 1.2), el ángulo Yaw de corrección coincide directamente con el ángulo asociado a la dirección de movimiento calculada por los métodos descritos. Este ángulo es la consigna que se aplica al autopiloto, el cual deberá aplicar una acción a los motores proporcional a la diferencia entre el ángulo Yaw calculado por el método y el eje X del quadcopter(Figura 4.7).

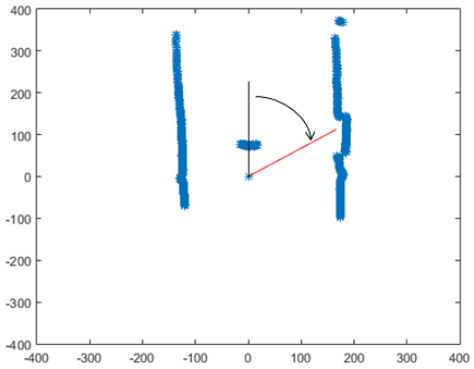
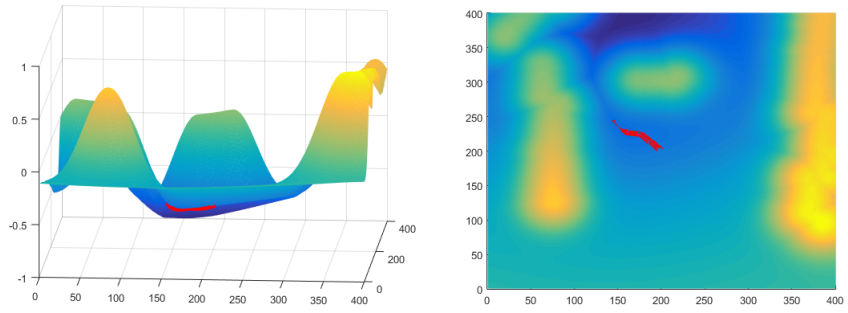


Figura 4.7: El eje X(linea negra) del quadcopter se alinea con el ángulo Yaw(linea roja) calculado

4.3. Comparación de métodos

Se ha evaluado la implementación de ambos métodos de evitación. Se ha tomado la decisión de utilizar el método ND en lugar del de campos de potencial debido las limitaciones de computo del procesador. El método de campos de potencial requiere más capacidad de procesamiento que el ND, y además presenta más dificultades a la hora de ajustar las constantes para crear el mapa de potencial. Si estas constantes no están bien ajustadas, se puede dar la posibilidad de encontrar un mínimo local debido a la combinación de potenciales atractivo y repulsivo.



(a) Mínimo Local 3D (b) Mínimo Local 2D

Figura 4.8: Bloqueo del avance debido a un mínimo local

Como se puede apreciar en la Figura 4.8, calculando el gradiente se llega a un mínimo local antes de llegar al objetivo, lo que implicaría una mala corrección de la dirección de movimiento del quadcopter, y posiblemente la colisión de este con el obstáculo o el bloqueo del movimiento de avance.

Otra de las razones por las que se ha decidido utilizar la versión simplificada del método ND, es que el tiempo de computo de este método es de 28ms frente a los 120ms del método de campos de potencial. Al utilizar el ND los sensores pueden muestrear más rápido que con el otro método consiguiendo así una mejor respuesta del sistema.

5. Implementación software del sistema

Un sistema en tiempo real [12] es un sistema informático que debe tener la capacidad de ejecutar las acciones requeridas en intervalos de tiempo bien definidos. Para ello, es necesario disponer de mecanismos adecuados para medir el tiempo, controlar la duración de las acciones del sistema, activar tareas o eventos en instantes determinados, reconocer fallos en los plazos de ocurrencia de los eventos y asegurar los plazos de ejecución de las acciones.[13]

5.1. Estructura del software

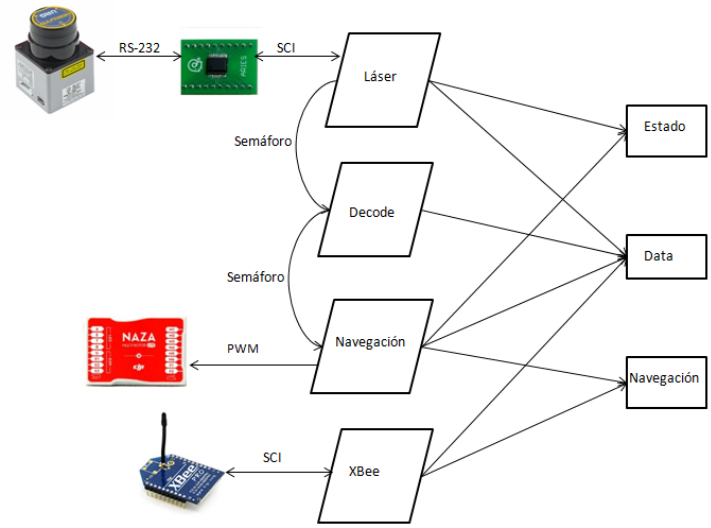


Figura 5.1: Estructura software del sistema

Las actividades que se van a implementar son las siguientes:

- Gestión del Sensor Láser. Este manda una trama cada 127 *ms* y se lee a través del puerto serie SCI.

- Decodificación de las medidas. Es necesario decodificar los datos recibidos del sensor láser, cada vez que el procesador recibe una trama de medidas completa. Para ello la tarea encargada de gestionar el láser, activa el semáforo que habilita esta tarea.
- Cálculo de direcciones de navegación. Esta tarea se activara cada vez que se haya decodificado una trama del láser mediante un semáforo lanzado por la tarea de decodificación.
- Envío de comandos al quadcopter a través del XBee. Cada 300ms se envían y reciben tramas de datos.

5.2. Implementación del software

El software está basado en tareas e implementado con las herramientas de SYS/BIOS. Cada tarea tiene una parte de inicialización en el main, un bucle infinito en el que se va a ejecutar el código y una primitiva bloqueante en la que se va a quedar esperando la tarea hasta ser ejecutada. Estas primitivas son los semáforos. Cuando una tarea está en el estado de espera, significa que está esperando la llegada de la señal del semáforo que le da permiso para ser ejecutada. Existen varios tipos de tareas en función del tipo de activación, estas son:

- **Tareas periódicas**: Se ejecuta cada cierto tiempo definido constante
- **Tareas esporádicas**: Se activan cuando ocurre un evento concreto, como por ejemplo una interrupción hardware o como es el caso de esta aplicación, algunas tareas deben ser ejecutadas cuando finaliza la ejecución de una parte de código. En este caso se trataría de interrupciones software.

En la siguiente imagen se explica el funcionamiento de los semáforos en el caso de que la tarea sea periódica, se puede ver como cuando llega un tick del reloj (1), que coincide con el periodo de la tarea(2), se lanza la señal del semáforo (3),(4), y si se cumple también que la tarea está lista para ejecutarse (5), esta entra en el procesador y empieza a ejecutarse(6).

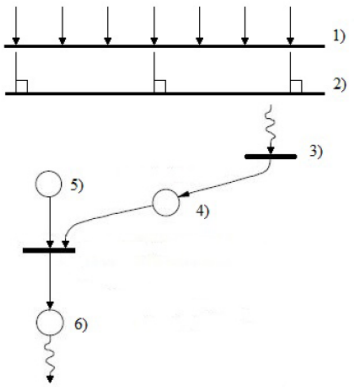


Figura 5.2: Activación de una tarea

En el caso de que las tareas sean esporádicas, la activación del semáforo no vendrá dada por el reloj del sistema, sino que será lanzada por la interrupción hardware que se de en el sistema como por ejemplo la llegada de caracteres por el puerto SCI, o en el caso de que la interrupción sea software, el fin de la ejecución de una tarea determinara el lanzamiento de un semáforo para que se ejecute otra tarea.

Estas tareas ademas de tener asignado un periodo y un semáforo como se ha dicho anteriormente, es necesario asignarle una prioridad. Existen varios tipos de planificación, pero en este caso vamos a centrarnos únicamente en la planificación basada en prioridades estáticas, dando prioridad siempre a la tarea más urgente, es decir, a la de menor separación mínima entre eventos. Para ello sera necesario asignar las prioridades de tal forma que en cada momento se ejecute la tarea más prioritaria entre todas las que sean ejecutables, pudiendo ser expulsada del procesador en el caso de que una tarea de mayor prioridad este preparada para ser ejecutada. En el siguiente diagrama se pueden ver los diferentes estados en los que se puede encontrar una tarea.

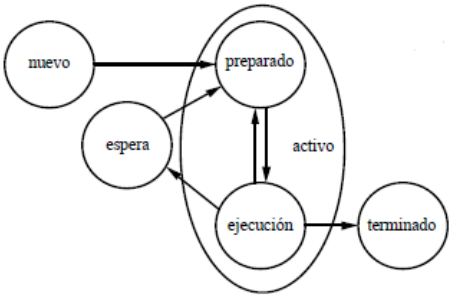


Figura 5.3: Estados de una tarea

Como se puede ver en la anterior imagen, cuando llega una tarea pasa a estar preparada, si llega la señal del semáforo, el procesador empieza a ejecutar dicha tarea. En el caso de que una tarea más prioritaria este preparada para su ejecución, la tarea que se estaba ejecutando inmediatamente pasará al estado de espera hasta que la tarea más prioritaria sea ejecutada. Una vez terminada dicha ejecución la tarea que estaba en espera podrá volver a ser ejecutada hasta que finalice su ejecución.

5.2.1. Tareas

Se han creado 4 tareas:

- **Recepción de tramas del laser**: Es la tarea principal del sistema con la que se recogen los escaneos enviados por el sensor. Este interrumpe al sistema cada vez que recibe un carácter y cuando se detecta el carácter que indica el final del escaneo, se activa el semáforo para que esta tarea entre en el procesador. En ella se realiza la verificación de

la trama y posteriormente se guardan los datos recibidos en el servidor de datos para que puedan ser utilizados por otras tareas.

- **Decodificación de tramas del laser**: Cada vez que se verifica la recepción de una trama se produce la activación del semáforo que da permiso a esta tarea para entrar en el procesador.
- **Cálculo de las direcciones para la navegación**: Esta tarea es periódica y se ejecuta cada 130ms. En ella se realizan todos los cálculos necesarios para obtener la dirección principal de movimiento, y detectar los obstáculos que se interpongan en el camino del quadcopter.
- **Envio y recepción de tramas con el ordenador**: Es una tarea periódica que se ejecuta cada 300 ms, esta envía la telemetría del quadcopter al ordenador e interpreta los comandos enviados desde el ordenador al láser.

5.2.2. Servidores

Los servidores son una herramienta necesaria cuando se está trabajando con un sistema de varias tareas. La función principal de estos es almacenar variables que van a ser utilizadas por varias de estas, para que en el caso de que una esté modificando una variable, otra no pueda acceder a ella hasta que la variable haya sido modificada.

Se han creado 3 servidores:

- **Servidor de Estados**: En este servidor se almacena el estado en el que se encuentra el láser, este puede ser: sin inicializar, parado o enviando. La tarea de recepción de datos del láser escribe en este servidor para que todas las demás tareas puedan acceder a el y leer el estado en el que este se encuentra y realizar una cosa u otra en función de este.

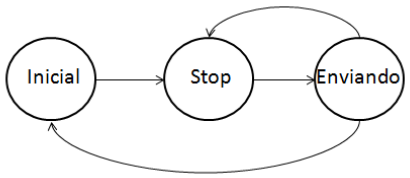


Figura 5.4: Estados del sistema

En el autómata de estados de la Figura 5.4, se pueden ver los diferentes estados en los que puede encontrarse el sistema. En el estado inicial, el sistema está esperando la llegada del comando de inicialización del sensor láser. Cuando se recibe este comando, se pasa al estado de *Stop*, en el que se está esperando la llegada de la orden que hace que el láser comience a enviar medidas. Cuando esto ocurre se pasa al estado *enviando*, en el que se

permanece hasta la llegada de un comando que detenga el envio de medidas, o un reset del sistema.

- **Servidor de datos**: En el escribe la tarea de recepción del láser la trama recibida en cada momento para que la tarea de decodificación pueda acceder a estos datos y procesarlos. Una vez procesados esta escribe en el las medidas decodificadas para que la tarea de navegación pueda realizar los cálculos necesarios con ellas.

- **Servidor de Navegación**: En el se escriben las direcciones calculadas por los distintos métodos

5.2.3. Optimización del Código

Todo el software implementado está cargado en la memoria flash del microcontrolador. Para reducir el tiempo de cómputo de las funciones que críticas(por ejemplo, las de navegación), se han copiado de la memoria flash a la RAM [1] ya que el tiempo de lectura y escritura en la segunda es mucho más rápido. Utilizando este procedimiento, se ha conseguido reducir el tiempo de cómputo de la tarea de navegación nombrada anteriormente de 120ms a 30ms, lo que supone una mejora importante en la respuesta del sistema.

5.3. *Drivers* Software Implementados

Se han implementado los *drivers* necesarios para poder integrar el hardware utilizado en el F28335

5.3.1. Sensor Láser

Para poder procesar las medidas enviadas por el sensor, es necesario crear una serie de funciones para procesar los caracteres recibidos por el puerto SCI. Para ello es necesario habilitar una interrupción hardware que llame a unas funciones de recepción y transmisión de caracteres, cada vez que el microcontrolador recibe un carácter.

Se ha realizado una adaptación de las funciones ya implementadas para recibir tramas del módulo XBee, para poder recibir los datos del sensor láser. Ademas de estas funciones de recepción, se han implementado todas las funciones de decodificación para poder hacer uso en los algoritmos de navegación de las medidas recibidas.

[1] RAM: Random Access Memory

5.3.1.1. Filtrado de medidas

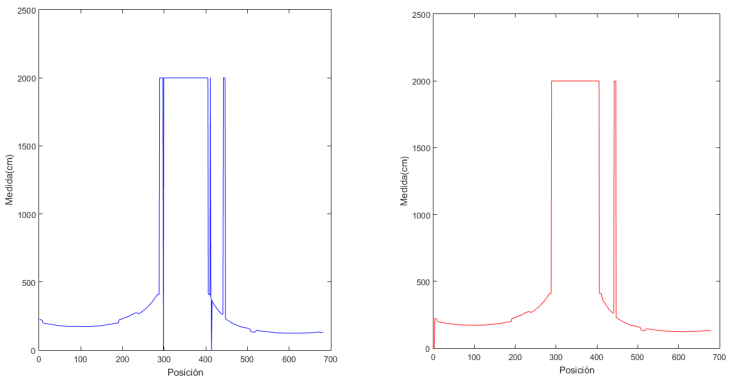
Como prácticamente en todas las aplicaciones en las cuales es necesario obtener datos a través de un sensor, se pueden adquirir datos erróneos debido a cualquier tipo de ruido que pueda alterar las lecturas del sensor. Este ruido, puede tratarse por ejemplo de una mala adquisición del dato por fallo del hardware, o del propio ruido del sensor que proporciona alguna medida espúrea. Para ello se va a hacer uso de técnicas de filtrado que eliminen o reduzcan en la medida de lo posible el ruido. En este caso, el tipo de errores que se van a dar, son medidas erróneas en determinados posiciones angulares del escaneo. Un filtro adecuado para corregir estos errores, es el de mediana.

- **Filtro de mediana**

El filtro de mediana es una técnica no lineal de filtrado digital comúnmente utilizada para reducir el ruido en imágenes o en señales. La técnica consiste en asignar a cada punto el valor de la mediana local (muestras alrededor de cada valor de la señal); sólo cambian aquellos valores que no corresponden a la mediana de la muestra. La agrupación de las muestras para el cálculo de la mediana se denomina ventana. Lo bueno de este tipo de filtrado, es que sustituye los valores erróneos por valores que se encuentran en la señal. Si el número de muestras es impar, el cálculo de la mediana es sencillo: Simplemente habrá que ordenar los valores de la ventana y escoger el valor central.

Con ± K vecinos ($2K+1$ muestras en total, incluyendo la central):

$$f(x) = ymed = median\{f(x-K), \cdots, f(x-1), f(x), f(x+1), \cdots, f(x+K)\} \quad (5.1)$$



(a) Medidas sin filtrar (b) Medidas filtradas

Figura 5.5: Medidas filtradas con un filtro de mediana

Para realizar el filtrado de un escaneo entero, es necesario aplicar la técnica explicada anteriormente con las 682 medidas tomadas por el sensor. El único inconveniente de aplicar

esta técnica, es que el filtro aplica un retraso de $\frac{2K-1}{2}$ valores. Por ello es necesario corregir este "offset"para que cada medida se corresponda con su posición real.

5.3.2. XBee

La comunicación inalámbrica se realiza a través del XBee. Con él se envían los comandos desde el ordenador al láser para inicializarlo, y se recibe la dirección de movimiento calculada con los métodos de navegación.

Este dispositivo es de gran utilidad, ya que proporciona la posibilidad de visualizar toda la información necesaria para estudiar el comportamiento del quadcopter en tiempo real.

El XBee va conectado al tercer puerto SCI del microcontrolador. El funcionamiento es igual que el del láser: se crea una interrupción hardware y se asigna a este puerto SCI para que llame a unas funciones ya implementadas que procesan los caracteres recibidos y enviados.

5.3.3. ESC y Motores

Es necesario enviar un PWM de unos valores y frecuencia determinados. La frecuencia del PWM enviado a los motores, puede estar entre 50 y 450 Hz en este caso se ha elegido 73Hz, lo que implica un refresco del valor del PWM de 13ms.

Para realizar el PWM se han utilizado los módulos PWM incorporados en el microcontrolador.

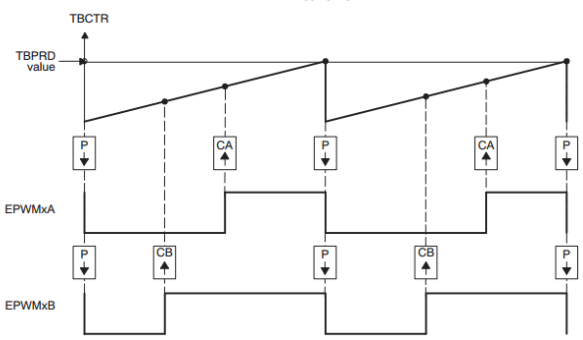


Figura 5.6: Registro PWM

Como se puede ver en la Figura 5.6, es necesario calcular el valor de los registros *TBPRD* y *CMPA*. El funcionamiento es el siguiente: Un *Timer* va contando ciclos de reloj hasta llegar al valor del registro *CMPA*, en ese momento se pone a 1 el valor del PWM. Cuando el *Timer* alcance el valor del registro de periodo(*TPRD*), el valor del PWM volvera a ser 0.

5.4. Diagrama de Tareas

Con el fin de tener una representación esquemática de la distribución de las tareas y servidores del sistema y sus características, se ha realizado un diagrama de bloques, en el que se representa el tiempo de lectura y escritura en los servidores e indicando que tareas acceden a ellos. Cada uno de los tiempos que se muestran en este diagrama, se han calculado utilizando las herramientas de SYS/BIOS.

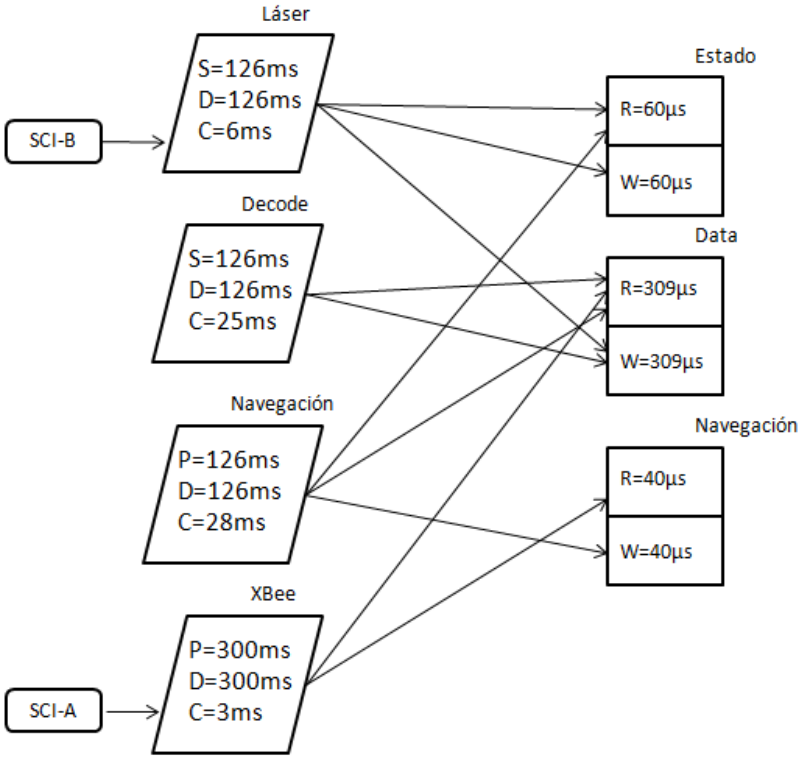


Figura 5.7: Esquema de Tareas del sistema

5.4.1. Cálculo de tiempos de bloqueo

Para calcular el tiempo máximo que una tarea puede ser bloqueada por un servidor, es necesario conocer todas las características temporales de las tareas del sistema.

Tarea	Prioridad	C(ms)	T(ms)	D(ms)	Bhp(ms)
Laser	4	7	126	126	0.369
Decode	3	29	126	126	0.369
Navigation	2	32	126	126	0.309
XBee	1	3	300	300	-

Cuadro 5.1: Tabla donde se recogen las características temporales del sistema

- **Tarea Laser** : Puede ser bloqueada por cualquiera de los servidores a los que esta tarea accede, en este caso la tarea *Navigation* que accede al servidor de *Estados*, puede bloquear a la tarea *Laser* 60 μs, asi como la tarea *Decode* que accede al servidor *Data* puede bloquearla 309 μs. Por ello el tiempo de bloqueo es igual a la suma de los dos bloqueos: 369 μs
- **Tarea Decode** : Puede ser bloqueada por las mismas tareas que la tarea *Laser*.
- **Tarea Navigation** : Puede ser bloqueada únicamente por la tarea XBee que accede al servidor *Data* durante 309 μs.

5.4.2. Cumplimiento de plazos

Para comprobar que todas las tareas del sistema se ejecutan en su plazo de respuesta, sera necesario comprobarlo de la siguiente forma:

$$W(D_i) = \sum_{j=1}^{i-1} C_j \lceil \frac{D_i}{P_j} \rceil + B_i < D_i \tag{5.2}$$

- $W(D_i)$ es el trabajo del procesador en el plazo de respuesta de la tarea i
- C_j es el tiempo que le cuesta al procesador ejecutar la tarea j
- D_i es el plazo de respuesta de la tarea i
- P_j es el periodo de la tarea j
- B_i es el tiempo de bloqueo que puede sufrir la tarea i

Esta condición es suficiente pero no necesaria para asegurar el cumplimiento de plazos de las tareas del sistema. En el caso de que todas las tareas del sistema cumplan esta condición, se asegura que el sistema cumple plazos.

- **Láser**

$$W(D_{Laser}) = 7 + 0{,}369 = 7{,}369ms < 126ms \tag{5.3}$$

- **Decode**

$$W(D_{Decode}) = \lceil \frac{127}{127} \rceil 7 + 29 + 0{,}369 = 36{,}369ms < 126ms \tag{5.4}$$

- **Navigation**

$$W(D_{Navigation}) = \lceil \frac{127}{127} \rceil 7 + \lceil \frac{127}{127} \rceil 29 + 32 + 0{,}369 = 68{,}369ms < 126ms \tag{5.5}$$

- **XBee**

$$W(D_{XBee}) = \lceil \frac{300}{127} \rceil 7 + \lceil \frac{300}{127} \rceil 29 + \lceil \frac{300}{127} \rceil 32 + 3 = 207ms < 300ms \tag{5.6}$$

Como se puede observar, se cumplen todos los plazos de respuesta de este sistema sin ningún problema. Se calcula la utilización del procesador, y debido al elevado coste de cómputo de las tareas implementadas, se está utilizando un 55 %

$$U = \sum_{j=1}^{n} \frac{C_j}{T_j} = 54{,}9\,\% \tag{5.7}$$

6. Pruebas de campo con el quadcopter

Para llegar a realizar las pruebas en el modelo real, previamente hay que realizar una serie de simulaciones de los métodos implementados y calibrar los motores adecuadamente. Dada la elevada potencia de los motores, es necesario probar el quadcopter de forma controlada para evitar que este se descontrole y provoque una avería o pueda llegar a lesionar a alguien.

Para realizar las siguientes pruebas se han utilizado ambos métodos: el de campos de potencial y el ND. Pero debido a las ventajas que presenta el método ND frente al de campos de potencial explicado en el capítulo 4, se ha utilizado finalmente para las pruebas reales el método ND. Las pruebas realizadas son las siguientes:

- El primer paso es comprobar que los métodos de navegación funcionan correctamente. Para ello se han simulado en Matlab dichos métodos y se ha ido desplazando el sensor láser de forma manual por un entorno similar al que se van a realizar las pruebas con el modelo real comprobando así que la consigna de orientación calculada es la correcta.

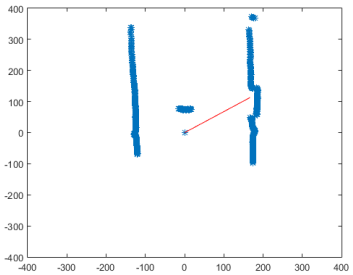


Figura 6.1: Prueba realizada desplazando el sensor manualmente

  Como se puede ver en la Figura 6.1, se ha ido desplazando el sensor manualmente, comprobando que la dirección hacia la que se tiene que mover el quadcopter(flecha roja en la Figura 6.1) siempre apunte hacia el hueco correcto.

- Una vez comprobado el correcto funcionamiento de los métodos de navegación, es necesario comprobar que la acción ejercida por los motores es suficiente para alcanzar las consignas de orientación. Para ello se ha sujetado el quadcopter con una pértiga(Figura 6.2) y con los motores a una potencia muy baja, se ha ido desplazando por el entorno de

vuelo de forma manual para comprobar que las correcciones de orientación enviadas por el procesador son correctas.

Figura 6.2: Prueba realizada con la pértiga de sujección

- Antes de pasar a volar el quadcopter sin ningún tipo de protección, se realizará una última prueba utilizando el mismo sistema de protección explicado en el punto anterior, pero esta vez la potencia de los motores será lo suficientemente elevada para que el quadcopter sea capaz de volar solo. De esta forma la pértiga no ejercerá ninguna fuerza sobre el quadcopter y será utilizada únicamente en el caso de que este se descontrole.

Una vez que se han realizado todas estas pruebas, llega el momento de probar el quadcopter sin ningún tipo de protección. Para ello se arrancará el quadcopter con un mando y se controlará con éste únicamente la potencia de los motores y la inclinación (ángulo Pitch) para que el quadcopter pueda avanzar con la orientación (ángulo Yaw) calculada por el método ND.

Se han realizado dos pruebas de vuelo diferentes:

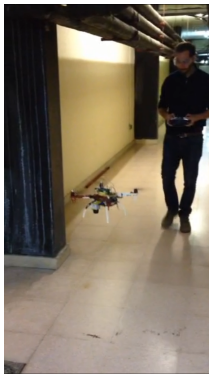


Figura 6.3: Navegación por una galería

Se comienza navegando por una galería sin ningún obstáculo. Como se puede ver en la Figura 6.3 el quadcopter navega por el centro de la galería sin ningún problema.

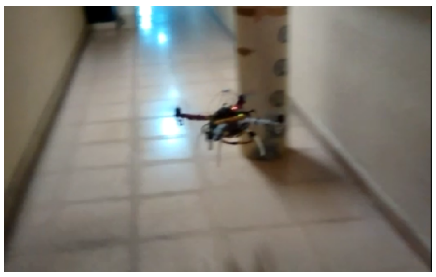
(a) Detección del hueco y reorientación hacia él

(b) Corrección de la dirección y entrada por el hueco

Figura 6.4: Detección del hueco por el que debe pasar y entrada a través de él

En la Figura 6.4 se puede ver como el quadcopter que estaba navegando por el centro de la galería, se encuentra con el camino bloqueado por unos obstáculos que le impiden el paso. Esto le obliga a corregir la dirección de movimiento orientándose así hacia el hueco por el que puede pasar sin ningún problema.

(a) Navegando por la galería sin detectar aun los obstáculos



(b) Detección del primer obstáculo y corrección de la orientación

(c) Detección del segundo obstáculo y corrección de la orientación

Figura 6.5: Vuelo por una galería en la que se encuentra con dos obstáculos

Por último se han realizado unas pruebas en el mismo entorno que la anterior prueba, pero esta vez los obstáculos se han situado en la galería por la que el quadcopter está navegando (Figura 6.5). Éste debe corregir su dirección para evitar la colisión con los obstáculos.

7. Conclusiones

El objetivo principal de este TFG es el desarrollo de técnicas de navegación, basadas en la información de un sensor rango láser integrado en el F28335. Este proporciona información de la distancia a la que se encuentran los obstáculos para poder crear un mapa local del entorno en el que navegará el quadcopter. A partir de la información sensorial, se ha calculado la dirección que este deberá llevar(ángulo yaw) para desplazarse por el entorno y esquivar obstáculos.

Estos objetivos se han conseguido de la siguiente forma:

- Integración de todo el hardware en el F28335
- Desarrollo de un *driver* software para la integración del sensor láser en el F28335.
- Filtrado y procesamiento de la información obtenida con el sensor.
- Desarrollo de un sistema de comunicación inalámbrica para visualizar la telemetría del quadcopter
- Desarrollo de técnicas de navegación basadas en la información sensorial.
- Implementación de un sistema de tiempo real capaz de gestionar todos los eventos necesarios para llevar a cabo el control del quadcopter.

Se han desarrollado y evaluado en el modelo real dos métodos de evitación de obstáculos. En las pruebas finales se ha decidido utilizar únicamente el método ND para esquivar los obstáculos que se interpongan en la dirección de movimiento del quadcopter. Ésto es debido a que el microcontrolador es demasiado lento para obtener un buen tiempo de respuesta con el método de campos de potencial, lo que impide realizar una corrección de ángulo lo suficientemente rápida para esquivar los obstáculos correctamente. Además es complicado ajustar las desviaciones típicas para que el quadcopter detecte los obstáculos con suficiente antelación para esquivarlos. Por estos motivos se ha utilizado en las pruebas finales el método ND.

El desarrollo de técnicas de navegación más complejas, se ha visto limitado como se ha explicado anteriormente por la velocidad de cálculo del procesador y por la imposibilidad de obtener una buena referencia absoluta con el IMU para poder saber en todo momento la posición en la que se encuentra el quadcopter.

Este TFG ha sido un primer paso hacia la integración de sensores en los drones, para darles la posibilidad de volar forma autónoma. Se plantea como futuro trabajo la integración de otros sensores capaces de obtener la información necesaria para controlar el resto de grados de libertad del quadcopter(x,y,z,Roll,Pitch) y el uso de una unidad de procesamiento más potente, para la implementación de métodos de navegación más complejos.

Se han realizado numerosas pruebas de vuelo en entornos cerrados con obstáculos en las que se han probado los métodos de navegación implementados, obteniendo excelentes resultados.

Bibliografía

[1] Información sobre los quadcopters. https://es.wikipedia.org/wiki/Quadcopter.

[2] Hokuyo. Documentación del sensor láser hokuyo 04lx. https://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx.html.

[3] Texas Instruments. Documentación técnica de el microcontrolador F28335. http://www.ti.com/tool/TMDSDOCK28335.

[4] Estructura quad. www.electronicarc.com.

[5] Texas Instruments. Página de descarga del software CodeComposer. www.ti.com/lsds/ti/tools-software.

[6] Información acerca de matlab. http://es.mathworks.com/.

[7] Digi. Documentación del dispositivo xbee. http://www.digi.com/lp/xbee.

[8] Maxim Integrated. Datasheet del conversor MAX3222. http://pdfserv.maximintegrated.com/en/ds/MAX3222-MAX3241.pdf.

[9] Javier Minguez and Luis Montano. Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios. *IEEE Transactions on Robotics and Automation*, 20(1), 2004.

[10] M. Jenkin G. Dudek. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000.

[11] I.R. Nourbakhsh R. Siegwart. *Introduction to Autonomous mobile robots*. Bradford Books, 2004.

[12] José Luis Villarroel Salcedo. Sistemas de tiempo real. http://web.archive.org/web/20140222165922/http://webdiis.unizar.es/~joseluis/STR.pdf,.

[13] Alan Burns y Andy Wellings. *Sistemas de tiempo real y lenguajes de programación, 3ª edición*. 2003.

ANEXOS

Anexo A

Codificación y Decodificación del sensor láser

Las medidas enviadas por el sensor como hemos dicho anteriormente vienen codificadas por dos caracteres cada una, teniendo un tamaño máximo de 12 bits por medida. La codificación se lleva a cabo separando los datos en parte alta y baja(6 bits cada parte) y después sumándoles "30H"para convertirlos a valores ASCII(Figura A.1).

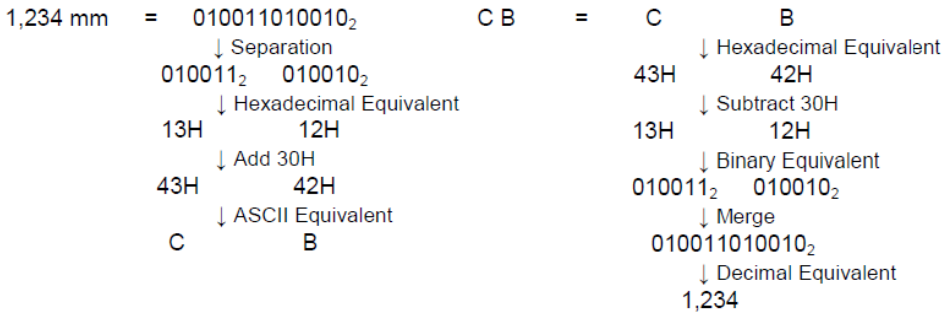


Figura A.1: Codificación y Decodificación de dos caracteres

Para que el láser comience a enviar tramas, es necesario enviarle una serie de comandos. Una vez enviados estos comandos, el sensor empezará a enviar tramas. El proceso de recepción es el siguiente: Cada trama empieza siempre por la letra *M*, por tanto en el momento que se detecta la recepción de este caracter por el puerto SCI, se almacenan los caracteres que se van recibiendo hasta que se detectan dos caracteres $\backslash n$ seguidos que indican el final de la trama.

A.1. Formato de comunicación

El sensor y el usuario se comunican con una serie de comandos predefinidos. En esta sección se va a explicar el funcionamiento de los comandos MD/MS, que son los utilizados para iniciar la transmisión de datos.

Cuando el sensor recibe un comando de este tipo, comienza a enviar medidas hasta completar el número de escaneos exigidos en el comando.

(HOST→ SENSOR)

M (4dH)	D (44H) or S (53H)	Starting Step (4bytes)	End Step (4 bytes)	Cluster Count (2bytes)	Scan Interval (1 byte)

Number of Scans (2 bytes)	String Characters (max 16-letters)	LF (1 byte)

Figura A.2: Estructura del comando MS

- En la primera parte del comando, elegimos el tipo de codificación, en este caso MS ya que en esta aplicación se va a codificar en dos caracteres.

- En los siguientes bytes, se elige el paso inicial y final del escaneo, por ejemplo:

  - Paso inicial: 0044(30H,30H,34H,34H)
  - Paso final: 0725(30H,37H,32H,35H)

- El siguiente registro de 2 bytes, sirve para agrupar aquellas medidas de pasos adyacentes, cuyo valor se pueda aproximar en una única medida, por ejemplo: Si el çluster count"vale 3, significa que en el caso de que 3 medidas de 3 pasos adyacentes sean 3059,3055 y 3062 el dato recibido sera 3055.

- En el "Number of Scans"se elige el número de escaneos que debe realizar el sensor. En el caso de que se quieran realizar multiples escaneos, se pone este registro a 00, y el sensor no parara de enviar medidas hasta que reciba la orden de detención

Una vez enviada la orden, el sensor puede responder de varias formas dependiendo de lo enviado en esta. En el caso de enviar el comando "MS0044072500000", que es el que se ha utilizado para la recepción de medidas en esta aplicación, la respuesta es la siguiente:

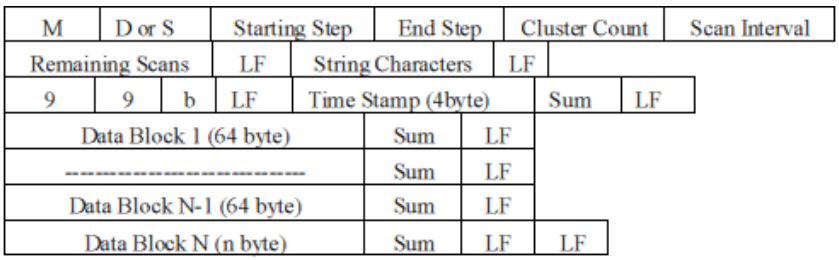


Figura A.3:

Se puede observar como el sensor devuelve el mismo comando que se le ha enviado para que empieze a enviar medidas, seguido de una serie de cabeceras y por ultimo una serie de N bloques de 64 bytes en los que se encuentran codificadas las medidas obtenidas en el escaneo.