



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática

Sistema de Navegación Semiautónomo en Interiores



Presentado por Mario Bartolomé Manovel
en Universidad de Burgos — 7 de abril de 2018

Tutores: Dr. Alejandro Merino Gómez
Dr. César Ignacio García Osorio
Dr. José Francisco Díez Pastor



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. Dr. Alejandro Merino Gómez, profesor del departamento de Ingeniería Electromecánica, área de Ingeniería de Sistemas y Automática. D. Dr. César Ignacio García Osorio y D. Dr. José Francisco Díez Pastor, profesores del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno D. Mario Bartolomé Manovel, con DNI 71298657Z, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado «Sistema de Navegación Semiautónomo en Interiores».

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 7 de abril de 2018

Vº. Bº. del Tutor:

Vº. Bº. del Tutor:

Vº. Bº. del Tutor:

D. Dr. Alejandro
Merino Gómez

D. Dr. César Ignacio
García Osorio

D. Dr. José Francisco
Díez Pastor

Resumen

El objetivo del proyecto es diseñar un sistema de navegación semi-autónomo en espacios cerrados, destinado a la asistencia en vigilancia de seguridad mediante drones. Dada una estancia, el drone deberá ser capaz de realizar un recorrido por el interior, grabando vídeo que será emitido a un servidor, y transmitiendo su posición dentro del mapa a un responsable de seguridad.

Descriptores

drone, UAV, semi-autónomo, semi-automático, vigilancia, navegación, interior, vídeo, filtro de partículas, campos potenciales, búsqueda de ruta, MSP, raspberry pi

Abstract

The point of this project is to design a semi-autonomous navigation system in enclosed spaces, destined to aid security vigilance using drones. Given an enclosed space, the drone should be able to make its path through it, recording video which will be streamed to a server, and updating its position inside of it to the security guard in charge.

Keywords

drone, UAV, semi-autonomous, semi-automatic, vigilance, navigation, indoor, video, particle filter, potential fields, path searching, MSP, raspberry pi

Índice general

Índice general	III
Índice de figuras	IV
Índice de tablas	V
Introducción	1
Objetivos del proyecto	2
Conceptos teóricos	3
3.1. Algoritmia	3
Filtros de Partículas	3
Campos Potenciales	3
Vector Field Histogram	9
3.2. Dispositivos Físicos	18
Raspberry Pi	18
Controladora de Vuelo	20
3.3. Protocolos	23
Pulse Width Modulation	23
MultiWii Serial Protocol	24
Secure SHell	27
Técnicas y herramientas	29
Aspectos relevantes del desarrollo del proyecto	30
Trabajos relacionados	31
Conclusiones y Líneas de trabajo futuras	32

Índice de figuras

3.1. Gradiente de Potencial de Atracción.	5
3.2. Gradiente de Potencial de Repulsión.	6
3.3. Mínimo local en zonas convexas.	7
3.4. Movimiento oscilatorio en corredores estrechos.	8
3.5. Distribución Histográfica de Probabilidades	10
3.6. Angulos relativos al VCP del drone	11
3.7. En azul Polar Obstacle Density(POD). En naranja el suavizado de este (sPOD).	14
3.8. Valle ancho encontrado entre el obstáculo, a la derecha, y un espacio abierto, a la izquierda.	15
3.9. Movimiento circular al encontrarse k_{targ} en un sector seguro por ecuación 15	16
3.10. Valle estrecho encontrado entre dos obstáculos. La ecuación 15 proporciona un sector seguro ubicado en la zona central del valle.	17
3.11. Raspberry Pi 3.	19
3.12. Benchmark de lector microSD OC.	20
3.13. Controladora de Vuelo Flip32.	21
3.14. Modulación en Ancho de Pulso. Arriba 100 % del ciclo usado. Centro 25 % del ciclo usado. Abajo 50 % del ciclo usado	23
3.15. Composición de un mensaje MSP	25

Índice de tablas

3.1. Componentes de una Raspberry Pi 3 Model B	18
3.2. Mensajes MSP. Estructura de datos y representación	26
3.3. Nomenclatura del módulo Struct de la implementación 3.5 de Python	27

Introducción

En este documento se encuentra toda la información relacionada con el Trabajo de Fin de Grado titulado *Sistema de Navegación Semiautónomo en Interiores*.

En él se puede encontrar la siguiente información:

- **Conceptos teóricos:** Ofrecen una base teórica de la que partir, para llevar a cabo el desarrollo completo del proyecto.
- **Técnicas y herramientas:** Se trata de las implementaciones de, ó uso dado a, los distintos conceptos teóricos anteriormente descritos.
- **Aspectos relevantes del desarrollo del proyecto:** Proporciona información detallada que se ha tenido en cuenta durante las diferentes fases de desarrollo del proyecto.
- **Trabajos relacionados:** Se trata de una lista, junto con una breve descripción, de los diferentes proyectos, papers o trabajos relacionados con el proyecto llevado a cabo.
- **Conclusiones y líneas de trabajo futuras:** Detalla una serie de posibles mejoras, modificaciones e incluso derivaciones, que pueden surgir del proyecto realizado.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

Conceptos teóricos

Esta sección auna los diferentes conocimientos teóricos necesarios para la realización del proyecto. A continuación, y en este orden, se explicarán los algoritmos utilizados, protocolos de comunicación y sistemas físicos empleados.

3.1. Algoritmia

Filtros de Partículas

Los Filtros de Partículas son modelos utilizados para tratar de estimar el estado de un sistema que cambia con el tiempo.

Fue definido como *bootstrap filter* en 1993 por N. Gordon, D. Salmond y A. Smith. Se trata de un método que pretende implementar filtros bayesianos recursivos, haciendo uso del método de Montecarlo, es decir, realiza repetidas medidas del estado para estimar la posición del sistema.

Campos Potenciales

A la hora de lograr una navegación segura en un entorno determinado, es necesario implementar un sistema de evasión de obstáculos.

Introducido por Oussama Khatib en 1986, el algoritmo de Campos Potenciales aporta una manera de evitar colisionar con los diferentes obstáculos existentes. Se basa en la idea de que el agente se mueve en un campo de fuerzas. La posición que debe alcanzar, su meta o destino, se presenta como una fuerza de atracción, y los obstáculos como fuerzas repulsivas. Los diferentes vectores fuerza, determinan la dirección y magnitud de la fuerza atrayente o repulsora.

Sea q la posición actual del agente, considerada como una partícula moviéndose en un espacio n -dimensional R^n . Por simplicidad, se considera la posición como una tupla, esto es, bidimensional, tal que $q = (x, y)$. El campo potencial en el que se encuentra el agente es una función escalar $U(q) : R^2 \rightarrow R$, generado

por la superposición de los campos atrayentes U_{att} y repulsores U_{rep} :

$$U(q) = U_{att}(q) + U_{rep}(q) \quad (1)$$

Por supuesto, la suma de todos los campos repulsores U_{rep} generan un vector repulsor, que sumado al de atracción puede determinar la necesidad de acercarse o alejarse de una determinada zona:

$$U(q) = U_{att}(q) + \sum_i U_{rep_i}(q) \quad (2)$$

Donde U_{rep_i} representa el campo potencial generado por un obstáculo i .

Suponiendo que $U(q)$ es diferenciable, en cada punto q , el campo potencial $\nabla(q)$ es un vector que apunta en la dirección que, **localmente**, incrementa $U(q)$.

De esta forma, si el potencial de atracción en la meta se considera 0 y a medida que el agente se distancia de ella, se incrementa, se puede establecer, teniendo en cuenta el potencial de repulsión producido por los diferentes obstáculos, un vector $F(q)$ cuya dirección apunta hacia donde se reduce el potencial U , y su magnitud establece la velocidad con que este se reduce:

$$F(q) = F_{att}(q) + F_{rep}(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q) \quad (3)$$

El potencial de atracción $U_{att}(q)$ se establece como una función cuadrática proporcional a la distancia a la meta. De esta forma crece exponencialmente a medida que el agente se aleja de su destino, y se reduce considerablemente en su cercanía. Así puede ser utilizado como fuente de información para establecer la velocidad de aproximación al objetivo, de forma que el agente no sobreactúe:

$$U_{att}(q) = \epsilon * \delta_{meta}(q)^2 \quad (4)$$

Donde δ es la distancia euclidiana desde la posición del agente q a la meta $meta$, y ϵ es un factor de escalado positivo, que puede utilizarse para ajustar la influencia de la distancia en la velocidad del agente. Y su gradiente negativo:

$$-\nabla U_{att}(q) = -\epsilon * (q - q_{meta}) \quad (5)$$

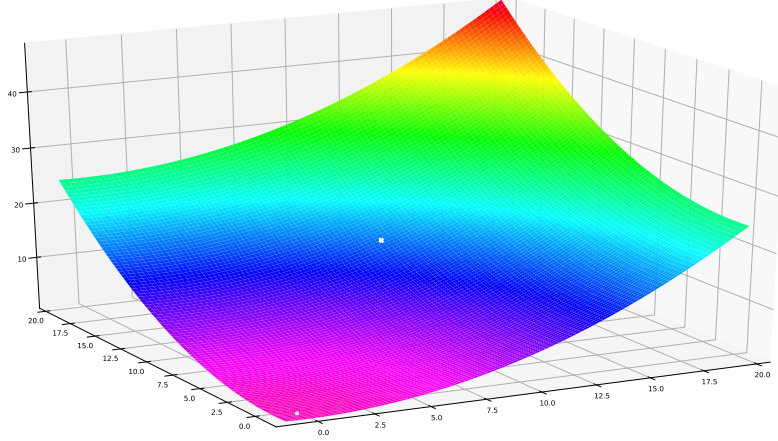


Figura 3.1: Gradiente de Potencial de Atracción.

En la figura 3.1 se ha establecido el agente en el punto (10, 12) y la meta en (0, 0). Puede verse el efecto del gradiente aplicado.

El potencial de repulsión $U_{rep}(q)$ lo establecen los obstáculos detectados por los sensores del agente. Si se establece una función proporcional a la distancia del agente a los obstáculos, el potencial repulsor crece en la cercanía a estos, y decrece con la distancia. De esta forma un obstáculo lejano al agente no debería presentar ninguna fuerza sobre él:

$$U_{rep}(q) = U = \begin{cases} 0 & \text{si } \delta > \rho \\ \eta * (\frac{1}{\delta_{obs}(q)} - \frac{1}{\rho})^2 & \text{si } \delta \leq \rho; \delta \neq 0 \end{cases} \quad (6)$$

Donde η es un factor de escalado positivo, que permite establecer como de fuerte se ve afectado el agente por el campo repulsor, δ es la distancia euclidiana desde la posición del agente q al obstáculo obs , y ρ es un factor de escalado positivo, que permite establecer la distancia de influencia de forma proporcional. Y su gradiente

$$-\nabla U_{rep}(q) = \begin{cases} 0 & \text{si } \delta > \rho \\ \eta * (\frac{1}{\delta_{obs}(q)} - \frac{1}{\rho}) * \frac{1}{\delta_{obs}(q)^2} \frac{q - q_{obs}}{\delta_{obs}(q)} & \text{si } \delta \leq \rho; \delta \neq 0 \end{cases} \quad (7)$$

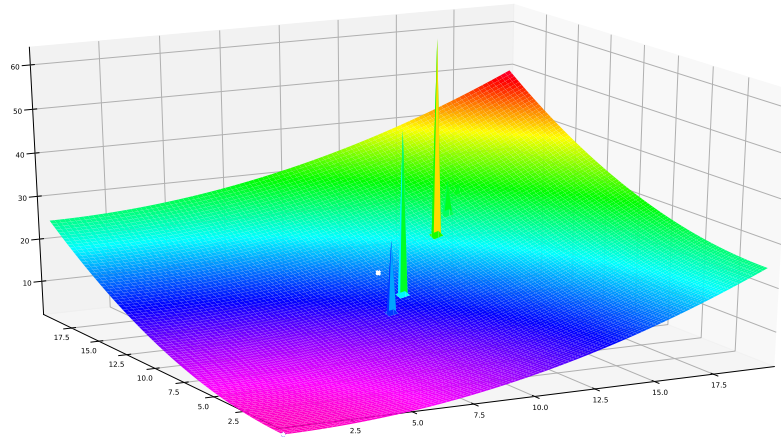


Figura 3.2: Gradiente de Potencial de Repulsión.

En la figura 3.2 se ha creado una serie de obstáculos artificialmente para ilustrar el comportamiento del algoritmo.

Desventajas

El algoritmo de campos potenciales se puede llevar a cabo con una implementación simple y eficiente haciendo uso de Numpy, dado que se puede crear una representación matricial del plano en que se desplaza el agente, que contenga los diferentes potenciales en cada posición. Esto hace sencillo realizar cálculo vectorizado sobre las matrices.

Sin embargo, tiene ciertas desventajas que lo han hecho, cuanto menos, inútil para las necesidades de este proyecto:

- Es un algoritmo basado en descenso de gradiente, y como tal puede quedar *atascado* en mínimos locales. Estos mínimos pueden encontrarse en zonas que contengan obstáculos en forma cóncava o de caja abierta, tal como se ilustra en la figura 3.3.



Figura 3.3: Mínimo local en zonas convexas.

- De encontrarse en un espacio con forma de corredor o pasillo, cualquier acercamiento hacia las paredes ocasionará una corrección en el sentido opuesto al campo repulsor, lo cual puede ocasionar un movimiento oscila-

torio si dicha corrección supone aproximar el agente a la pared opuesta, tal y como se ilustra en la figura 3.4.

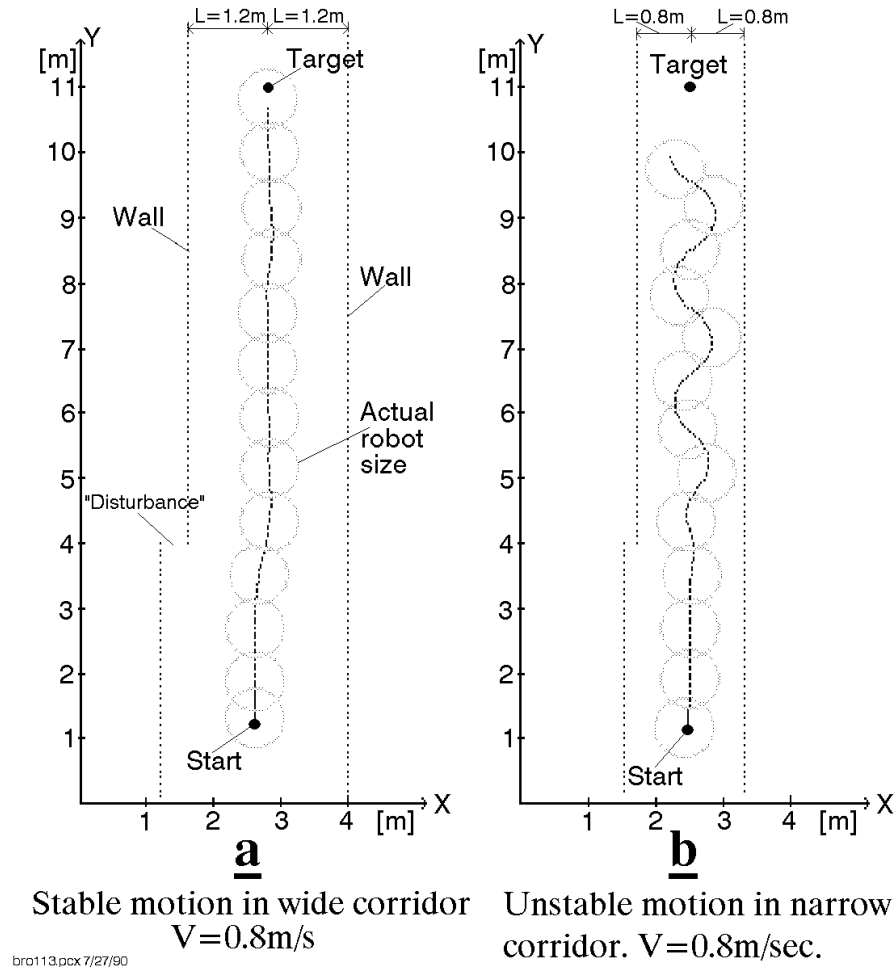


Figura 3.4: Movimiento oscilatorio en corredores estrechos.

- Alta complejidad computacional pese a poder modelarse como calculo vectorizado, requiere de muchas más operaciones que otros métodos más avanzados.
- No funciona correctamente con agentes que se desplazan a alta velocidad.
- Al realizar la adición de todos los campos potenciales, tanto repulsores como atrayente, se produce una gran pérdida de información sobre la distribución local de los obstáculos.

Por estos motivos, se ha desechado el algoritmo de Campos Potenciales como sistema de evasión de obstáculos, y se ha pasado a un modelo más

avanzado basado en un plano cartesiano, conocido como *Vector Field Histogram* y detallado en el apartado **Vector Field Histogram**.

Vector Field Histogram

El Vector Field Histogram, o Histograma de Campos Vectoriales, *VFH* de aquí en adelante, fue introducido por Borenstein y Korem en el año 1991. Permite la detección de obstáculos y su evasión mediante el control de la dirección y velocidad del agente en tiempo real.

Para ello realiza una representación del entorno del agente en forma de histograma. Dicho entorno, llamado *Región Activa C** se representa como una *ventana* que se desplaza con el agente en el centro. Se trata un algoritmo de búsqueda local, sin embargo se ha mostrado que suele producir rutas cercanas al óptimo, aunque en el caso de este proyecto no es relevante, dado que la intención es que se exploren todas las zonas del entorno.

Es un algoritmo más robusto, dado que presenta cierta insensibilidad a lecturas erróneas, y eficiente que el algoritmo de Campos Potenciales explicado en **3.1**.

Se compone de tres partes:

1. Red Cartesiana de Histogramas: Se construye un plano cartesiano representando el entorno cercano del agente. Por *cercano*, se entiende una distancia que los sensores de distancia puedan abarcar, dado que no tendría sentido tratar de computar zonas inalcanzables en un momento determinado. Dicho plano será representado como un histograma, y de ahí su nombre habitual *Cartesian Histogram Grid*. En **3.1**.
2. Histograma Polar: Una representación unidimensional, obtenida a partir de una reducción de la Red Cartesiana definida en el punto anterior. Se compone de n sectores angulares k , de anchura α , de forma que $n \cdot \alpha = 360$; $n \in \mathbb{Z}$. Dichos sectores k contienen un valor h_k que representa la densidad de obstáculos, *Polar Obstacle Density* o *POD* de aquí en adelante, contenida en él. En **3.1**.
3. Capa de salida: Representa el resultado del algoritmo. A partir del POD, se obtienen los posibles *valles*¹ candidatos existentes entre obstáculos, y se elige el que más se aproxime a la dirección de la meta establecida. Finalmente, entrega los valores de cambio de dirección necesario en cada momento. En **3.1**.

¹Así denominados por su representación en forma de histograma

Cartesian Histogram Grid

Se modela el espacio cercano al drone en forma de malla. En cada celda (i, j) de esa malla se encuentra un valor $c_{i,j} \in \mathbb{Z}$ que establece la certeza de la existencia de un obstáculo. La cantidad de celdas existentes en la malla se establece en función de los límites del sensor, y teniendo en cuenta que es necesario establecer una precisión c , i.e: Si se dispone de un sensor con un rango de medidas de hasta 400cm se puede crear una malla de 80x80, suponiendo que cada celda tenga un tamaño de $c = 5$, $5 \times 5 \text{cm}$. De esta forma se consigue una precisión de $\pm 5 \text{cm}$

Dicho valor se obtiene de la lectura proporcionada por el sensor de distancia, en este caso un sonar, y se establece en el centro del ángulo de barrido del sonar, ver ???. La eficiencia superior de este algoritmo se basa precisamente en incrementar el valor $c_{i,j}$ de únicamente una celda con cada medida. Dicha celda (i, j) se encuentra a una distancia de $R = \frac{\delta_{obs} q}{c}$ celdas y en la bisectriz del ángulo barrido por el sonar. Si bien esta solución puede parecer una simplificación del problema, se llega a generar una distribución probabilística tomando múltiples medidas de forma continua mientras el agente se desplaza por el entorno. Por tanto, la misma celda, cercana a un obstáculo, y sus vecinas serán incrementadas repetidas veces, tal y como se muestra en la figura 3.5

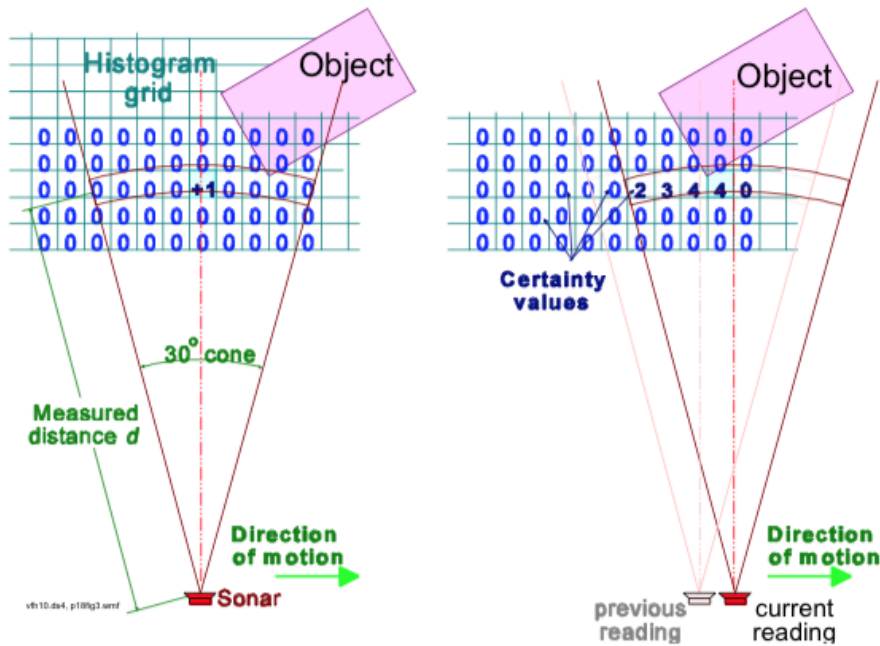


Figura 3.5: Distribución Histógrámica de Probabilidades

Polar Histogram

A partir de las medidas obtenidas en el Histogram Grid, se realiza la primera reducción de información para construir un Histograma Polar. Para ello, los contenidos de la región activa C^* son tratados como un *vector de obstáculos* cuya dirección está definida por la dirección β de la celda en cuestión al centro del agente VCP^2 , y que viene definida por la función:

$$\beta_{i,j} = \tan^{-1} \frac{y_i - y_0}{x_i - x_0} \quad (8)$$

La ecuación 8 devuelve un valor $\beta \in [-\frac{\pi}{2}, \frac{\pi}{2}]rad$. Al hacer uso de Numpy, es sencillo calcular los ángulos en los que se encuentra cada celda de la región activa C^* con respecto al agente que se encuentra en su centro, de forma vectorizada. Tal y como puede verse en la figura 3.6


-135			-90			-45
	-135		-90		-45	
		-135	-90	-45		
±180	±180	±180		0	0	0
		135	90	45		
	135		90		45	
135			90			45

Figura 3.6: Ángulos relativos al VCP del drone

²El VCP o *Vehicle Center Point* se define como el centro geométrico del vehículo. En nuestro caso, al tratarse de un drone, se considera el centro del mismo como VCP

Nótese que para llevar a cabo la obtención de estos ángulos **no** se ha hecho uso de la función *arctan* disponible en Numpy, dado que esta devuelve valores, en radianes, pertenecientes al 1º y 4º cuadrante únicamente. En su lugar, se ha utilizado la función *arctan2*, que tiene en cuenta el signo de las componentes del vector dirección $[y_i - y_0, x_i - x_0]$ resultante. De forma que devuelve el ángulo de giro, sea en sentido horario (valores positivos) o antihorario (valores negativos), más corto posible.³

La magnitud de cada zona es entonces calculada de la siguiente forma:

$$m_{i,j} = c *_{i,j}^2 * (a - b d_{i,j}) \quad (9)$$

Donde:

$m_{i,j}$ = Magnitud de obstáculos en la celda (i, j)

$c*_{i,j}$ = Certidumbre de obstáculo en la celda, en la región activa, (i, j)

y:

a = Representa la fuerza con que un obstáculo afecta al dron

b = Representa la distancia desde la que un obstáculo afecta al dron

son positivos

El plano es convertido en una secuencia de sectores que cubren los 360°. Se definen, por tanto, $n \in \mathbb{Z}$ sectores k de amplitud α , por ejemplo $n = 72; \alpha = 5$. Y se distribuyen las medidas tomadas anteriormente en los diferentes ángulos relativos al VCP del dron, en los sectores k_i correspondientes, véase 10

$$k = \frac{\beta_{i,j}}{\alpha} \quad (10)$$

con un valor h_k , véase 11, de densidad de obstáculos en ellos:

$$h_k = \sum m_{i,j} \quad (11)$$

Dada la discretización de los datos obtenidos del Polar Histogram el resultado, al realizar esta conversión y al obtener el sumatorio h_k de todos los valores pertenecientes a un sector k , puede parecer que el histograma varía de forma muy repentina entre dos sectores. Por ello se aplica una función de suavizado. Según J. Borenstein y Y. Koren la función de suavizado utilizada se corresponde con:

$$h'_k = \frac{h_{k-l} + 2h_{k-l+1} + \dots + lh_k + \dots + 2h_{k+l-1} + h_{k+l}}{2l + 1} \quad (12)$$

³Obviamente es lo deseable, no sería eficiente realizar un giro de 225° en el sentido de las agujas del reloj, cuando uno de 135° en el sentido contrario a las agujas del reloj bastaría.

Donde l es una constante positiva a la que se le ha dado un valor de 5.

Sin embargo, existe una errata en el paper publicado, y la ecuación 12 no es consistente, i.e: La función de suavizado parece poder expresarse de forma general tal que:

$$h'_k = \frac{1}{2l+1} \sum_{n=-l}^l (l - |n| + 1) * h_{k+n} \quad (13)$$

$$\text{ej: } l = 5 \quad h'_k = \frac{1h_{k-5}+2h_{k-4}+3h_{k-3}+4h_{k-2}+5h_{k-1}+6h_k+\dots+2h_{k+l-1}+h_{k+l}}{2l+1}$$

Como puede verse, el valor central $6h_k$ correspondiente a $n = 0$, no coincide con la posición dada para ese valor de n en la función de suavizado propuesta por J. Borenstein y Y.Koren, donde el coeficiente que multiplica a h_k es igual a l en lugar de $l + 1$.

Por ello, se ha hecho uso de una función de suavizado preexistente en SciPy, en concreto en el módulo *signal*. Se ha elegido la función de Hann, llamada así por el meteorólogo asutriaco Julius van Hann. Se conoce a esta función por el nombre de Campana de Coseno, y se define por la ecuación 14.

$$w(n) = 0,5 - 0,5\cos\frac{2\pi n}{M-1} \quad 0 \leq n \leq M-1 \quad (14)$$

Donde M es el número de puntos en la ventana de salida. En este caso tomará el valor de l .

De esta forma, de obtener un histograma de n sectores con valores h_k posiblemente dispares sectores k contiguos, se pasa a obtener un histograma suavizado, que será de mayor utilidad para el sistema de evasión de obstáculos. Véase la figura 3.7

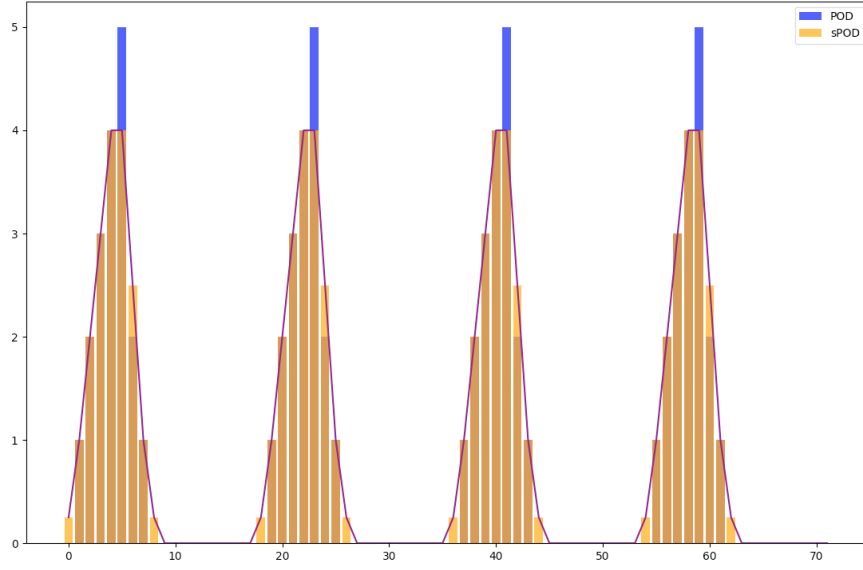


Figura 3.7: En azul Polar Obstacle Density(POD). En naranja el suavizado de este (sPOD).

Output Layer

En el último estadio del algoritmo VFH, se computa el ángulo de giro θ que se debe aplicar al dron para dirigirlo, por una ruta segura, hacia el destino establecido. Una vez obtenido el *Polar Obstacle Density*, y aplicado el suavizado correspondiente, se puede determinar en que sectores k existe un *valle*. Se define un *valle candidato* como una zona, compuesta de varios sectores por los que es seguro navegar, es decir, están libres de obstáculos o su densidad de obstáculos esta por debajo de cierto umbral.

Por lo general se crean dos o más valles candidatos al analizar el entorno, de forma que es necesario elegir aquel que dirigirá al dron en dirección al destino k_{targ} . Una vez escogido el valle, se deberá seleccionar el sector k idóneo. Para ello se mide el tamaño del valle, es decir el número de sectores consecutivos por debajo del umbral que lo componen, de esta forma se distinguen dos tipos de valles, amplios y estrechos. Los valles amplios son el resultado de espacios grandes entre obstáculos, o de situaciones en las que únicamente existe un obstáculo lo suficientemente cerca del dron. Por *grande* se define una constante s_{max} que determina el número de sectores k que componen un valle amplio.

El sector más cercano a k_{targ} y por debajo del umbral se denomina k_n , y representa el *borde cercano* del valle. El más lejano k_f que en el caso de los valles amplios coincide con el valor de $s_{max} + k_n$. Véase la imagen 3.8.

El sector que debe seguir el dron para dirigirse hacia k_{targ} sin peligro, se

denomina θ_{yaw} , y su valor es la media de las medidas de k_n y k_f , como puede verse en la ecuación 15.

$$\theta_{yaw} = \frac{k_n + k_f}{2} \quad (15)$$

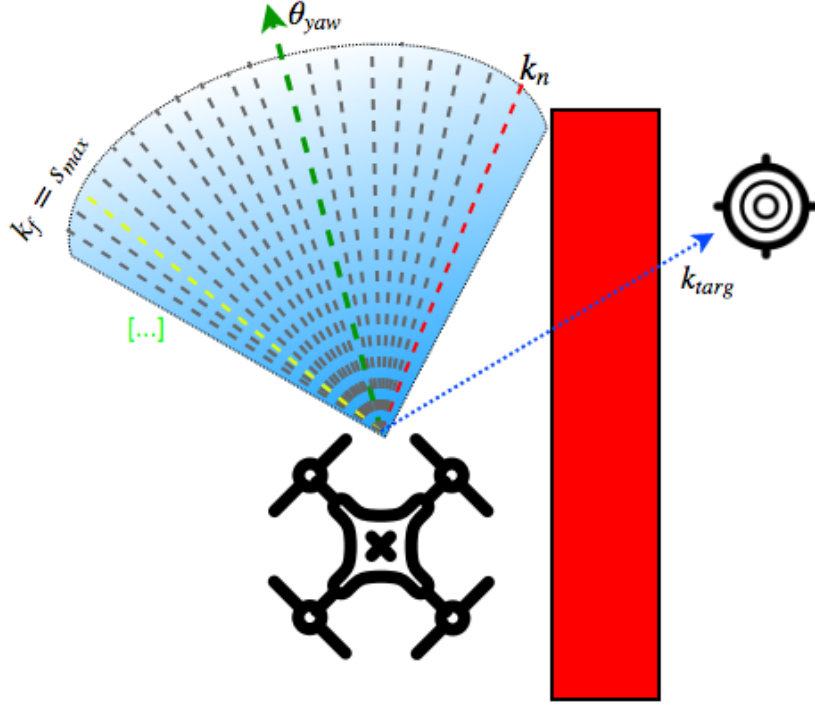


Figura 3.8: Valle ancho encontrado entre el obstáculo, a la derecha, y un espacio abierto, a la izquierda.

Sin embargo, la ecuación 15 presenta un comportamiento errático cuando el destino se encuentra dentro de un valle, dado que se establece θ_{yaw} como la media entre los sectores cercano y lejano, al encontrarse la meta en un sector dentro del valle, el drone no se aproxima a ella sino que realiza círculos cada vez más cerrados hasta llegar al destino, como puede verse en la imagen 3.9

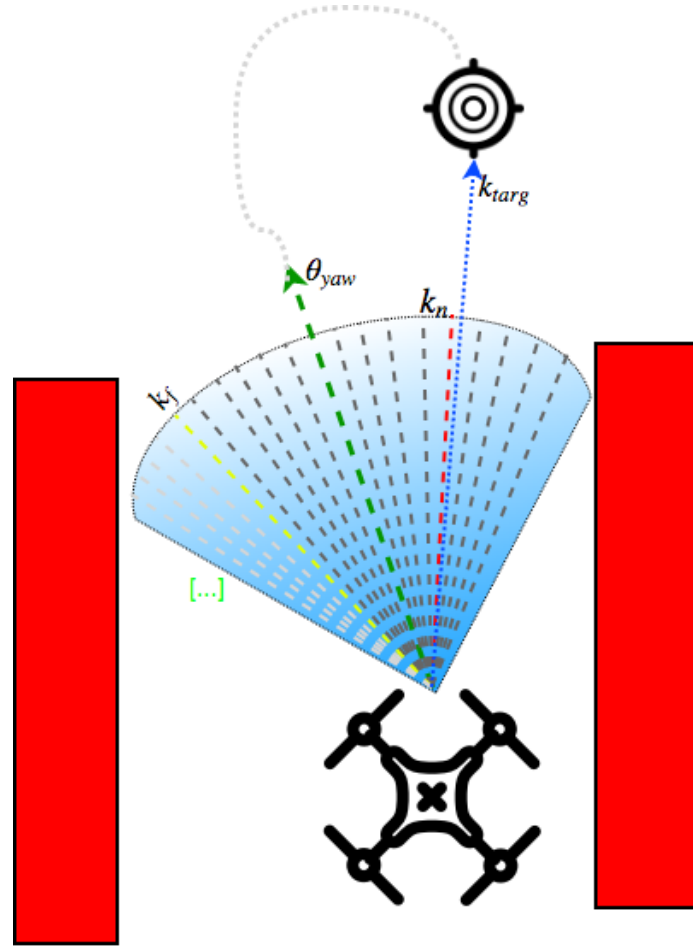


Figura 3.9: Movimiento circular al encontrarse k_{targ} en un sector seguro por ecuación 15

La intención tras la ecuación 15 es la navegación segura en entornos con obstáculos cercanos, ya que aporta la cualidad de escoger la zona central entre dos obstáculos para usarla como zona de navegación segura⁴. Además en el momento en que la posición del dron sea intermedia entre dos obstáculos, la ecuación 15 dará como resultado el mismo sector, con lo que se consigue un movimiento rectilíneo.

Sin embargo en el momento en el que el destino k_{targ} se encuentra en un sector k por el que es seguro navegar, no existe la necesidad escoger la zona intermedia del valle, sino que es deseable dirigirse hacia él directamente.

⁴Una cualidad de la que el algoritmo de campos potenciales adolecía, generando movimientos oscilatorios en espacios cerrados.

En el caso de *valles estrechos*, la mecánica es prácticamente la misma. Para este caso el valor del último sector con un POD por debajo del umbral escogido será $k_f < s_{max}$. De nuevo se utiliza la ecuación 15 para obtener el valor de θ_{yaw} , siempre centrado entre los dos obstáculos, ilustrado en la imagen 3.10.

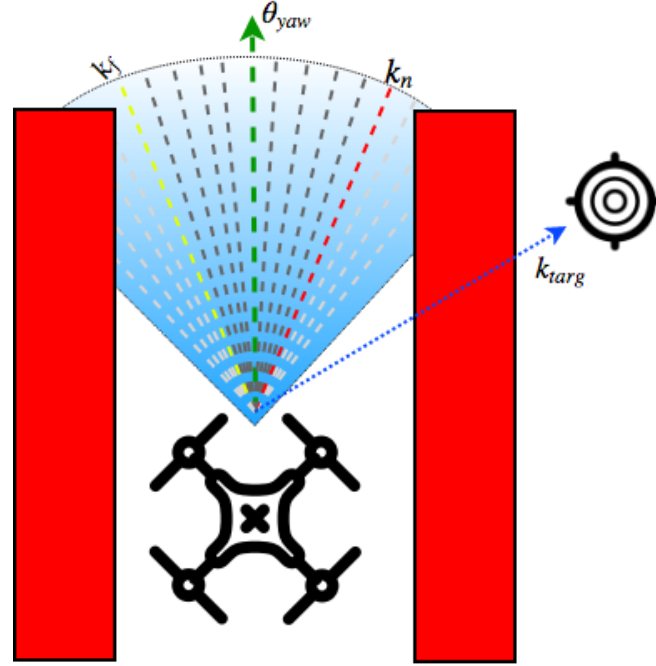


Figura 3.10: Valle estrecho encontrado entre dos obstáculos. La ecuación 15 proporciona un sector seguro ubicado en la zona central del valle.

Se ha referenciado numerosas veces el *umbral* que parece definir un valle como candidato. Su definición se ha dejado para el final debido a que, el VFH presenta una gran robustez ante configuraciones probablemente erróneas. Aumentar o reducir el umbral, únicamente, afecta a la anchura de los valles candidatos cuando estos son estrechos. En el caso de los valles anchos simplemente se aumenta o reduce la distancia existente entre el obstáculo y el dron.

Por ejemplo, establecer un umbral muy alto hace que el dron no sea consciente de la existencia de obstáculos y que se aproxime a ellos. Sin embargo las repetidas medidas de un obstáculo en ese sector, harán que el valor de certidumbre del Polar Histogram se eleve hasta superar el umbral, produciendo que el dron trate de variar su curso. De esta forma el dron se acercará mucho a los obstáculos y es posible que llegue a colisionar si la velocidad de aproximación es demasiado rápida. Por el contrario, si el umbral es muy bajo,

hará que los valles candidatos se vean reducidos, haciendo que el drone no pueda pasar por espacios estrechos.

3.2. Dispositivos Físicos

Raspberry Pi

Componente	Raspberry Pi 3B
CPU	BCM2837
Núcleos	4
Velocidad	1.2GHz
RAM	1GB
Coms	Ethernet, WiFi, Bluetooth
USB	4 (2.0)
GPIO	40
Consumo máximo	6.7W

Tabla 3.1: Componentes de una Raspberry Pi 3 Model B

Una Raspberry Pi es un pequeño ordenador desarrollado en UK por la Raspberry Pi Foundation, con la intención de promover el aprendizaje de informática básica en colegios y países en desarrollo. El modelo base se compone de una única placa de medidas 85mm x 56mm (LxA), y unos 42g de peso.

Concretamente el modelo empleado es una Raspberry Pi 3B, y consta de los componentes detallados en la tabla 3.1

Su reducido tamaño y bajo consumo lo hacen ideal para este tipo de proyecto. En este caso se utiliza bajo una distribución GNU/Linux llamada Raspbian⁵, basada en Debian. Para tratar de mejorar su rendimiento y reducir al mínimo el consumo, se ha escogido la versión Lite del sistema, es decir, un sistema mínimo sin entorno de escritorio y con la mayor parte de servicios desactivados por defecto.

Una vez descargado el sistema, este ha sido instalado en una micro SD mediante el siguiente procedimiento por consola⁶:

- `diskutil list` Permite localizar el dispositivo en el que se encuentra la tarjeta. En nuestro caso `/dev/disk4`
- `diskutil umountDisk /dev/disk4` Permite desmontar el volumen.

⁵Descargable desde: <https://www.raspberrypi.org/downloads/raspbian>

⁶Realizado en OSX, aunque en Linux es muy similar

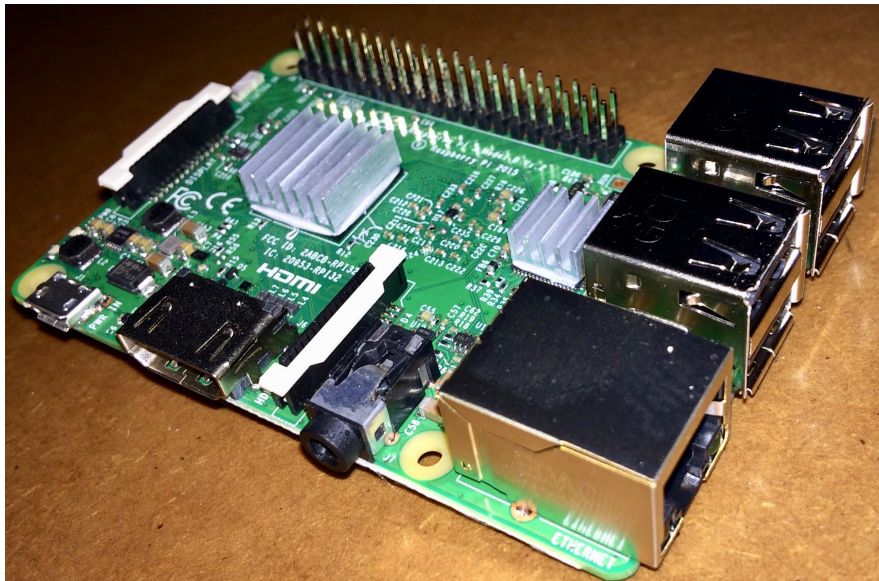


Figura 3.11: Raspberry Pi 3.

- `sudo dd if=raspbian-stretch.img of=/dev/rdisk4 bs=1m` El comando `dd` copia la entrada estándar a la salida estándar. Mediante `if/of` se establece el fichero de entrada/salida. Mediante `bs` se establece el tamaño de bloque a copiar. Se está utilizando `/dev/rdisk4` en lugar de `/dev/disk4` debido a la capacidad de OSX de trabajar con dispositivos en bruto, *raw*, de forma que es posible acceder al dispositivo de forma directa⁷, sin almacenar en un buffer la lectura del archivo, proporcionando velocidades de escritura/lectura hasta 20 veces más rápidas.

Una vez realizados estos pasos, se puede insertar la microSD en la Raspberry Pi. Para encenderla basta con utilizar el puerto micro-usb de que dispone. La Raspberry Pi 3 requiere de una fuente de alimentación capaz de proporcionar 2,5A⁸ para funcionar al máximo nivel de estrés para el procesador y alimentar dispositivos USB.

Sin embargo, este no es estrictamente nuestro caso, véase el apartado **Modificaciones**. Se requiere un dispositivo cuyo consumo sea lo más reducido posible, pero que sea rápido en la ejecución, y que muestre poca latencia en operaciones de `IO`, que es donde se encuentra el cuello de botella.

⁷Véase `man hdiutil`, sección *DEVICE SPECIAL FILES*

⁸Véase <https://www.raspberrypi.org/help/faqs/#power>

```

CONFIG:
CLOCK : 100.000 MHz
CORE  : 400 MHz, turbo=0
DATA  : 512 MB, /root/test.dat

HDPARM:
=====
Timing 0_DIRECT disk reads: 108 MB in 3.02 seconds = 35.78 MB/sec
Timing 0_DIRECT disk reads: 108 MB in 3.02 seconds = 35.72 MB/sec
Timing 0_DIRECT disk reads: 108 MB in 3.02 seconds = 35.75 MB/sec

WRITE:
=====
536870912 bytes (537 MB, 512 MiB) copied, 21.7232 s, 24.7 MB/s
536870912 bytes (537 MB, 512 MiB) copied, 20.7831 s, 25.8 MB/s
536870912 bytes (537 MB, 512 MiB) copied, 21.6338 s, 24.8 MB/s

READ:
=====
536870912 bytes (537 MB, 512 MiB) copied, 14.3944 s, 37.3 MB/s
536870912 bytes (537 MB, 512 MiB) copied, 14.3785 s, 37.3 MB/s
536870912 bytes (537 MB, 512 MiB) copied, 14.3567 s, 37.4 MB/s

```

Figura 3.12: Benchmark de lector microSD OC.

Una vez encendida, se accede a ella con el usuario por defecto `pi` y la contraseña por defecto `raspberrypi`. Obviamente ambas **han sido cambiadas** por motivos de seguridad.

Modificaciones

- Se ha desactivado el puerto HDMI para reducir el consumo en ~30mA:
Para ello se ha incluido en `/etc/rc.local` la línea `/usr/bin/tvservice -o`.
- Se ha overlockeado el lector de microSD a 100MHz, en lugar de los 50MHz por defecto:
Para ello se ha incluido en `/boot/config.txt` la línea `dtoverlay=sd_overclock=100`.
Y que arroja los resultados mostrados en la imagen 3.12
- Se han incluido una serie de disipadores para evitar sobrecalentamiento de la placa. Así como un pequeño ventilador de bajo consumo.

Controladora de Vuelo

Una controladora de vuelo (*FC* de aquí en adelante) es un pequeño circuito integrado, que contiene un procesador, una serie de sensores, y una serie de entradas y salidas. La FC se encarga de mantener el sistema de estabilización del dron, tomando medidas de los sensores de que dispone, tales como un acelerómetro, giroscopio, magnetómetro, barómetro... etc.

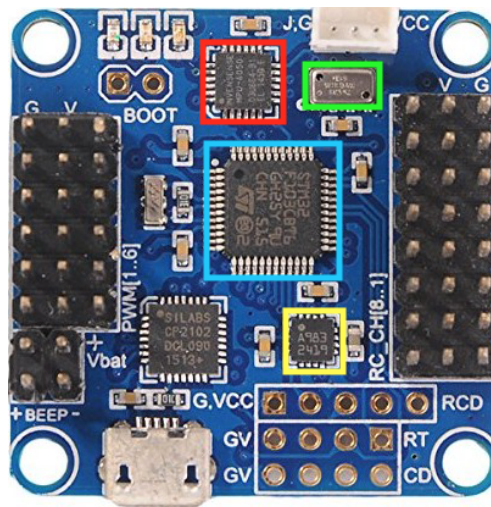


Figura 3.13: Controladora de Vuelo Flip32.

En el caso de la FC usada para este proyecto, llamada Flip32 y mostrada en la imagen 3.13, se dispone de un IC MPU-6050, en rojo, con acelerómetro y giroscopio, así como de un barómetro M55611, en verde, y un magnetómetro HMC5883L, en amarillo. El núcleo de esta pequeña placa es un procesador STM32F103, en cyan, a 72MHz y basado en arquitectura ARM el cual dispone de dos puertos serie, que permiten establecer comunicación entre la FC y otros dispositivos, como emisoras u otros sensores.

El puerto micro-USB de que dispone es utilizado para establecer comunicación entre el configurador de opciones del drone, o en el caso de nuestro proyecto, para hacer uso del [MultiWii Serial Protocol](#).

Entradas

En el lado derecho de la imagen 3.13 pueden verse una serie de pines que actúan como 8 canales de entrada desde el receptor de radio. Dichos canales de entrada, por defecto, reciben una señal modulada en ancho de pulso o **PWM**

Salidas

En el lado izquierdo de la imagen 3.13, pueden verse otros pines que actúan como salida de señal hacia los controladores de velocidad de los motores del drone (ESC o *Electronic Speed Controller*). Estos pines de salida, emiten una señal que será interpretada por los ESC del drone, para determinar la frecuencia dada al voltaje que alimenta los motores. En el caso de nuestro proyecto, los ESC disponibles reciben una señal PWM, al igual que la FC desde un receptor de radio.

Sensores

La controladora de vuelo Flip32 en su versión más completa, dispone de los siguientes sensores:

- IMU⁹ MPU-6050: Se trata de un circuito integrado compuesto de un acelerómetro de tres (3) ejes y un giroscopio de tres (3) ejes. El acelerómetro mide las fuerzas en los tres diferentes ejes (en g), el giroscopio se encarga de medir la velocidad angular en cada uno de los tres ejes (en $\text{deg}^\circ/\text{s}$)
- Magnetómetro HMC5883L: Se trata de un pequeño magnetómetro digital capaz de medir el campo magnético terrestre (en Gauss). Se debe tener en cuenta que según la posición en el planeta, el campo magnético varía entre $G0,25$ - $G0,65$, así como la declinación magnética de la zona en la que se realiza la medición.¹⁰
- Barómetro M55611: Se trata de un altímetro de alta precisión, con resoluciones de hasta 10cm, que funciona midiendo la presión atmosférica (en milibar). Hay que tener en cuenta que los cambios de presión, como los generados por las hélices del drone, hacen variar la medida del sensor. Por ello, en el caso de usarlo, se cubre con un pequeño filtro de un material absorbente, lo suficientemente denso como para evitar el contacto directo entre una corriente de aire y el sensor.

⁹Unidad de Medición Inercial.

¹⁰Disponible en: <http://magnetic-declination.com/>

3.3. Protocolos

Pulse Width Modulation

La modulación por ancho de pulso, se basa en medir el transcurso de tiempo entre el flanco de subida de una señal, y el flanco de bajada, tal y como puede verse en la imagen 3.14. En este contexto, un receptor de radio recibe una señal de la emisora, y genera una señal PWM acorde que será transmitida a la FC. Estas son capaces de entender señales de entre 1000 y 2000 μ s. Cualquier valor por debajo, o por encima, haría entrar la controladora en FailSafe¹¹ En el caso de este proyecto, se ha determinado que no se hará uso de este protocolo para la comunicación entre la Raspberry Pi y la controladora de vuelo, véase ?? y 3.3, y por lo tanto el uso de estos pines queda descartado.

Sin embargo, la comunicación entre la FC y los ESC se realiza mediante este tipo de señal, por ello se ha considerado relevante explicar, brevemente, su funcionamiento.

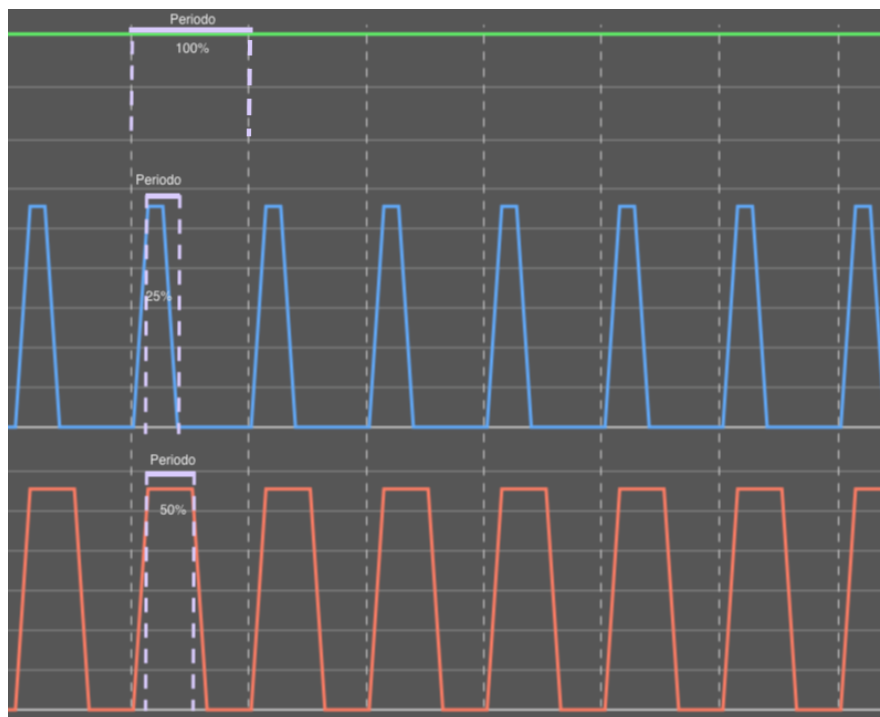


Figura 3.14: Modulación en Ancho de Pulso. Arriba 100 % del ciclo usado. Centro 25 % del ciclo usado. Abajo 50 % del ciclo usado

¹¹ Al recibir una señal inválida por parte del receptor, la controladora de vuelo puede ser configurada para desactivar el drone, mantener la última medida buena conocida, intentar aterrizar... etc. Este modo es conocido como FailSafe

MultiWii Serial Protocol

MultiWii es un software de control de multirrotores y está basado en componentes de la Nintendo Wii. Concretamente, los mandos de control de la consola de Nintendo poseen tres acelerómetros para determinar la posición angular y medir aceleraciones laterales. El problema es que los acelerómetros no son precisos para variaciones pequeñas, de forma que Nintendo creó un complemento que se podía conectar al propio mando, el Wii Motion Plus, que dispone de tres giroscopios, de forma que unidos a los tres acelerómetros iniciales, proporcionan una medición mucho más precisa de la posición del mando.

A los primeros creadores del sistema de control MultiWii se les ocurrió la posibilidad de hacer uso de estos sensores para crear un sistema de control de drones. Uniendo estos sensores a un controlador, en principio se trató de un Arduino Mini, y programándolo para tal efecto, se logra crear una FC algo rudimentaria, pero que da muy buenos resultados.

Para crear un entorno amigable para el usuario común, se desarrolló una aplicación de escritorio capaz de configurar los parámetros de esta controladora de vuelo poco convencional. De alguna forma debía lograrse una comunicación entre la FC y la aplicación, y así se implementó MultiWii Serial Protocol. Conocido como *MSP*, se trata de un protocolo que se ha mantenido en diferentes implementaciones de sistemas de control de vuelo. No solo aquellos basados en sensores de la Nintendo Wii controlados por Arduino, sino en otros más modernos y potentes que han ido surgiendo los últimos años. Como el utilizado en nuestro proyecto.

Es un protocolo eficiente y sencillo, que permite una comunicación rápida y completa. Su composición puede verse en la imagen 3.15

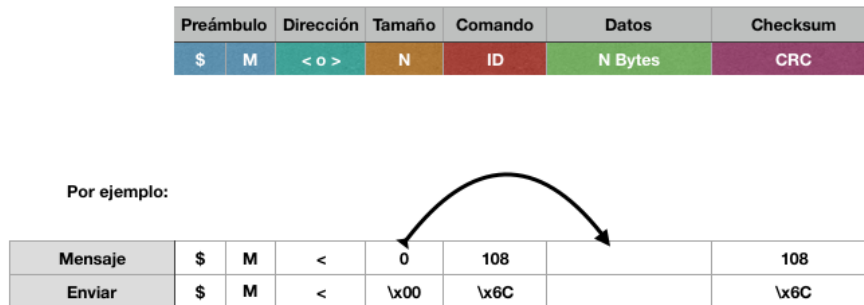


Figura 3.15: Composición de un mensaje MSP

- **Preámbulo:** Se trata de dos caracteres ASCII que determinan el comienzo de un nuevo mensaje. Se compone de un símbolo '\$' y una letra 'M'.
- **Dirección:** Se trata de un carácter ASCII que determina la dirección del mensaje, **hacia** la controladora '<' o **desde** la controladora '>'.
- **Tamaño:** Se trata de un byte que determina el tamaño, en bytes, de los datos enviados o recibidos.
- **Comando:** Se trata de un byte que determina el comando a ejecutar. Véase ?? para una relación de los comandos implementados.
- **Datos:** Se trata de una serie de bytes, de longitud definida por el byte <tamaño>, que establecen el resto de parámetros a pasar con la función definida por el byte <comando>.
- **Checksum:** Se trata de un byte de control. Se define su valor mediante la función XOR entre <tamaño>, <comando> y cada byte contenido en <datos>

En el caso de nuestro proyecto, se hará uso de este protocolo dado que existe la posibilidad de establecer la recepción de los canales de radio a través de un puerto serie, así como de solicitar información sobre el estado del dron a la FC. Es decir, en lugar de utilizar un receptor de radio, se utilizará un puerto serie para obtener la telemetría¹², y establecer las entradas de los canales de radio. En la tabla 3.2 se dispone de los mensajes implementados para la realización de este proyecto.

¹²Se define telemetría como un sistema de medición de magnitudes a distancia; en este contexto el término se ha desvirtuado y se entiende como la información que es transmitida de vuelta a la emisora de vuelo, siendo esta generalmente información de los sensores de la FC, GPS o voltaje disponible en la batería.

Mensaje	Código	Estructura	Parse	Descripción
MSP_MOTOR	104	8x UINT16	<8H	Devuelve la señal PWM que los ESC envían a los motores.
MSP_RC	105	18x UINT16	<18H	Devuelve la señal PWM disponible en cada canal.
MSP_ATTITUDE	108	3x INT16	<3h	Devuelve las inclinaciones de los ejes X e Y, así como la orientación.
MSP_ANALOG	110	1x UINT8 3x UINT16	<B3H	Devuelve el estado de los sensores incluidos en la FC: voltaje de la batería, RSSI, corriente instantánea y consumo
MSP_SET_RAW_RC	200	Nx UINT16	N/A	Establece el valor de los canales de radio-control.
MSP_SET_RAW_MOTOR	214	Nx UINT16	N/A	Establece la señal PWM a enviar a los motores.

Tabla 3.2: Mensajes MSP. Estructura de datos y representación

La columna *Mensaje* sigue esta nomenclatura por razones históricas, para mantener cierta coherencia con la tabla disponible en la descripción del protocolo MultiWii,[?]. La columna *Código* establece el número de comando enviado. Los comandos que empiezan por 1, se corresponden con solicitudes de información, y aquellos que comienzan por 2 son órdenes. Este valor será utilizado por la FC para establecer el parseo de la información enviada, de ser alguna. La columna *Estructura* establece el tipo y la cantidad de cada dato que se envía o recibe. La columna *Parse* establece el tipo de parser que se aplicará a la respuesta recibida de la FC. Nótese que solo se define el parser para los comandos que comienzan por 1, es decir, para las solicitudes de información. La información del parser proporciona la manera de decodificar la cadena de bytes devuelta por la FC al realizar una solicitud. Para ello se ha seguido la nomenclatura utilizada en el módulo *struct* de Python 3.5, descrita en la [tabla 3.3](#)

Los enteros sin signo representados en la [tabla 3.3](#), tienen su equivalencia a enteros con signo mediante el mismo caracter en minúscula. La implementación realizada sigue un paradigma funcional, de forma que la función utilizada para leer las respuestas de la FC puede ser utilizada con nuevas solicitudes de información, con tan solo pasar un nuevo parser en forma de cadena de texto.

Elemento	Descripción
'<', '>'	Little o Big Endian respectivamente. Establece la ubicación del byte menos significativo, y por lo tanto determina la dirección de lectura de los grupos de bytes de la cadena recibida.
'B'	Caracter utilizado para representar un entero de un byte sin signo.
'H'	Caracter utilizado para representar un entero de dos bytes sin signo.
'c'	Caracter utilizado para representar un caracter de un byte.

Tabla 3.3: Nomenclatura del módulo Struct de la implementación 3.5 de Python

Secure SHell

Secure SHell o *SSH* de aquí en adelante, es un protocolo de red cifrado el cual se basa en el uso de claves públicas compartidas para crear un canal de comunicación seguro en una red no segura. Al aceptar una conexión, el servicio SSH presenta una shell sobre la que el cliente puede realizar las operaciones necesarias y propias de su nivel de privilegio. Además proporciona la posibilidad de redirigir el sistema de ventanas remoto al cliente, aunque en este caso se está haciendo uso de una versión reducida del SO, y por lo tanto no presenta entorno de escritorio. El acceso a la Raspberry Pi que controla el drone, se lleva a cabo mediante el uso de este protocolo, siguiendo los siguientes pasos para su configuración:

- Generar el par de claves pública-privada en el cliente mediante el comando `ssh-keygen`
- Copiar la clave pública del cliente en el archivo `.ssh/authorized_keys` de la carpeta `home` del usuario a utilizar en el sistema remoto.
- Editar el archivo `/etc/ssh/sshd_config` para:
 - permitir unicamente acceso a usuarios que envíen una clave pública contenida en `authorized_keys`.
 - desactivar el acceso al usuario root, estableciendo la propiedad `PermitRootLogin` a `no`.
 - desactivar el acceso por contraseña, estableciendo la propiedad `PasswordAuthentication` a `no`.

De esta forma se logra dotar de acceso seguro a un cliente autorizado. Al tratar de conectar al sistema de control del drone, se muestra un banner

en el que se advierte a usuarios malintencionados de la existencia de un log de conexiones recibidas. Para tratar de mitigar los intentos más comunes de intrusión en el sistema, algo tan simple como cambiar el puerto en el que responde el servidor SSH, se ha mostrado tremendamente efectivo contra los ataques automatizados más simples y comunes.

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Si se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas se puede hacer un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas. No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas, sino comentar los aspectos más destacados de cada opción, con un repaso somero a los fundamentos esenciales y referencias bibliográficas para que el lector pueda ampliar su conocimiento sobre el tema.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros³, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.