



Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México

Escuela de Ingeniería y Ciencias

Integración de Robótica y Sistemas Inteligentes

**Grupo 501
Equipo 11 - La Voluntad**

Evidencia:

Navegación reactiva con VANTS cuadricópteros

Profesor:

Dr. Alejandro Aceves López
Dr. Aldo Iván Aguilar Aldecoa
Dr. Miguel Ángel Galvez Zuñiga
Dr. Alf Kjartan Halvorsen
Dr. Francisco Javier Ortiz Cerecedo
Dr. Arturo Vargas Olivares

Alumno

Bruno Sánchez García	A01378960
Carlos Antonio Pazos Reyes	A01378262
Manuel Agustín Díaz Vivanco	A01379673

Atizapán de Zaragoza, México a 7 de junio de 2023

Introducción

Se le denomina Vehículo Aéreo No Tripulado (VANT) a cualquier máquina capaz de volar sin ser operada por un piloto a bordo de ésta. Aunque este concepto, junto con su nombre coloquial (dron) son términos modernos, se considera que los primeros registros del uso de vehículos aéreos no tripulados datan de los últimos 10 años del siglo 19, donde nacía en Austria la estrategia militar de emplear globos cargados con bombas como un bombardeo no dirigido sobre la ciudad de Venecia. Con el paso de los años, esta idea se fue empleando cada vez más. Durante la Segunda Guerra Mundial y más adelante en la Guerra Fría, se emplearon aviones no tripulados para tareas de reconocimiento.

Hoy en día existen *drones* en todos lados. Se emplean desde la vigilancia y la industria militar para realizar entregas de paquetería, fotografía, video y probablemente el uso más conocido es del sector de entretenimiento con los VANTS para vuelos recreacionales o los espectáculos de luces. Debido a lo anterior, para su construcción se requiere una correcta integración de software y hardware dentro de una estructura ligera y aerodinámica que permita que este pueda volar por un espacio establecido y seguir un propósito predefinido.

Los *drones* se clasifican en varios tipos dependiendo su proceso de despegue, estructura y dinámica de vuelo. Entre los más conocidos se encuentran los VANT de *ala fija* y los *multirrotores*. Los VANT de ala fija son cuerpos alargados que cuentan con dos alas, alerones, motores frontales y/o traseros que en conjunto permiten que estos vuelen y planeen acorde a la dirección de frente, lo que los hace capaces de alcanzar velocidades muy altas. Al compartir similitudes con los aviones militares y comerciales, estos deben ser controlados tomando en cuenta la misma dinámica que los aviones. Los multirrotos por su parte, son vehículos con una estructura en forma de estrella que emplean por lo general de 3 a 8 hélices al final de cada una de sus extremidades, los cuales ejercen una fuerza de empuje hacia arriba que al ser mayor que la fuerza de la gravedad, este pueda desplazarse hacia arriba, mantenerse en cierta posición en el aire cuando la fuerza de empuje (thrust) y la gravedad se cancelen, y finalmente descender cuando se disminuye el *thrust* paulatinamente hasta que el VANT toca el suelo.



Figura 1: Drone de ala Fija (fuente: Internet)



Figura 2: Drone Multirotor (fuente: Internet)

Imagen I: Ala fija vs multirrotos (Villalobos A., 2019)

Los multirotores a su vez, se clasifican en varios subgrupos dependiendo de la cantidad de hélices y su configuración. Los tricópteros cuentan con 3 hélices y una estructura o *frame* en forma de la letra “Y”. Los cuadricópteros cuentan con 4 hélices y tomando como referencia su frente pueden tener un *frame* con forma de cruz, letra “X” o letra “H”. Los hexacópteros (6 hélices) y octacópteros (8 hélices) de forma similar a los cuadricópteros, tienen un frame que cuenta con el frente perpendicular o paralelo a una de las extremidades. Adicionalmente, los hexacópteros y octacópteros también cuentan con dos motores uno encima del otro en cada extremidad, lo que lleva a tener hexacópteros con forma de “Y” y octacópteros con forma de “X”, esta configuración se le denomina coaxial.

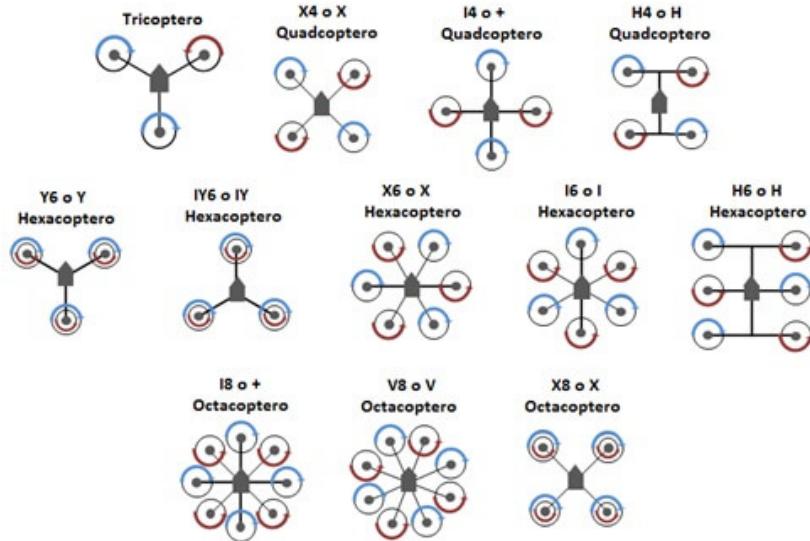


Imagen 2: Tipos de drones multirotor. (Imagen recuperada del blog de la empresa Jugetnic, 2017)

Descripción del reto

Se requiere implementar un algoritmo de navegación reactiva que, haciendo uso de visión computacional, permita que un VANT multirotor cuadricóptero se mueva por un entorno de forma autónoma. El programa propuesto es que el robot debe ser capaz de seguir un objeto de referencia que le permita ajustar su posición en el plano tridimensional para alinearse con el objeto. El VANT debe ser capaz de evadir obstáculos de un color específico que se encuentren dentro de su rango de visión y finalmente aterrizar al final de su trayecto. El algoritmo debe ser robusto para filtrar ruido y permitir que el vehículo despegue, realice su tarea de forma eficiente y finalmente aterrice de forma controlada sin sufrir daños.

El algoritmo deberá ser implementado en uno de los drones SK450 con los que cuenta la escudería de robótica del Tecnológico de Monterrey Campus Estado de México. Dicho VANT cuenta con las siguientes características:

- Frame tipo SK450.
- Kit conversor Death Cat para frame de cuadricóptero
- 4 Motores Brushless Multistar 2213-980

- 4 ESC AfroESC 20A
- 4 Hélices HobbyKing 10 x 4.5 in
- GPS Here 2
- Receptor de radio control Turnigy IA8 2.4 Ghz 2A
- Mando RC Turnigy 9x
- Altímetro Betterware Micro LiDAR TFMini
- Controlador de vuelo Pixhawk Cube Black 2.1
- Batería LiPo Multistar 3S 11.1V 10C
- Power Distribution Board generica
- Carrier Board Odroid XU4
- Webcam Microsoft LifeCam USB 1080p



Imagen 3.1: Fotografía del dron SK450

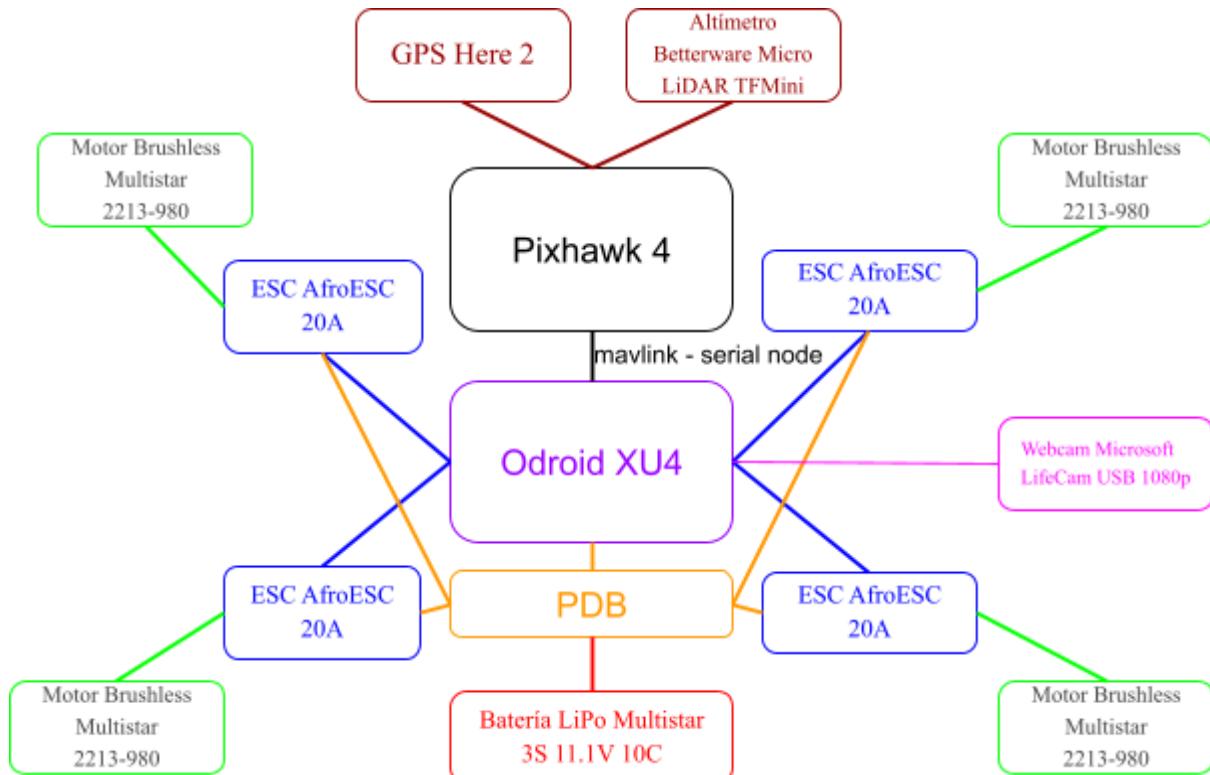


Imagen 3.2: Diagrama bloques del VANT SK450 ensus componentes electrónicos

Con el objetivo de darle autonomía al VANT y que este no dependa de comandos enviados mediante radiofrecuencia, se requiere actualizar la versión de ROS dentro de la *Carrier Board* con la que cuenta el dron. De esta forma el robot será capaz de navegar de forma autónoma o dirigida mediante una red de área local donde se comuniquen la *Carrier Board* y la computadora de control en tierra.

Solución elegida

Dentro del software utilizado, se optó por emplear la distribución de ROS *Melodic Morenia* que venía instalada la *Odroid*, junto con el paquete de MAVROS, el lenguaje de programación Python 2.7 para la creación de nodos y la librería de OpenCV para la visión computacional. Se decidió emplear MAVROS ya que contiene servicios, mensajes y nodos que son necesarios para la recepción y envío de mensajes mediante el protocolo MAVLink que es ampliamente usado para la comunicación con vehículos no tripulados como VANTS, Rovers, Hovers, entre otros. Particularmente, MAVROS permite establecer una comunicación entre la *Carrier Board* y el controlador de vuelo Pixhawk Cube Black mediante comunicación serial. Por su parte, se decidió emplear Python ya que este lenguaje de programación es de los más usados en la industria de la robótica debido a su cantidad de recursos disponibles para los cálculos matemáticos y la programación a alto nivel. De forma similar, se empleó OpenCV para los algoritmos de visión computacional ya que esta facilita el trabajo con imágenes al proporcionar herramientas como funciones para el filtrado de imágenes, reducción de ruido y detección de objetos, los cuales son requerimientos claves

para la resolución del reto, además que la documentación es extensa y completa. Como herramientas para probar los algoritmos, se emplearon las simulaciones incluidas en el paquete que proporciona el fabricante de controladores PX4 (diseñadores del Pixhawk Cube Black), así como la librería de la empresa de Hector modificada con contribuciones de la comunidad (entre ellas la realizada por el Dr Alejandro Aceves López), la cuál permite hacer uso de este paquete de simulaciones con la distribución ROS Melodic de forma simple y práctica.

Desarrollo de la solución

Empleando los recursos mencionados previamente, se comenzó por descargar las librerías y paquetes para poder tener un modelo virtual con un comportamiento similar al VANT real.

Simulaciones

Para desarrollar la solución, primero entendimos el hardware y el funcionamiento de software dentro del robot. Como el tiempo de vuelo de un dron es muy limitado, además que es dependiente a disponibilidad de permisos y condiciones meteorológicas aceptables, pero sobre todo el realizar pruebas físicas directamente con el robot sin una prueba previa en simulación es muy peligroso, nos dedicamos a desarrollar las soluciones primeramente en simulación.

Utilizamos dos simuladores:

- Simulación PX4: proveída por el manufactor Pixhawk para probar VANTs y otros robots que utilizan su hardware. Utilizamos este simulador principalmente para conocer los tópicos, servicios, nodos y funcionamiento de MAVROS en un controlador PX4. Este simulador cuenta con varios cuadricópteros, y puede utilizarse en varios simuladores; nosotros utilizamos JmavSim y Gazebo con el cuadricóptero default que viene incluido. Este es el más apegado en cuestión de software a nuestro dron físico. A la par de este simulador, se acostumbra usar el software QGroundControl como programa de comunicación con tierra (*Ground Control System - GCS*) que provee información importante como la ubicación satelital, trayectoria y estado del dron. En pruebas físicas, nuestro sistema de comunicación terrestre es mediante un enrutador que levanta una red, con asignación de dirección IP estática para el dron, y conectando la computadora a la misma red.



Imagen 4: Simulador JmaSim (derecha) y QGroundControl (izquierda) con cuadricóptero default

- Hector en Gazebo: ya directamente implementado en ROS, nos ofrece una variedad de opciones, materiales y cuadricóptero para hacer pruebas en un simulador del cuál tenemos más conocimiento. Principalmente para implementar algoritmos vistos con el Puzzlebot en el dron. Hector provee directamente un cuadricóptero con cámara y con sensor lidar, material necesario para implementar algoritmos de navegación reactiva.

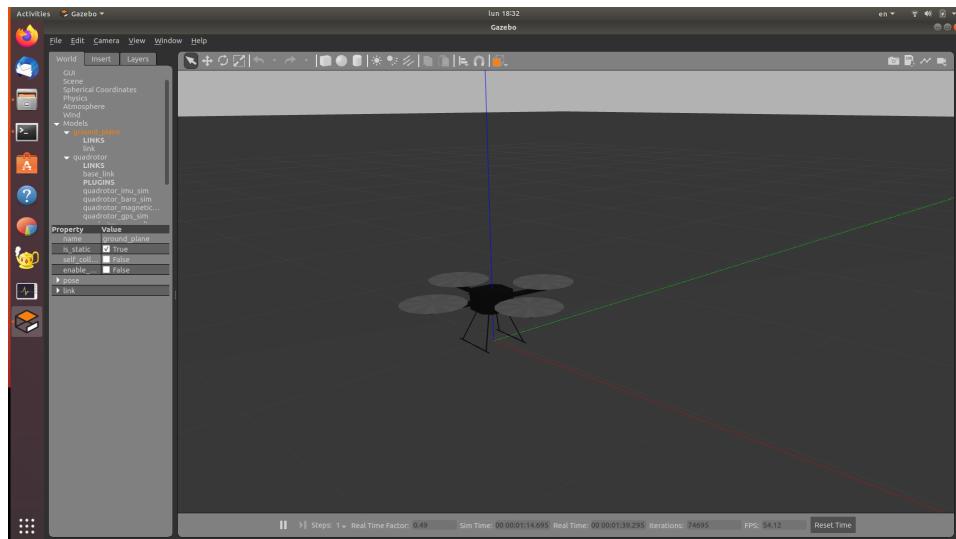


Imagen 5: Simulación cuadricóptero con Hector en Gazebo

Hover y Trayectorias

Usando simulador PX4 logramos hacer el primer hover, con base en el código proporcionado directamente en la página del simulador. En esencia, se aprovecha el potencial computacional del controlador del vuelo para hacer lazos cerrados. La lógica de este primer programa es publicar constantemente en el tópico `/mavros/setpoint_position/local` que manda el dron al punto publicado. Para hacer hover, simplemente se debe mandar constantemente un punto con alguna altura superior a 0. Este tópico refiere a *local* debido a que el *frame* de referencia es el del dron al ser iniciado, que en simulación es 0 en todos sus ejes como su origen. Su complemento es *global* cuya referencia es su posición geosatelital. Entonces un hover es simplemente publicar una pose con valores 0 en x, y, pero con algo mayor a 0 en z; el controlador se encargará con base en la información de sus sensores el controlar el robot para mantener la posición publicada.

Además de este tópico, destacan la lógica de armado y modo de vuelo. El dron necesita tener un *setpoint* constantemente (a un *rate* superior a 2Hz para ser más específicos) pues sino el robot entra a un estado *Position* (al menos en simulador, en físico preferimos no probamos este comportamiento) que provoca un aterrizaje ya que es un vuelo sin objetivo. A la par, antes de poder mandarle un *setpoint* definido a dónde dirigirse, el robot debe tener el modo de vuelo *GUIDED* y estar armado. Esta información la podemos obtener del tópico `/mavros/state`. Para cambiar el modo de vuelo se utiliza el servicio `/mavros/set_mode`, y para el armado del dron se usa el servicio `/mavros/cmd/arming`.

De esta forma, surge la lógica de un nodo que haga una trayectoria con base en publicación de *setpoints* como los puntos meta. La lógica se implementa en una máquina de estados:

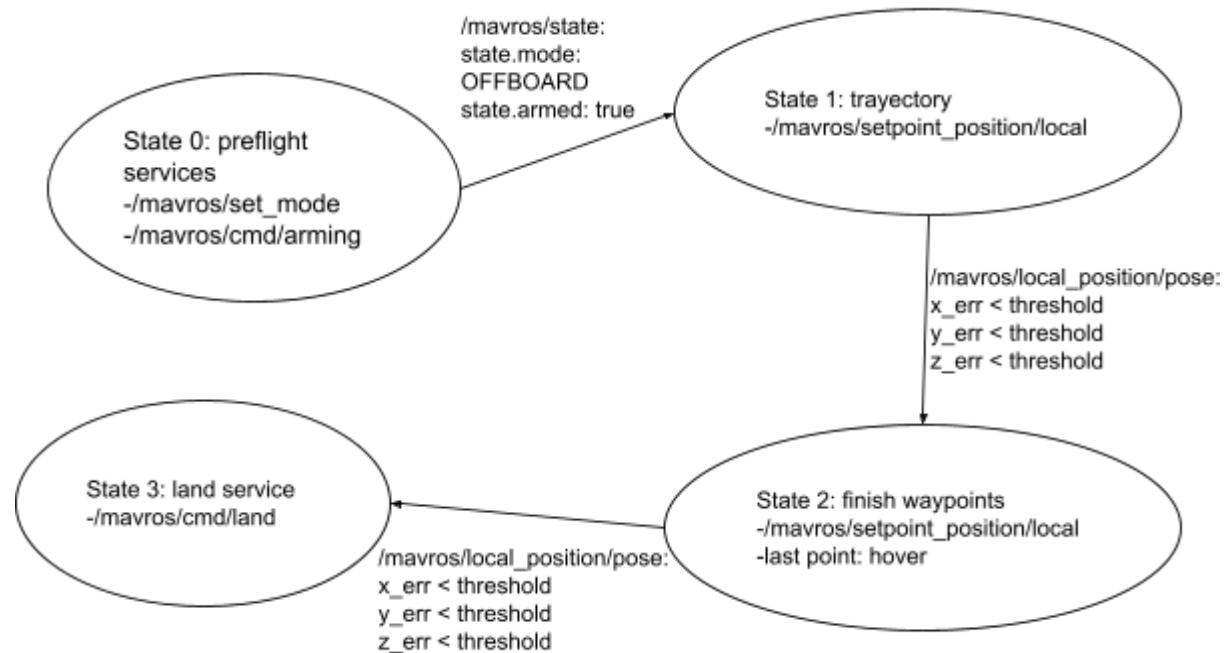


Imagen 6: Máquina de estado código `waypoints.py`

Se logra hacer una trayectoria recibiendo la información de la posición (también *local*) del robot del tópico */mavros/local_position/pose*, para cambiar de *setpoint* al recibir en la pose que el robot se ha acercado lo suficiente al punto en x,y,z.

Al momento de pasar a pruebas de vuelo físicas, destaca que el modo de vuelo no es *GUIDED* para el dron físico, sino *OFFBOARD*. Notamos una alta dependencia en los sensores del robot, principalmente del GPS, pues al momento de correr el hover, refiere su centro no como su origen sino que otro punto lejano a donde lo colocamos, punto que variaba a cada corrida del nodo. Las decisiones con base en este comportamiento fue que optaríamos por usar otro tópico, uno que tome un Twist para publicar, el cual es */mavros/setpoint_velocity/cmd_vel_unstamped*, que no controla directamente las velocidades del robot, sino que actualiza el *setpoint* deseado y sigue utilizando el controlador para llegar al punto. La otra decisión fue que para hover utilizamos el servicio */mavros/cmd/takeoff* que recibe una altura en uno de sus argumentos y se basa en la información del lidar. Para usar el lidar se instaló la paquetería *mavros_extras* y se tuvo que reemplazar el sensor lidar altímetro debido a que el anterior ya estaba muy deteriorado y entregaba mediciones muy imprecisas que el mismo dron reconocía como peligrosas, por ejemplo un hover a 1 metro de altura, el sensor entregaba altura de 15 metros, medida imprecisa e inconsistente ya que a la siguiente medición entregaba 8 metros y valores similares, lo que implica un comportamiento físicamente imposible donde el dron cambia su altura más de 5 metros en menos de medio segundo, por lo que la respuesta del robot es aterrizaje por seguridad. Finalmente, para cualquier prueba de vuelo física, el robot debe estar en modo *OFFBOARD*, armado y antes de mandarlo a hacer cualquier cosa, debe hacer hover mediante el servicio *takeoff*.

Joy - Teleoperación

Se implementó un código que empleando un mando de xbox y la librería de Joy, permite controlar un VANT cuadricóptero incluido en el paquete de Hector. Al levantar el nodo “*hover_drone*” será posible controlar la altura del VANT usando los gatillos inferiores del control de xbox. De la misma forma se puede cambiar la dirección en z (“Yaw”) usando los gatillos superiores así como la velocidad en X y Y con las palancas izquierda y derecha del mismo mando. Tras la realización de este código se logró reconocer apropiadamente la funcionalidad y la forma en la que se puede controlar el VANT mediante ROS pues la simulación que permite que el VANT se mueva por el espacio mediante las publicaciones en el tópico de “*cmd_vel*”, también publica información que cada vehículo aéreo multirotor emplea con regularidad dentro de MAVROS como la lectura del altímetro que mide la distancia del piso, las inclinaciones en los ejes de giro *roll* , *pitch* y *yaw* así como la aceleración lineal y angular en los ejes X, Y y Z (que son medidas que proporciona una IMU).

Right Hand Rule

Se implementó un nodo ROS simulado en Gazebo para que el dron haga la regla de la mano derecha, con este algoritmo se busca que el dron siga la pared que se encuentra a la

derecha, controlando la distancia que hay entre estos dos. Además, se busca que el dron siempre se encuentre en paralelo con la pared y la cámara viendo al frente, con la pared a la derecha. La distancia a la pared se controla con el error multiplicado por una K proporcional y esta se distancia se obtiene utilizando el lidar, específicamente los rayos que se encuentran entre los ángulos de 90° a 125° . De igual manera el ángulo del dron a la pared se controla con el error multiplicado por una proporcional K y se obtiene mediante trigonometría y ley de cosenos. Donde el ángulo que se busca, que llamaremos θ_h se obtiene mediante las siguientes fórmulas:

$$d = r_1 - r_2 * \cos(\alpha)$$

$$c = r_1^2 + r_2^2 - 2 * \cos(\alpha)$$

$$\sin(\theta_h) = \frac{d}{c}$$

Con esto en mente, si la pared se cierra al robot, el ángulo obtenido es positivo y en caso contrario, donde la pared se abre al lado contrario, el ángulo es negativo.

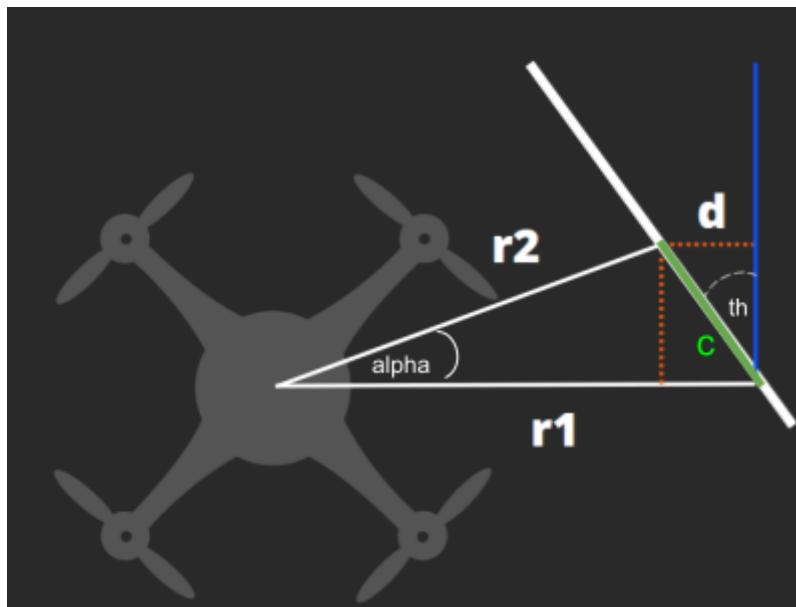


Imagen 7.1: Ángulo del dron a la pared.

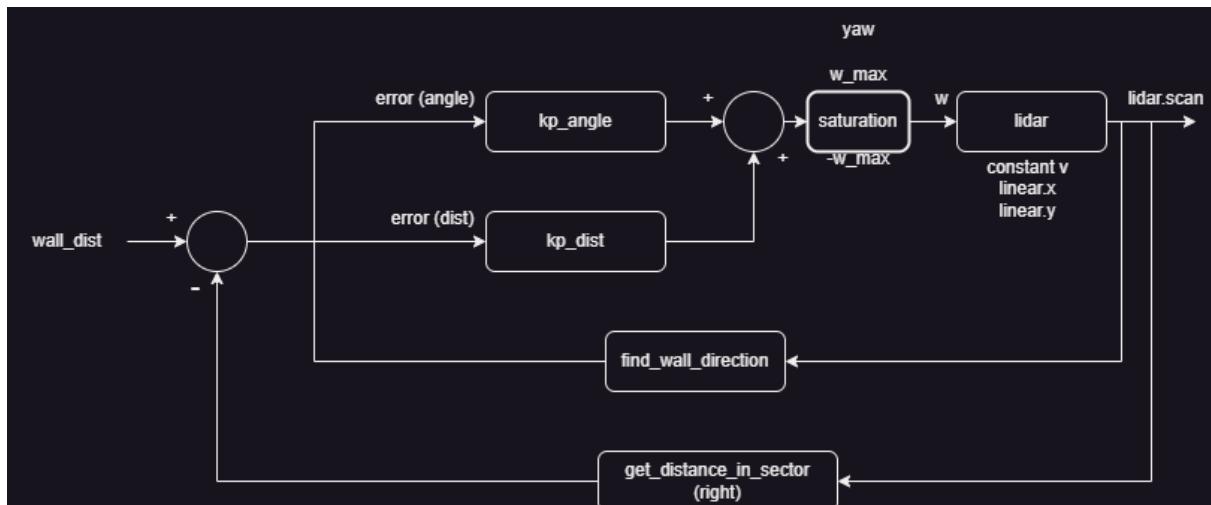


Imagen 7.2: Diagrama del control para seguir la regla de la mano derecha mediante la información del LiDAR

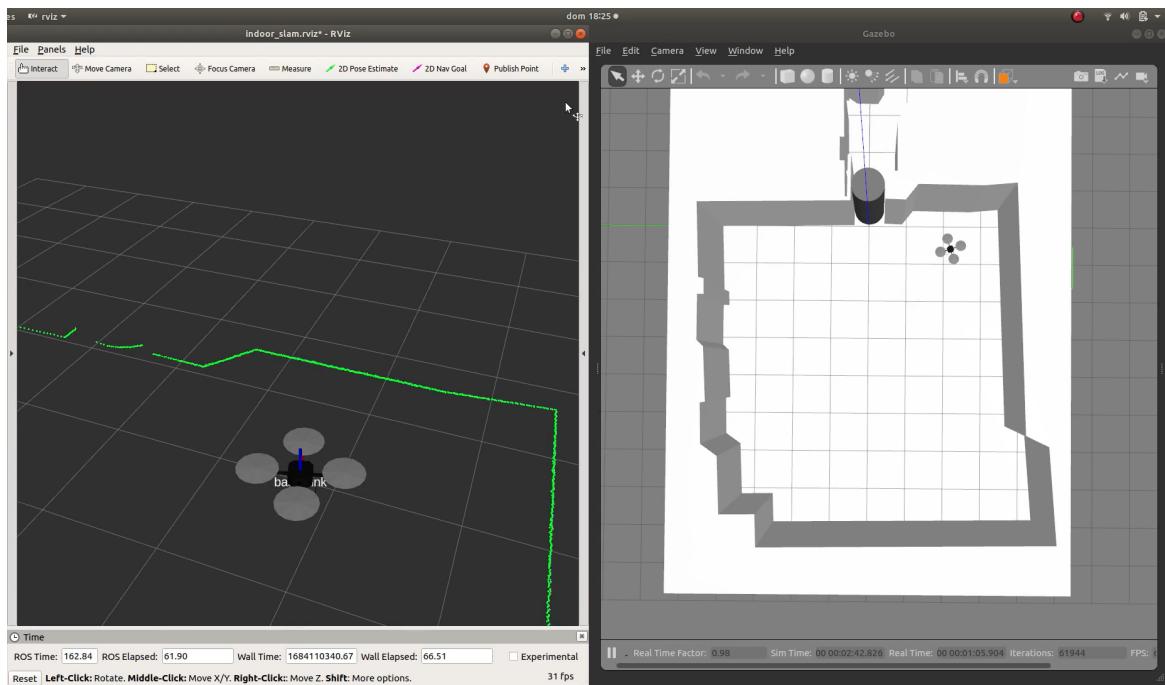


Imagen 7.3: Simulación del VANT de Hector con sensor Lidar integrado siguiendo la regla de la mano derecha.

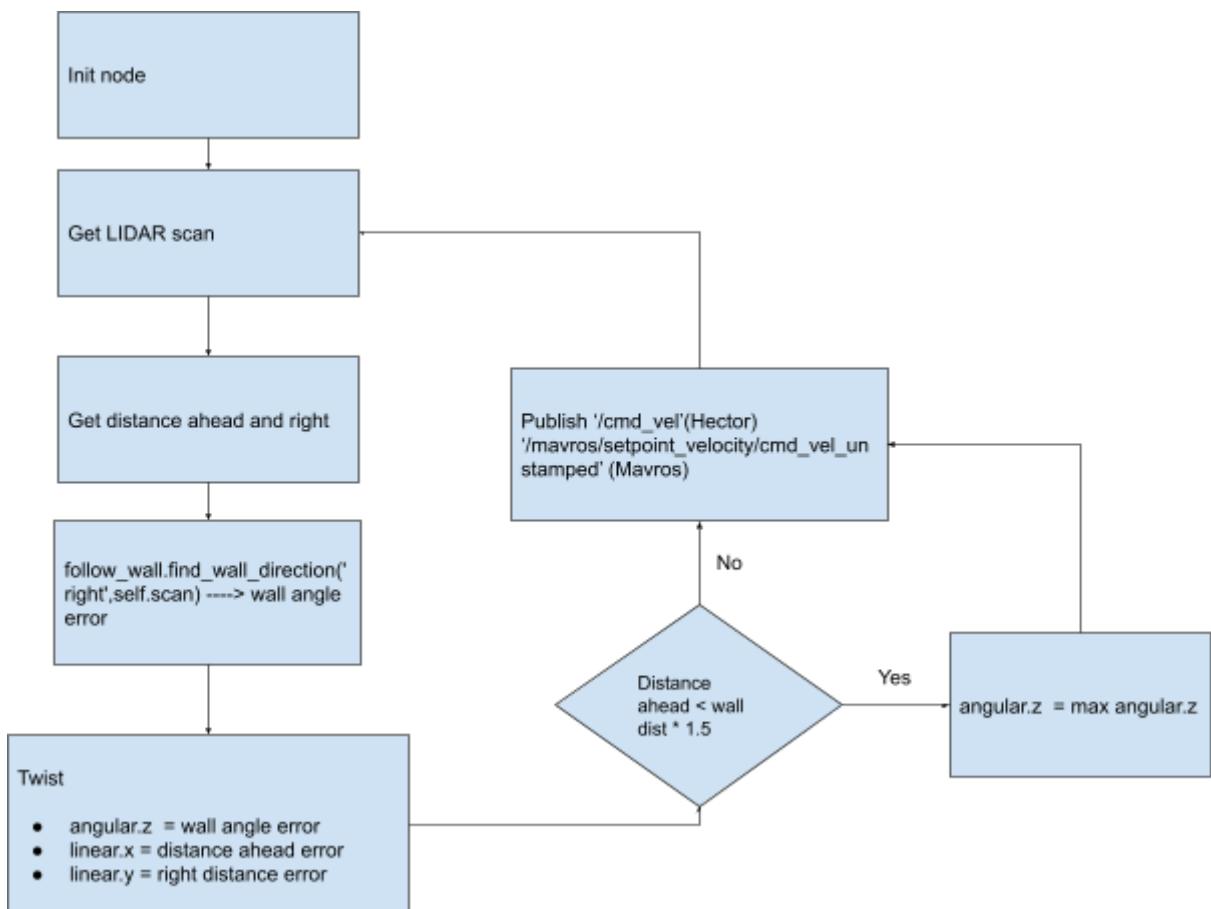


Imagen 7.4: Diagrama de flujo para seguir la regla de la mano derecha mediante la información del LiDAR

Navegación Reactiva con Cámara

Una vez que se tuvieron este par de simulaciones de reconocimiento y familiarización, se creó un mundo en Gazebo donde se incluyesen obstáculos de distintos colores con el objetivo de desarrollar el algoritmo principal que debería seguir el VANT para poder cumplir con el reto. Este mundo cuenta con un circuito sencillo donde sea posible observar el comportamiento deseado del robot el cual debe seguir la siguiente lógica:

- Al detectar un objeto con la cámara verde, el vehículo debe de seguirlo y alinearse lo más posible al centro de este.
- En caso de ver un objeto rojo, el VANT deberá de esquivarlo en la dirección donde haya más espacio (izquierda, derecha por encima)
- Al no ver objetos verdes o rojos y detectar un objeto amarillo con un área de más de 30,000 pixeles (aproximadamente 173x173) este deberá de aterrizar de forma suave.

Se generó un mundo con un total de 3 paredes rojas colocadas de tal forma que se pueda contemplar que el VANT puede esquivar los objetos por una de las direcciones establecidas, 3 paredes verdes sin colisión con el objetivo de que probar el seguimiento de objetos y finalmente una pared amarilla detrás de todos los objetos para que al estar a una distancia considerable de esta, el robot haga un aterrizaje.

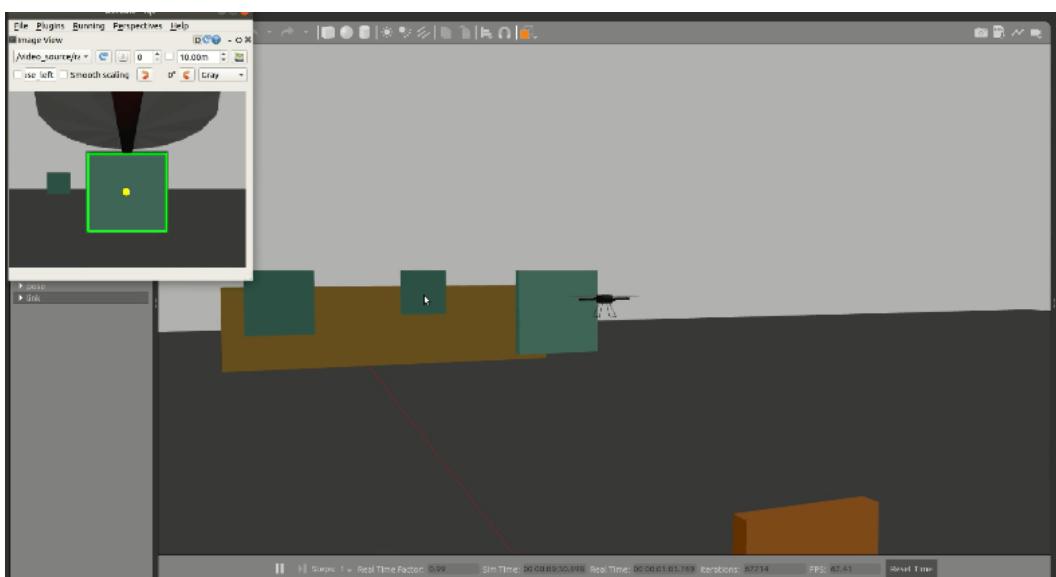


Imagen 8: Captura de pantalla del mundo generado

Se crearon un total de 2 nodos que permiten que el VANT se mueva por el mapa siguiendo la lógica establecida. El nodo “guided_set_point_controller” recibe la información del tópico “mavros/set_point_velocity/cmd_vel_unstamped” que es el tópico donde se debe publicar en mavros para que un vehículo pueda moverse acorde a los comandos mandados en un mensaje tipo Twist desde la computadora. Este mismo nodo permite que el VANT mantenga su altura al no recibir velocidad en Z de forma similar a como lo haría el robot en la vida real.

El nodo “drone_follow_obj” recibe la información proporcionada por la cámara frontal y posteriormente convierte la imagen a HSV y genera 3 máscaras para filtrar cada

uno de los colores. Tras aplicar cada máscara a la imagen original, se realiza una erosión y una dilatación con kernel de 5x5 para filtrar el ruido. Una vez que se tiene la imagen filtrada, se hace uso del algoritmo *find contours* para detectar bordes y ubicar el centroide del área más grande(si bien esto sería difícil con la imagen original, al filtrar por color con las máscaras, la posición del objeto se vuelve muy evidente). Se almacena en un vector de booleanos si existe se detectan objetos de alguno o de todos los colores establecidos (verde, rojo, amarillo).

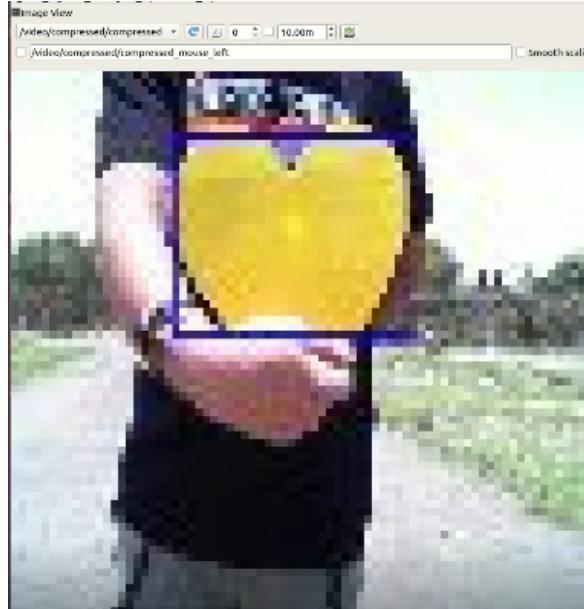


Imagen 9: Captura de pantalla de la detección de objetos

Dependiendo del objeto que se detecte (siguiente la jerarquía: rojo -> verde -> amarillo), el VANT se comportará de una manera específica. Al detectar un objeto rojo se realizará una acción de control proporcional donde se hace uso de la posición del centro del objeto dentro de la imagen como valor de entrada y se emplea una referencia acorde a la dirección donde hay mayor espacio para que el robot evada el objeto.

- La referencia sería $x = 0$ cuando hay más espacio hacia la derecha
- La referencia sería $x = W_{px}$, donde W_{px} es el ancho de la imagen en píxeles, cuando hay más espacio por la izquierda
- La referencia sería $z = (H_{px} * 0.90)$, donde H_{px} Es el alto de la imagen en píxeles, cuando hay más espacio hacia arriba del objeto.

Una vez que se calcula el error en la posición del objeto, este se multiplica por una constante proporcional igual al factor $(n * \frac{W_{px}}{x'_{max} * 0.5})$ para el control en x y $(n * \frac{H_{px}}{z'_{max} * 0.5})$ para el control en z (donde x' y z' son velocidades máximas definidas arbitrariamente para limitar el movimiento del VANT por motivos de seguridad). De esta forma, la velocidad que se publicará en el mensaje Twist irá siempre de 0 a la velocidad máxima permitida dependiendo de si el error es 0 o el máximo largo/ancho de la imagen y su ganancia estará definida por el número n.

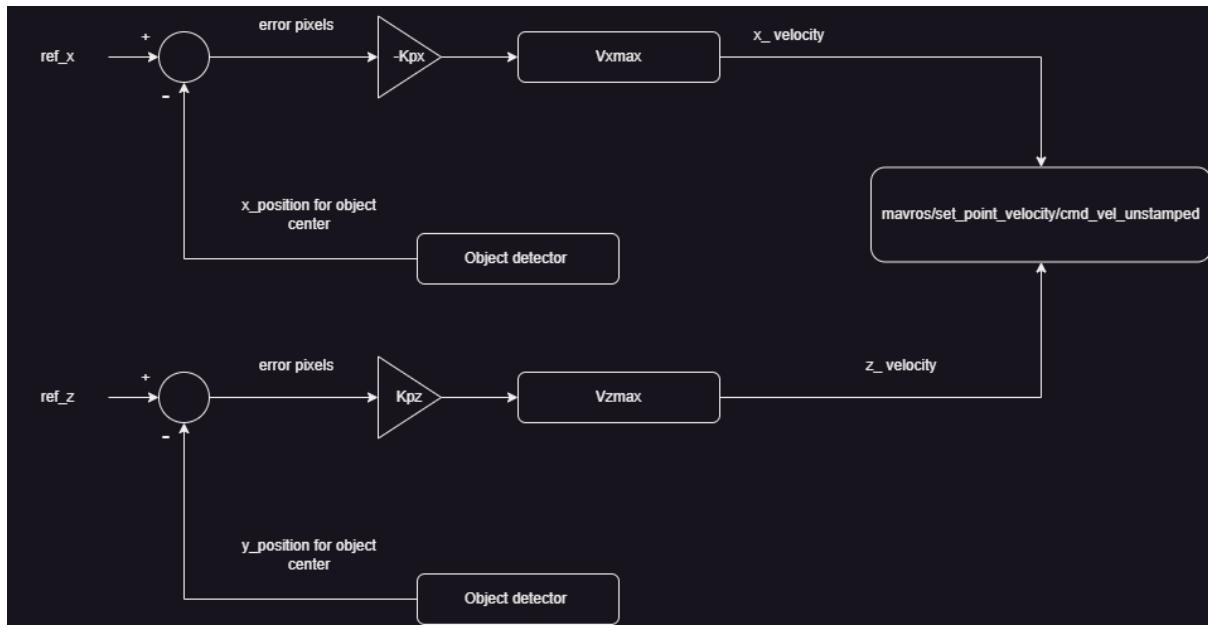


Imagen 10.1: Lazos de control para evasión de objetos.

De forma similar, al detectar un objeto verde, se realiza un control proporcional con un lazo cerrado en x, z. Se toma como entrada la posición del centro del objeto. Sin embargo, en este caso se emplean los lazos de x, z simultáneamente con el fin de que el robot se alinee con el centro del objeto.

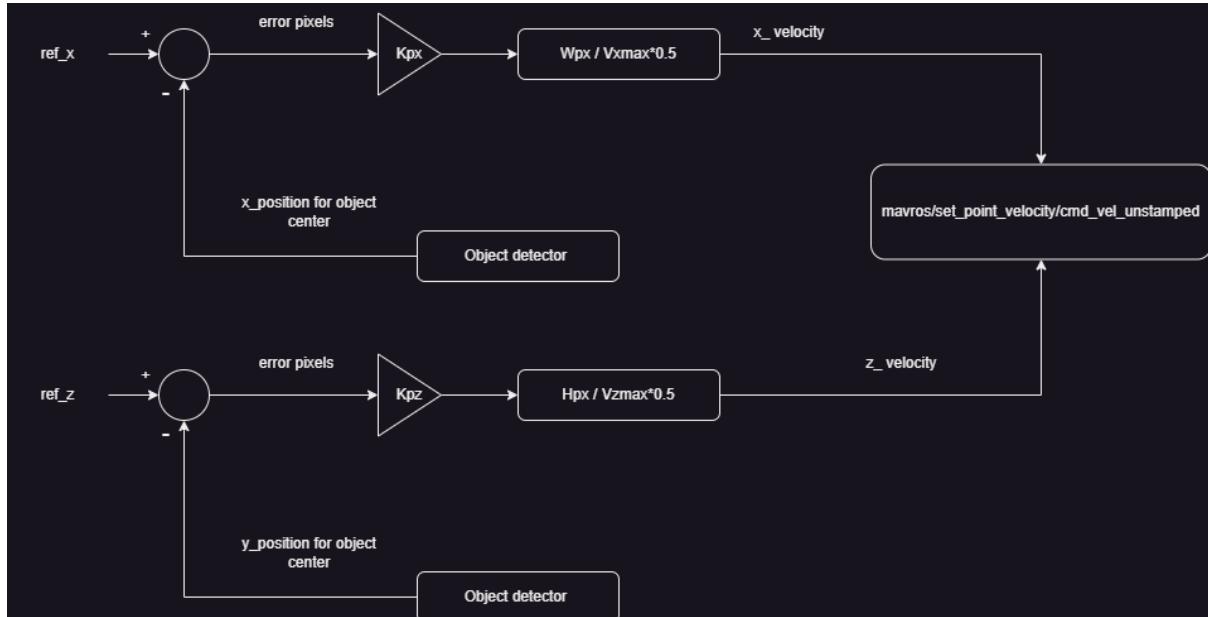


Imagen 10.2 : Lazos de control para seguimiento de objetos.

En caso de no detectar alguno de los objetos mencionados anteriormente, el VANT avanzará a la mitad de su velocidad máxima y mantendrá la posición en x, z. Al detectar un objeto amarillo de más de 30,000 pixeles de área (~173 x 173) el VANT aterrizará concluyendo el proceso definido en su programación.

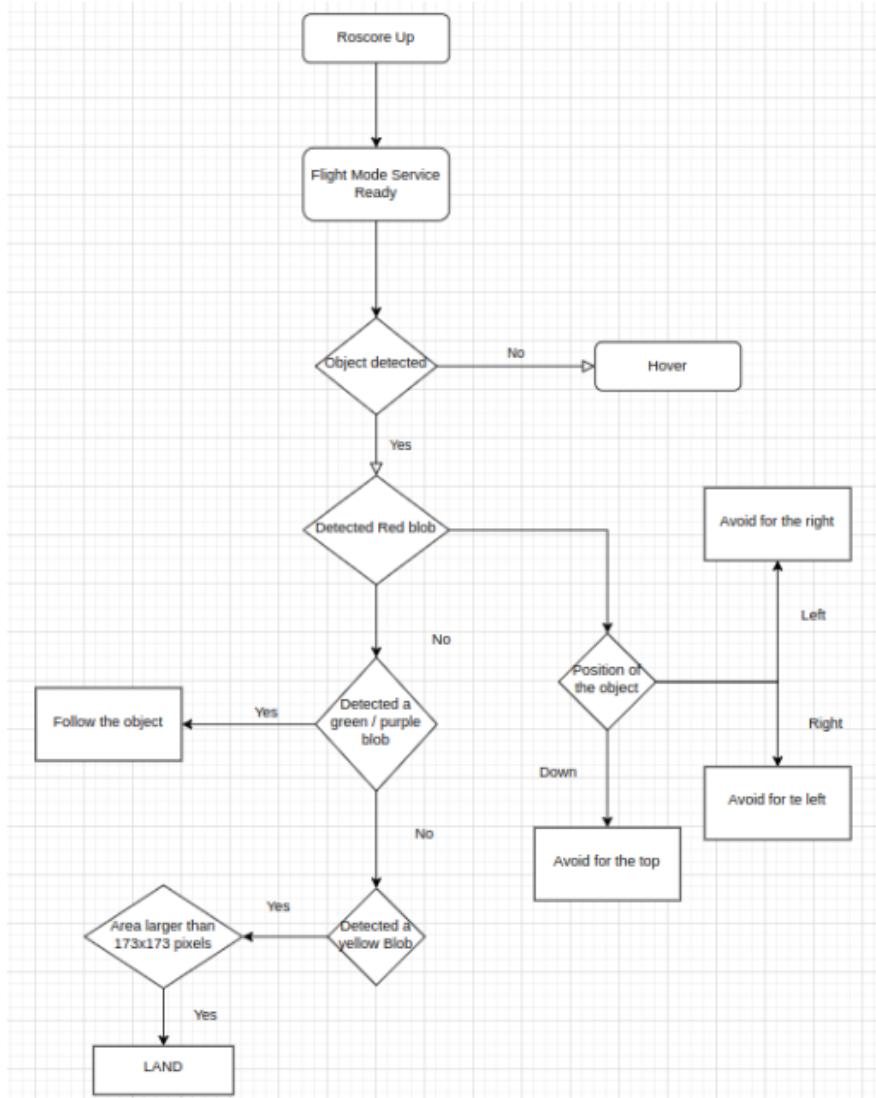


Imagen 10.3: Diagrama de bloques de la lógica del algoritmo.

Para la implementación en físico, se empleó el mismo algoritmo utilizado para la tercera simulación en el paquete de Hector (Imagen 10.3) y se le hicieron algunos ajustes para poder probar la funcionalidad de forma segura sin perder el control del VANT y adaptándose al ambiente real. Primero, se sintonizaron las máscaras para que funcionasen con la iluminación del ambiente de pruebas. Se cambió el color de los objetos de seguimiento por morado ya que este es menos común y esto permite que la visión sea más robusta al no confundirse con el pasto o los árboles. Como segunda modificación eliminó el control de lazo cerrado en z, ya que las condiciones ambientales como el viento le impedían al VANT mantener o modificar la altura de forma lo suficientemente eficiente como para poder obtener métricas. De la misma forma, se observó que el LiDAR era suficientemente preciso para que el vehículo pudiese mantener la altura en condiciones con bajo viento, pero manteniendo un error de 10 a 30 cm, cuál de la misma forma, sería un error considerable como para poder medir el desempeño del VANT al realizar control en la altura. Se definió que el VANT permaneciera en hover al no detectar ningún objeto y solo realice movimientos laterales para esquivar los objetos rojos, avance para seguir los objetos morados y aterrice con el objeto

amarillo. Finalmente, se modificaron los tópicos para que el programa funcionase con MAVROS y se usaron servicios para realizar el armado , despegue y aterrizaje.

Pruebas de funcionalidad

Como pruebas de funcionalidad se decidió correr del código de evasión de obstáculos “follow_objects_v2.py” el algoritmo tanto en simulación como en implementación física. Para la simulación se le indicó al dron que avanza hacia en frente a la mitad de su velocidad máxima y realizar las acciones necesarias para evadir los objetos rojos y seguir los verdes. En la implementación física se conectó una computadora al dron mediante una red LAN usando un router y se le enviaron los comandos necesarios para que éste se arme, despegue y finalmente corriese de forma nativa desde la Odroid el código, así como transmitir a una resolución menor a la nativa las imágenes de la cámara del dron hacia la computadora. Dentro del código se implementó una función que almacena los datos de la posición en x, z de cada objeto así como su referencia en el lazo de control. Una vez que se terminase de correr el código, se guardaron los datos en archivos csv para su posterior análisis.

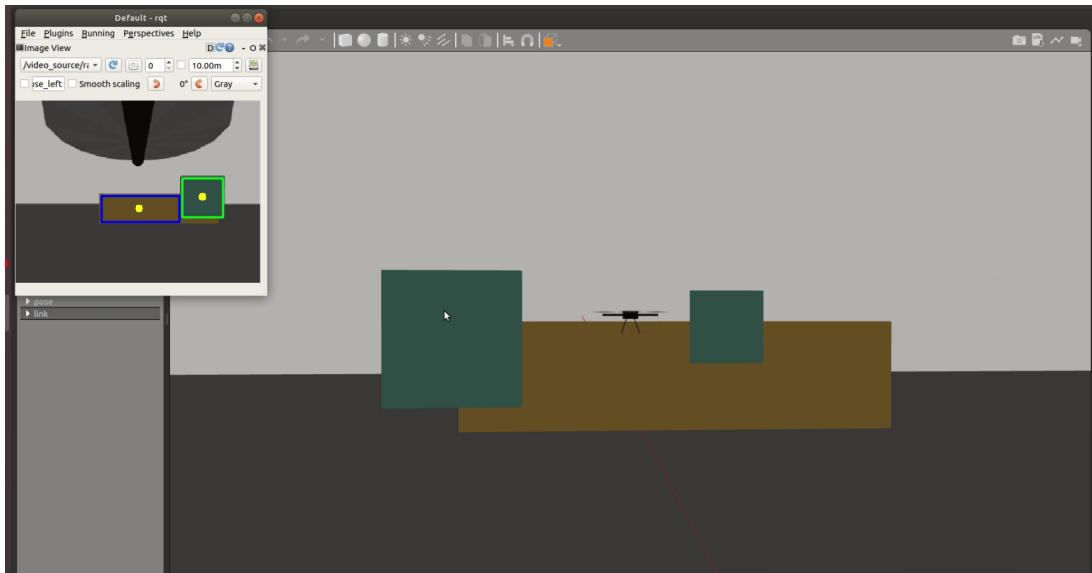


Imagen 11.1: Resultados en simulación



Imagen 11.2: Resultados en implementación física.

Resultados

Para tomar las muestras, se consideró como tiempo inicial, el instante en el que la posición del objeto llega a su pico, ya que este es el momento en el que se detecta un nuevo objeto en los límites de la pantalla. Como tiempo final se tomó el tiempo en el que la posición llega a un error de menos del 10% con respecto a la referencia. La diferencia de estos dos instantes sería el tiempo de establecimiento (ts). Se obtuvo el promedio de cada muestra y además un promedio global de los promedios tanto de la evasión en X y Z, tanto en el seguimiento como en evasión. En la simulación, el promedio general fue de un tiempo de establecimiento de 5.745 segundos. Por su parte, en la implementación real, el tiempo de establecimiento fue de 1.644 segundos. Esto se debe a que en la implementación física se utilizó un controlador más agresivo que en la simulación debido a que a la hora de realizar la implementación en físico, se consideró el viento y se ajustaron las ganancias proporcionales de tal forma que la velocidad fuera suficiente en para poder moverse venciendo la fuerza del viento en cualquier dirección.

Seguidor						Evasor							
eje	muestra	tiempo inicio	tiempo final	ts	Promedio	eje	muestra	tiempo inicio	tiempo final	ts	Promedio		
x	1	550.965	551.164	0.199		x	1	530.254	540.064	9.81			
x	2	562.364	570.164	7.8		x	2	540.064	545.564	5.5			
x	3	570.164	579.564	9.4		z	3	545.864	547.064	1.2			
		5.79966667						5.50333333					

1	Seguidor					Evasor						
2	muestra	eje	tiempo inicial	tiempo final	ts	promedio	muestra	eje	t inicio de señal	ts > 90%	ts	promedio
5	1 x		1.73	5.53	3.8		1 x		23.83	25.93	2.1	2.1
6	2 x		11.64	13.83	2.19		2 x		28.13	29.33	1.2	1.65
7	3 x		14.43	16.93	2.5		3 x		33.54	34.93	1.39	1.56333333
8	4 x		17.43	18.43	1		4 x		39.33	42.44	3.11	1.95
9					2.3725		5 z		31.03	32.03	1	1.76
11	Promedio global					1.64375						
12	muestra	eje	tiempo inicial	tiempo final	ts	promedio						
13	1 z		1.63	3.14	1.51							
14	2 z		3.73	4.43	0.7							
15	3 z		5.23	7.03	1.8							
16	4 z		8.03	9.03	1							
17					1.2525							
18												

Imagen 12.2: Tablas con los datos de las muestras para la implementación real

Para evaluar el comportamiento general, se graficó además el comportamiento de los datos en las gráficas es posible observar como en cuanto la posición llega a un pico, comienza a crecer o a decrecer en dirección hacia la referencia. En la simulación las gráficas presentan crecimientos y decrecimientos de forma más suaves respecto a la simulación, esto se debe a que en el mundo de gazebo, los objetos eran rígidos además de encontrarse en un ambiente con poco ruido por parte del viento y en la implementación real por el contrario, fueron sostenidos por un compañero del equipo y las oscilaciones que se observan en las gráficas del dron real, son ruido producido por parte del pulso del brazo humano, así como el viento que se presentaba en el lugar de las pruebas.



Imagen 12.3: Gráficas del comportamiento del control proporcional para la simulación

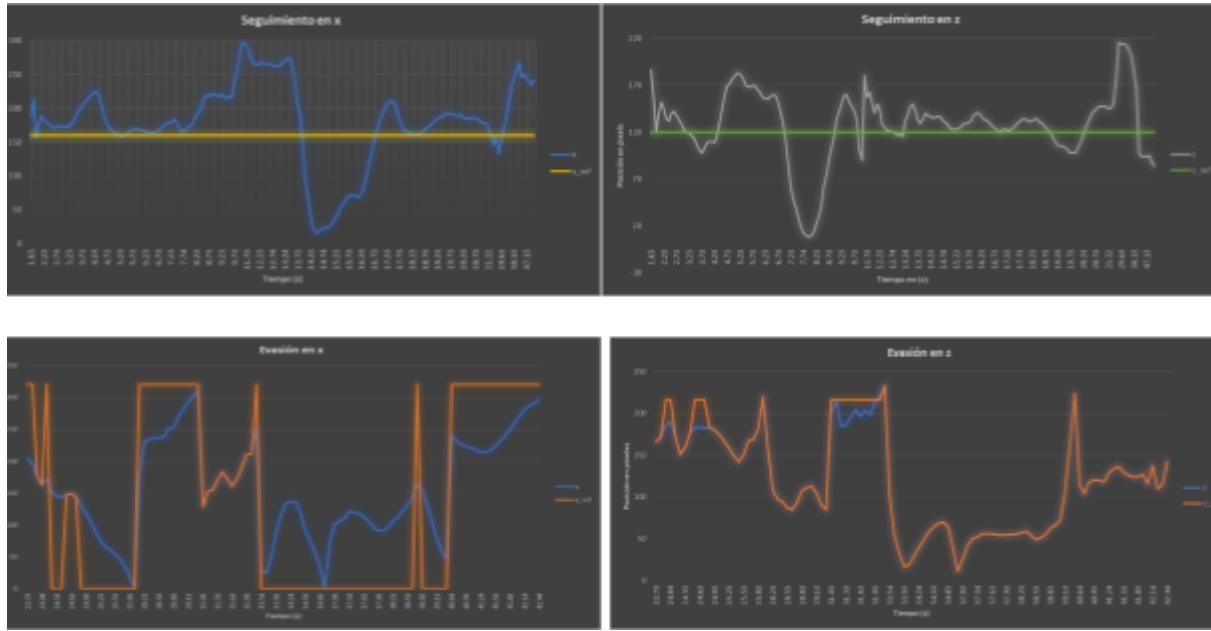


Imagen 12.4: Gráficas del comportamiento del control proporcional para la implementación real

Discusión

Como se pudo observar en los resultados fueron los esperados y el robot completa los objetivos de forma correcta, pues la calibración de colores distingue cada objeto sin detectar otro erróneamente. También el ruido resultante del procesamiento de la imagen es prácticamente despreciable. Comprobamos que el algoritmo de control siempre logra seguir, evadir u aterrizar cada vez que se le pone un objeto del respectivo color. El algoritmo es versátil y fácilmente configurable para la reacción autónoma del dron, pues con la configuración de las velocidades y ganancias se puede modificar el comportamiento y agresividad del dron; se ajustan estos parámetros de acuerdo a la aplicación en concreto. Con este ejercicio también confirmamos que la cámara es un instrumento adecuado para hacer navegación autónoma en el dron.

Sin embargo, durante las pruebas encontramos áreas de oportunidad. La primera son las limitaciones que conlleva utilizar hsv para la detección de colores, ya que se podría llegar a confundir, generando que el dron choque o falle. Hay que considerar que la calibración por HSV es susceptible a cambios de color; nosotros probamos el algoritmo a la misma hora del día, con condiciones meteorológicas muy similares siempre, pensando en que lo ideal es que el dron vuela únicamente cuando el clima lo permita de esa manera, pero por ejemplo, hubo veces que una sombra significativa llega a afectar la detección de color. La otra debilidad es del sensor lidar altímetro, que llega a entregar mediciones imprecisas pero aceptables, como lo es pedirle al robot 1 metro de altura hover, y el lidar indica la altura a 70 cm. Es suficiente esta medición, sin embargo para algoritmos más precisos valdría la pena explorar mayor precisión en el sensor.

Evidencias

Se pueden encontrar las diferentes pruebas simuladas y con el dron real dentro de la carpeta siguiente:

<https://drive.google.com/drive/folders/1--TE2AgF2fUqsA5xBUFMPEHbP6NWGj7m?usp=sharing>

El código desarrollado y documentado se encuentra en el repositorio:
https://github.com/RoboticaInteligente8voTecCEM2023/drone_navigation.git

Manuales

Como parte del reto es la documentación completa para la continuación de desarrollo de diferentes algoritmos a futuro con el dron, se desarrollaron tres materiales particularmente para la comunicación de información importante alrededor de este robot:

- Videoguía para teleoperar mediante control remoto el dron:
https://drive.google.com/file/d/1doE2gkYuS4HlfFpLUNnD73CoDUY_ox-1/view?usp=sharing
- Guía de instalación de simuladores PX4 y Hector:
- [PX4]https://docs.google.com/document/d/18Tcwx6_DLCHzPNpdn7GPGdp5dtcxANIN70IxZyv8_18/edit?usp=sharing
- [Hector]<https://docs.google.com/document/d/1XRpTD6-JkO1hiPoi--qo9vFdtrTaAjhLzxeoqLcS9NI/edit?usp=sharing>

Adicionalmente se incluye el manual que fue proporcionado por parte de la escudería de robótica del Tecnológico de Monterrey Campus Estado de México como material para el desarrollo del reto.

- Manual para configurar el controlador de vuelo del dron mediante Ardupilot:
- <https://docs.google.com/document/d/1MTGtoz5Yv9TsQaBM4K406g1otWGX-f1O/edit?usp=sharing&ouid=105777465025548288351&rtpof=true&sd=true>

Conclusión

Este proyecto fue conformado de varias partes: la instalación de software, chequeo de hardware y exploración de las posibles capacidades como teleoperación y navegación reactiva. Estos tres pasos permiten ver que el equipo de trabajo fue capaz de adaptar y aplicar todo lo visto en la carrera para programar diferentes tipos de robots para que puedan realizar diferentes tareas. Con esto en mente, los resultados fueron los esperados y concluimos que el equipo está preparado para llevar estos conocimientos a la industria de la robótica. El producto final presentó un comportamiento satisfactorio y las herramientas tanto de simulación como implementación física, junto con la documentación pertinente, que se

generaron en este proyecto y probaron ser suficientes para una integración exitosa que permitirá el escalamiento modular para futuros miembros de la escudería de robótica, así como alumnos de la carrera.

Posteriormente, lo ideal sería incluir un LIDAR para que el dron pueda tener navegación reactiva. Con el sensor LiDAR, sería posible la implementación de algoritmos como Bug0, Bug2 o algún algoritmo nuevo y finalmente podría hacerse el ejercicio de incluir el filtro de Kalman desde la información proporcionada por algún sensor, como el IMU, para obtener una localización precisa y de esta forma realizar un SLAM completo. Y valdría mucho la pena explorar algoritmos por Sistema Global de Navegación por Satélite, explotando los recursos de GPS del dron, que ya está integrado y funcional.

Se ha dejado la configuración y documentación del dron para futuras implementaciones de algoritmos de control con base en el hardware y software disponible.

Referencias

- [1] Cultura tecnológica: ¿Qué es un dron y cuáles son sus usos? (s. f.). GCFGlobal.org. <https://edu.gcfglobal.org/es/cultura-tecnologica/que-es-un-dron-y-cuales-son-sus-usos/1/>
- [2] VANT – Ala Fija y Multirrotor – Innovación – Facultad de Ciencias Agropecuarias – UNC. (s. f.). <https://innovacion.agro.unc.edu.ar/vant-ala-fija-y-multirrotor/>
- [3] Delgado V. (2019). Historia de los drones. El Drone. <http://eldrone.es/historia-de-los-drones/>
- [4] Juguetecnic. (2017). ¿Qué es un drone? Tipos, nombres y componentes. Retrieved from <https://juguetecnic.com/blog/que-es-un-drone-tipos-nombres-componentes.html>
- [5] ROS.org. (n.d.). mavros - ROS Wiki. Retrieved: 07/06/2023, from <http://wiki.ros.org/mavros>
- [6] Dr. Alejandro Aceves. Implementación de robótica inteligente (s.f.). Módulo 4 - Vehículos aéreos no tripulados. Tecnológico de Monterrey. Campus Estado de México. 2023. https://github.com/aaceves/hector_quadrotor
- [7] Dr. Alf Kjartan Halvorsen. Integración de robótica inteligente (s.f.). Actividad M3.1: Sensores LIDAR. Tecnológico de Monterrey. Campus Estado de México. 2023.

https://experiencia21.tec.mx/courses/363736/assignments/11473840?module_item_id=21623

999

[8] ROS with Gazebo Classic Simulation | PX4 User Guide. (s. f.). Drones APP & APPIs /

ROS 1 with MAVROS. https://docs.px4.io/main/en/simulation/ros_interface.html

[9] The Cube User Manual V1.0 - CubePilot. (s. f.). Recuperado 23 de septiembre de 2022,

de <https://docs.cubepilot.org/user-guides/autopilot/the-cube-user-manual>

[10] Hex Cube Black Flight Controller | PX4 User Guide. (s. f.). Recuperado 23 de

septiembre de 2022, de https://docs.px4.io/v1.12/en/flight_controller/pixhawk-2.html

[11] The Cube User Manual V1.0 - CubePilot. (s. f.). Recuperado 1 de mayo de 2023, de

<https://docs.cubepilot.org/user-guides/autopilot/the-cube-user-manual>

[12] Dr. Alejandro Aceves López, Bruno Sánchez García, Carlos Antonio Buendía López.

Programación y configuración de drone 2022 (23 de julio de 2022), Recuperado el 1 de mayo

de 2023.

<https://docs.google.com/document/d/1MTGtoz5Yv9TsQaBM4K406g1otWGX-f1O/edit?usp=sharing&ouid=105777465025548288351&rtpof=true&sd=true>