Plataforma hardware y software para el desarrollo y validación de algoritmos de control de actitud para cuadricópteros

Miguel Fernández Cortizas

Índice general

Resumen

hola

Introducción

Un multirrotor es un vehículo aeŕeo no tripulado o UAV (por sus siglas en inglés *Unmanned Aerial Vehicles*) cuyos motores y hélices están orientadas de forma vertical.

Estas aeronaves son mucho más maniobrables que una aeronave de ala fija, ya que, al poder mantenerse estables en una posición fija pueden realizar trayectorias de forma más precisa en espacios reducidos. Comparado con un helicóptero, el cual también puede mantenerse estable en un punto fijo, los multirrotores cuentan con un mantenimiento mucho más sencillo debido a la ausencia de complejos mecanismos.

Generalmente los motores de estas aeronaves son eléctricos, ya que poseen una gran velocidad de reacción y son capaces de manejar una gran cantidad de energía con rendimientos muy elevados. La energía que requieren se proporciona a través de baterías, lo cual limita el tiempo de vuelo de estas aeronaves, cuya autonomía es significativamente inferior a la de las aeronaves de ala fija o a la de los helicópteros con motores de combustión.

Todo esto, unido a su reducido precio, ha permitido que se potencie el uso masivo de este tipo de aeronaves para labores de: agricultura de precisión, topografía, inspección industrial y rodajes cinematográficos, entre otros usos.

Existen diversas topologías asociados a los multirrotores, dependiendo del número de motores y la disposición de éstos. En este trabajo vamos a trabajar con un cuadricóptero o cuadrirrotor (cuenta con cuatro motores) con disposición en X (fig. 1.1).

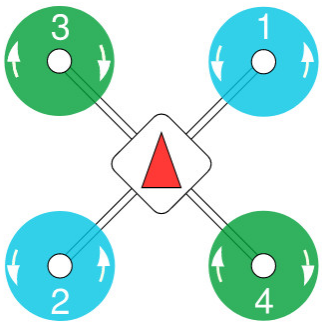


Figura 1.1: Esquema cuadrirrotor en X

Se ha escogido esta topología de multirrotor debido a que posee el mínimo numero de motores que permiten un modelo sencillo de la dinámica de la aeronave.

1.1. Motivación

La popularidad de estas aeronaves en ámbitos como la inspección, la topografía o el ámbito cinematográfico requieren que la nave sea muy estable y que se pueda manejar con precisión. Todas estas aeronaves cuentan con un sistema electrónico que se encarga de

que la nave sea estable y que facilite el pilotaje de la misma. A este sistema se le conoce como la controladora de vuelo o autopiloto.

Figura 1.2: Autopiloto comercial Pixhawk 4

Actualmente, los autopilotos emplean algoritmos de control clásicos, basados la gran mayoría en reguladores PID para asegurar la estabilidad de la aeronave y permitir que se pilote de forma precisa. Debido a la peligrosidad de los drones (hélices afiladas girando a miles de revoluciones por minuto) no se pueden probar nuevos algoritmos de control sin tener sujeto al cuadricóptero.

Las teoría clásica de control empleada para estabilizar un cuadrirrotor requiere de un fino ajuste de los parámetros del modelo. Los últimos avances en aprendizaje automático han permitido que se desarrollen nuevos algoritmos de control empleando técnicas de aprendizaje por refuerzo y redes neuronales.

1.2. Solución propuesta

Con el objetivo de poder desarrollar nuevos algoritmos para el control de cuadrirrotores se ha desarrollado una plataforma, tanto hardware como software, que permita diseñar y probar estos algoritmos de forma segura. Esta plataforma esta constituida de :

- **Entorno de simulación**, sobre el que se puedan probar los algoritmos de control en una aeronave simulada.
- **Cuadrirrotor con autopiloto de diseño propio**, será la aeronave donde se probarán los algoritmos diseñados. Al contar con una controladora de vuelo de diseño propio se pueden implementar los distintos algoritmos de control a bordo.
- **Interfaz con el autopiloto**, el cual comunica a la aeronave con un ordenador, el cual puede enviar comandos a la aeronave y recibir el estado de la misma. Esto permite la implementación de algoritmos *en tierra*, en los cuales la computación del algoritmo se realiza en un ordendador en vez de en el microcontrolador del autopiloto.
- **Banco de pruebas**, con distintas configuraciones en función del experimento que se desee realizar. Este banco permitirá probar los algoritmos de control de forma segura.

Adicionalmente, se ha probado el rendimiento de distintos algoritmos de control basados en técnicas de aprendizaje por refuerzo para poder compararlos con los algoritmos de control clásicos PID.

1.3. Objetivos

Para poder llevar a cabo esta plataforma y poder implementar algún algoritmo basado en aprendizaje por refuerzo es necesario desgranar los objetivos principales en tareas de alcance más reducido:

- **Entorno de simulación:**
  - Estudio del estado del arte acerca de entornos de simulación para el diseño de algoritmos de control para cuadricópteros.
  - Estudio del estado del arte acerca de las técnicas de aprendizaje por refuerzo y el control de drones.
  - Adaptación del entorno de simulación a la plataforma escogida e integración con ROS.
  - Implementación de algoritmos clásicos en el entorno de simulación.
  - Diseño de nuevos algoritmos de control basados en aprendizaje por refuerzo y entrenamiento de los mismos.
  - Comparativa de los algoritmos en simulación.
- **Cuadrirrotor con autopiloto de diseño propio:**
  - Estudio del estado del arte acerca de los autopilotos comerciales.
  - Diseño y construcción completa del autopiloto: componentes, placa de circuito impreso y soldadura de componentes.
  - Programación del *firmware* del autopiloto.
  - Diseño CAD de los componentes mecánicos del cuadrirrotor e impresion 3D de los mismos.
  - Montaje y esamblaje de todos los componentes de la aeronave.
- **Interfaz con el autopiloto:**
  - Diseño del protocolo de comunicación WiFi
  - Integración del protocolo con ROS y con el autopiloto.
- **Banco de pruebas**:
  - Diseño CAD de las piezas de los distintos bancos de pruebas e impresión 3D de las mismas.
- **Experimentación de los algoritmos en el mundo real:** reducir o ampliar con los experimentos disponibles
  - Implementación a bordo de los algoritmos clásicos.
  - Implementación externa de los algoritmos clásicos.
  - Implementacion externa de los algoritmos basados en técnicas de aprendizaje por refuerzo.

Estado del arte

2.1. Plataformas de control de cuadricópteros

2.2. Aprendizaje por refuerzo

Las teoría clásica de control empleada para estabilizar un cuadricóptero requiere un fino ajuste de los parámetros del modelo. Los últimos avances en aprendizaje automático han permitido que se desarrollen nuevos algoritmos de control empleando técnicas de aprendizaje por refuerzo y redes neuronales.

En 2004, HJ Kim et al. [1] emplearon algoritmos de *reinforcement learing* para estabilizar (*hoover*) un helicóptero y conseguir realizar maniobras acrobáticas.En 2006 Andrew Y. et al [2] siguieron con esta investigación consiguiendo que el helicóptero se estabilizara al revés (*inverted hoover*). En 2010 Travis Dierks et al. [3] desarrollaron un controlador no lineal, basado en redes neuronales, para estabilizar un cuadricóptero y seguir trayectorias.

Unos años después, en 2017 Jemin Hwangbo et al. [4] desarrollaron un método para controlar un quadricóptero con una red neuronal usando técnicas de *reinforcement learning*. En 2018 William Koch et al. [5] desarrollaron un entorno de simulación, GYMFC, para el desarrollo de sistemas de control empleando RL.

Fundamentos teóricos

3.1. Control clásico

Una vez conseguida una estimación del estado en el instante t , S_t se necesita emplear un algoritmo de control que genere las acciones A_t correspondientes para llegar al estado $S_{t'}$ deseado de forma óptima.

BUCLE DE CONTROL

Existe una gran cantidad de controladores clásicos que se pueden emplear para generar estos comandos, por ejemplo: PID o LQR. En este trabajo se han explorado 2 controladores: el primero de ellos se trata de un controlador clásico PID y el segundo consiste en un controlador no lineal modelizado por una red neuronal.

3.1.1. Controlador PID

El controlador PID es un método de control clásico basado en generar una salida A_t basada en el error e_t, es decir, la diferencia entre el estado deseado S_{ref} y el estado actual S_t.

$$e_t = S_{ref} - S_t \tag{3.1}$$

La salida del controlador PID se contruye sumando la contribución de 3 términos:

1. **Parte proporcional (P)** La parte proporcional, como su propio nombre indica, crece de forma proporcional con el error e_t, según la constante de proporcionalidad K_p

$$P_{out} = K_p \cdot e_t \tag{3.2}$$

2. **Parte integral (I)** La parte integral

3.2. Redes neuronales

3.3. Aprendizaje por refuerzo

El aprendizaje por refuerzo o *Reinforcement learning* [6] es un área del aprendizaje automático o *Machine Learning* en el que un agente interactúa con un entorno buscando la mejor acción a realizar en función de su estado actual.

Se diferencia de otras técnicas de aprendizaje automático es su enfoque orientado a la interacción directa con el entorno, sin basarse en un modelo completo del entorno o en un conjunto de ejemplos supervisados.

El aprendizaje por refuerzo emplea el marco formal de los procesos de decisión de Markov (*MDP*) en los cuales para definir la interacción entre en agente y el entorno en términos de estados, acciones y recompensas.

Un proceso de decisión de Markov (*MDP*)

Estos procesos de decisión incluyen causalidad, la existencia de recompensas explícitas a sense of uncertanty and nondeterminism

Además del agente y el entorno se pueden identificar cuatro elementos principales más en un sistema de aprendizaje con refuerzo:

- **Política** (Proveniente del termino anglosajón *policy*, el cual es el término empleado en el estado del arte). Define el conjunto de acciones que debe realizar el agente para conseguir maximizar su recompensa en función su estado, el cuál es percibido a través del entorno. La *policy* constituye el núcleo del agente y nos permite determinar su comportamiento. Estas políticas pueden ser estocásticas.

- **Recompensa**. Define el objetivo del agente en un problema de aprendizaje por refuerzo. En cada salto de tiempo (*step*) el agente recibe una recompensa por parte del entorno (un número).

  El objetivo del agente es maximizar su recompensa a largo plazo.

- **Función de valor**. Define el comportamiento que va

- **Policy**. Define el comportamiento que va

3.3.1. DQN

3.3.2. DDPG

3.3.3. TRPO

3.3.4. PPO

Hardware

Un cuadricóptero o cuadrirrotor es una aeronave no tripulada (UAV) que está propulsada por cuatro motores cuyas hélices están orientadas verticalmente.Se ha diseñado y construido un cuadrirrotor casero para poder probar en la realidad el control del dron.

Un cuadricóptero convencional cuenta con: un chasis o *frame* que lo sustenta, cuatro motores y la electrónica necesaria para controlarlos, una controladora de vuelo que lo comanda y baterías que le proporcionan energía.

A continuación se detallará como son las distintas partes del dron que se ha fabricado.

4.1. Cuadro

El *frame* está compuesto por perfiles de aluminio y piezas de PLA fabricadas mediante impresión 3D de diseño propio. La estructura básica está formada por 5 conjuntos de piezas distintos:

- **Portamotores**: son las piezas donde se alojan los motores. Para conseguir una buena fijacion los motores se atornillan a los portamotores empleando 4 tornillos dispuestos seǵun los vértices de un rombo.

  Imagen

- **Brazos**: se encargan de unir los portamotores con el *núcleo* de la estructura. En este diseño, los brazos consisten en perfiles de aluminio de sección cuadrada de 8mm de lado.

- **Núcleo**: Es la pieza principal del diseño, en la que se anclan el resto de las partes y donde se alojan los componentes electrónicos. Esta pieza sustenta los brazos y el porta-baterias de la aeronave. Cuenta con agujeros a medida para poder situar el autopiloto y los variadores.

  Imagen

- **Separadores**: su propósito es mantener unidos el *núcleo* y el porta-baterías manteniendo una separación fija entre ellas.

  Imagen

- **Porta-baterías**: Es la parte inferior del cuadricóptero. En ella se apoya la batería Li-Po que alimenta al dron y se mantiene anclada durante el vuelo. Además posee unas protuberancias cuya función se asemejaría a las de un tren de aterrizaje.

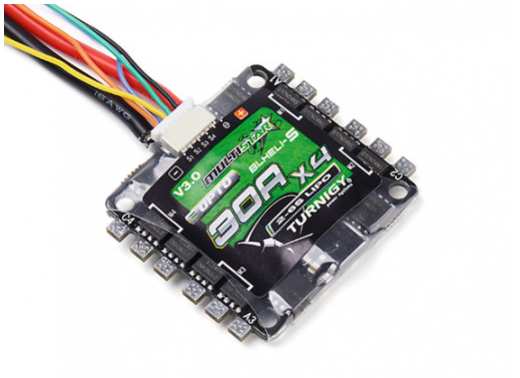
  Imagen

4.2. Motores y variadores(ESC)

El dron cuenta con 4 motores sin escobillas (*brushless*) LHI MT2204 II de 2300KV con una tensión de alimentación entre 7.2 V y 11.1 V (2s -3s en una batería LiPo) y una corriente continua máxima de 16A.

Estos motores son trifásicos, es decir, se alimentan con 3 corrientes alternas monofásicas de igual frecuencia y amplitud, desfasadas 120° eléctricos. Para obtener estás formas de ondas a partir de la corriente continua de las baterias, se utilizan los variadores o *ESC*.

(a) Motores LHI MT2204 II empleados



(b) ESC Multistar Race 4 in 1 30A BLHeli empleado

Un variador o *ESC (Electronic Speed Control)* es un circuito electrónico que controla y regula la velocidad de un motor eléctrico.

https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/

Profundicar en las ondas generada por el ESC

4.3. Baterias

Para alimentar al dron, se han elegido baterias tipo LiPo por su alta tasa de descarga (la batería que se ha escogido es capaz de entregar hasta 130 A) y la su estabilidad en la tensión mientras están cargadas.

4.4. Autopiloto

En los drones, el sistema que se encarga de estabilizar al cuadricóptero y hacerlo pilotable se denomina la controladora de vuelo o el Autopiloto. Existe una gran variedad de controladoras en el mercado, pero para este trabajo se ha diseñado una controladora propia con el fin de poder tener acceso a todos los sensores y a implementar el algoritmo de control de forma óptima. El autopiloto consta de 3 partes diferenciadas: la electrónica de potencia, el microcontrolador y los sensores. A continuación se tratará sobre estas partes con más detalle.

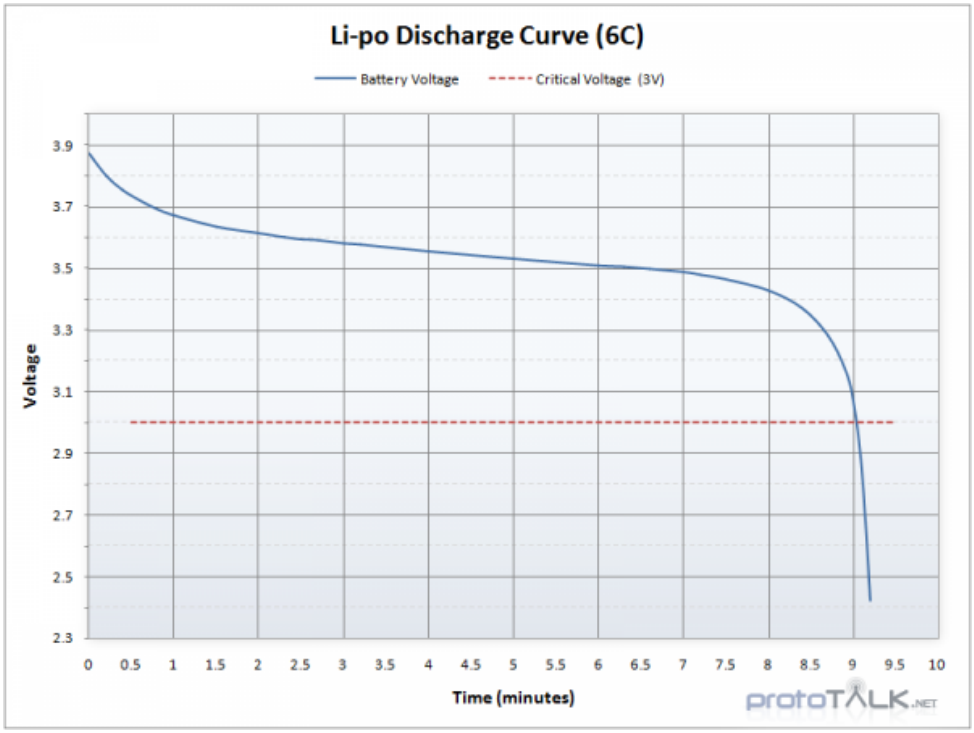


Figura 4.2: Curva típica de descarga de una bateria LiPo (fuente: ProtoTalk.net)

Estaría bien un par de imágenes de la PCB (anverso y reverso)

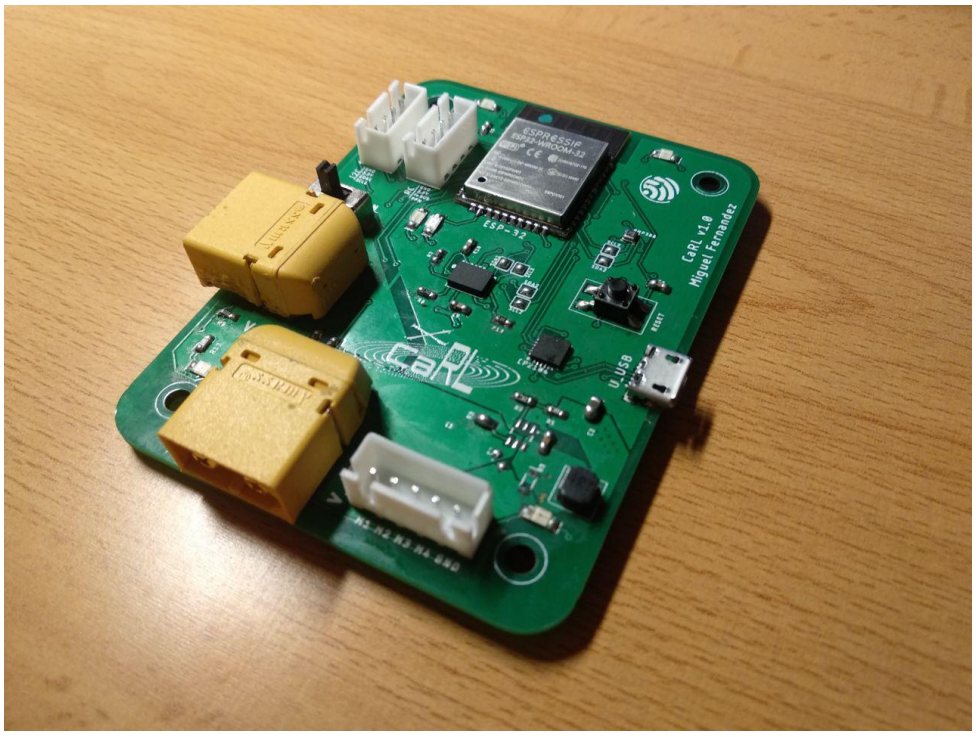


Figura 4.3: Autopiloto CaRL (*Cuadcopter with autopilot based on Reinforcement Learning*).

4.4.1. Fase de Potencia

Con el fin de poder gestionar la potencia entregada por las baterias a la placa y a los motores se ha diseñado una etapa de potencia en la que se debe mencionar dos partes: el interruptor de potencia y el regulador a 3.3 Voltios.

Interruptor de potencia

Los motores del dron pueden llegar a consumir 12 Amperios cada uno, lo que los cuatro motores pueden llegar a consumir 48 Amperios. Un interruptor con tamaño reducido no puede manejar tanta corriente, por ello se ha empleado un transistor MOSFET de canal P por el que pueden circular hasta 100 Amperios, con el fin de abrir o cerrar el paso de corriente desde las baterías al resto de la placa. El MOSFET se controla con un interruptor de poca potencia entre drenador y puerta.

Cuando se cierra el interruptor se alimenta directamente al ESC y al regulador de tensión.

Regulador a 3.3V

La eléctronica digital de la PCB se alimenta y emplea lógica a 3.3 Voltios, por lo que no la podemos conectar a las baterías de 11.1 Voltios. Para adecuar la tensión se ha escogido un regulador Step-down de tipo Buck (Figura 4.4) (¿explico como funciona un convertidor Buck?). El circuito integrado que se encarga de conmutar la fuente es el chip AP3211.

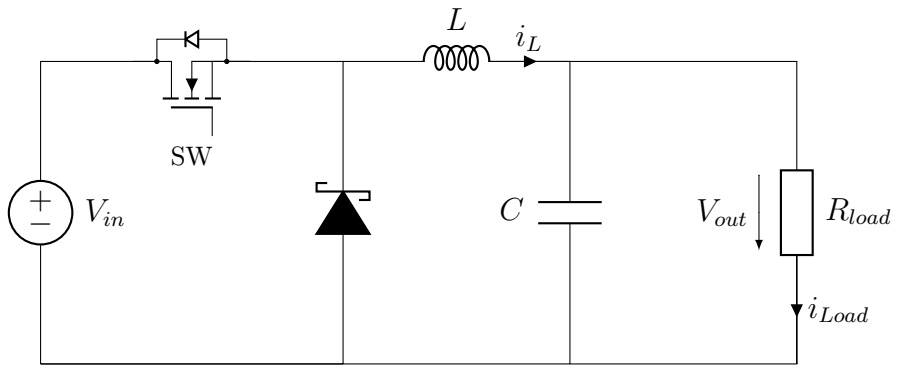


Figura 4.4: Esquema de un convertidor Buck

4.4.2. El microcontrolador (ESP32)

El microcontrolador por el que se ha optado para este Autopiloto es el ESP32, un microcontrolador de doble núcleo con dos CPUs XTensaL6 con arquitectura Harvard [7]. El ESP32 tiene una frecuencia de reloj de hasta 240MHz ,y cuenta con una antena WiFi a 2,4 GHz y conexión Bluetooth 4.2 BLE [8]. Los motivos por los que se ha decidido emplear este microcontrolador son:

- Elevada frecuencia de procesamiento y dos nucleos de procesamiento.
- Antena WiFi incorporada.

- Bajo consumo de potencia.

Para poder programar el microcontrolador se utiliza un convertidor USB (Bus Serie Universal) a UART (Transmisor-Receptor Asíncrono Universal) que permite conectar por USB el microcontrolador para poder programarlo y hacer depuración utilizando comunicaciones Serial. El chip que realiza esta funcion es el CP2104.

4.4.3. Sensores

La principal fuente de información procedente del exterior que recibe una controladora de vuelo se la proporcionan las unidades de medición inercial (IMU). Las IMUs son dispositivos electrónicos que son capaces de medir aceleraciones, velocidades y detectar la orientación de un sistema. El principal problema de estos sensores es que sufren error acumulativo a la hora de estimar posición y velocidad. Para corregir este error acumulativo en los drones, se suelen fusionar estas medidas con otras provenientes de mediciones absolutas tales como GPS o Láser, aunque en este autopiloto unicamente emplearemos las medidas de las IMUs.

Otros sensores utilizados frecuentemente en los autopilotos son brújulas (se encuentran integrados en la IMU para corregir errores de orientación) y barómetros (para estimar la altitud a la que se encuentra el dron).

Nuestro autopiloto cuenta con dos IMUs de 9 Grados de Libertad y un barómetro para conseguir una mejor estimación del estado del cuadricóptero:

1. **BNO 055 (BOSCH)**: El circuito integrado de Bosch es un sensor "inteligente" que incluye los sensores y la fusión de las lecturas de los distintos sensores en un único componente. Este encapsulado cuenta con: un acelerómetro, un giróscopo y un magnetómetro triaxial. Además integra un microcontrolador de 32 bits en el que se ejecuta el algoritmo de fusión integrado. El sensor se encuentra en un encapsulado LGA de 28 pines con una huella (*footprint*) de $3,8$ x 5,2 mm^2.

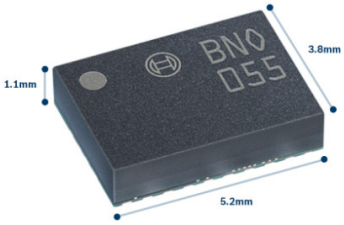
   Este sensor nos proporciona estimaciones del estado completo de la aeronave con una frecuencia de refresco de 100Hz.

2. **MPU 9250 (TDK InvenSense)**: El sensor inercial de TDK es un módulo multichip compuesto por un MPU6050 (Contiene un accelerómetro y un giróscopo triaxial con un *procesador digital de movimiento* (DMP)) y un AK8963 (un magnetómetro digital triaxial). El sensor posee un encapsulado QFN de 3x3x1 mm con 24 pines.

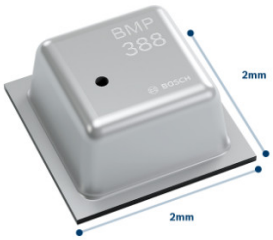
   Este dispositivo nos proporciona medidas del accelerometro y el giróscopo a una frecuencia superior al BNO055 pero con una estimación peor de los angúlos que el dispositivo de BOSCH.

3. **BMP388 (BOSCH)**: El BMP388 es un barómetro digital de 24 bits con bajo consumo y bajo ruido. Se encuentra en un encapsulado LGA de 10 pines de dimensiones 2 x 2 x 0,75 $mm3$.

   Aunque el autopiloto cuenta con este sensor, éste no se ha utilizado en este trabajo debido a que no se tiene en cuenta la altitud en los controladores.



(a) Sensor BNO055



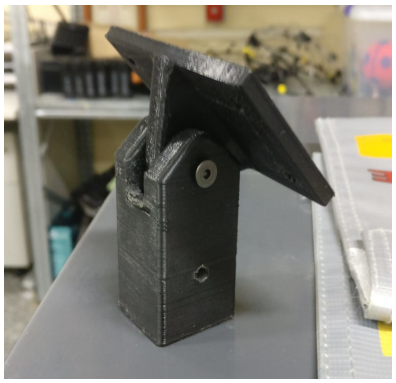
(b) Sensor BMP388

4.5. Banco de pruebas

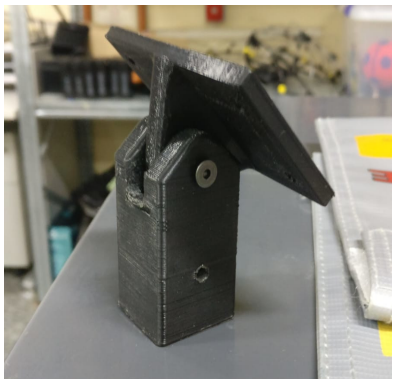
Para poder realizar la experimentación real de forma segura, se han diseñado distintas estructuras para poder sujetar al cuadricóptero permitiéndole rotar con distintos grados de libertad (GdL) en función de la estructura. Estas uniones han permitido poder probar distintos controladores de forma segura y controlada.

Se pueden distinguir 2 tipos de estructuras en función de sus grados de libertad:

- **Rótulas con 1 único grado de libertad**: Para las primeras pruebas de los reguladores es fundamental poder descomponer el control en problemas más sencillos, en este caso permitiendo al sistema rotar en 1 grado de libertad. Se han construido 2 versiones de la misma rótula, una permite el movimiento en Pitch y la otra lo permite en Roll. Esto permite desacoplar los bucles de control y poder probar distintos algoritmos de control.



(a) Motores LHI MT2204 II empleados



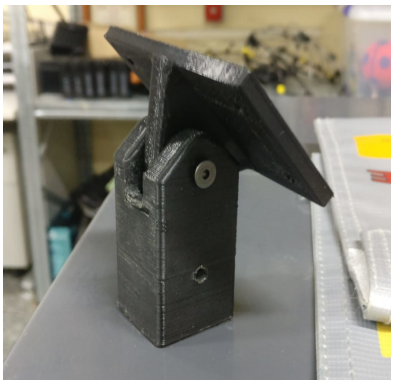
(b) ESC Multistar Race 4 in 1 30A BLHeli empleado

  Estas uniones son sencillas y robustas lo que nos da seguridad a la hora de poder probar y ajustar los reguladores. Las restricciones de movimiento de estas uniones permiten rotaciones de $\pm 60°$ en el angulo permitido.

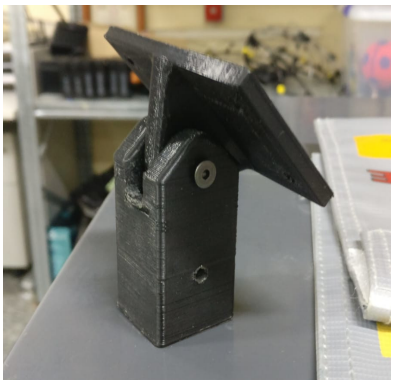
- **Rótula con múltiples grados de libertad**: Después de conseguir estabilizar al cuadricóptero en pitch y en roll de forma individual la siguiente aproximación consiste en emplear rotulas con 3 GdL. Sobre estas uniones también se han realizado 2 versiones. La primera consta de una única rotula con sus 3 grados de libertad con restricciones de movimiento de:

$$-60° \leq \varphi \leq 60° \quad \text{Roll}$$
$$-60° \leq \theta \leq 60° \quad \text{Pitch}$$
$$-180° \leq \psi \leq 180° \quad \text{Yaw}$$

La segunda consta de acoplar 2 juntas esféricas como las anteriores una a continuación de la otra, lo que permite, además de disminuir las restricciones de movimiento en las rotaciones, pequeños desplazamientos en el espacio tridimensional.



(a) Motores LHI MT2204 II empleados



(b) ESC Multistar Race 4 in 1 30A BLHeli empleado

imagen rotulas y/o CAD

Software

Durante el transcurso de este trabajo se ha dividido el desarrollo del software en varias partes, en las cuales se profundizará a continuación.

5.1. Software del Autopiloto

El autopiloto es la parte del multirrotor que se encarga de generar las acciones de control o comandos, que permitan que el dron llegue a un determinado estado. Para generar esas acciones de control, el autopiloto estima el estado de la aeronave en cada instante mediante las lecturas de las IMUs.

5.1.1. Estimación del Estado

Para estimar el estado, el autopiloto toma las medidas procedentes de las 2 IMUs a través del protocolo I2C con una frecuencia máxima de 100Hz. Posteriormente fusiona las medidas de ambos sensores empleando un filtro complementario y un filtro paso bajo para reducir el ruido de alta frecuencia de los sensores. Con este método se consigue estimar el estado deseado $S_t = (\varphi, \theta, \psi, \dot{\varphi}, \dot{\theta}, \dot{\psi})$ consiguiendo minimizar la deriva de la estimación (lecturas BNO055 muy estables) sin perder la reactividad de las estimaciones que proporciona el MPU6050.

Profundizo en el filtrado?

5.1.2. Generacion de comandos

OFFBOARD ->Comunicacion a traves de un socket ROS

ONBOARD ->Algoritmo interno a Freq variable

5.2. Entorno de simulación

drones peligrosos

Debido a la naturaleza del apredizaje por refuerzo, este requiere de la interacción de un agente con un entorno para que el agente aprenda que acciones son las que debe tomar en cada estado. Esto significa que al comienzo del entrenamiento el agente realiza acciones aleatorias para poder explorar cuales son las que le proporcionan una recompensa mayor.

Debido a esta forma de explorar, el agente requiere de una gran cantidad de pruebas, de ensayo y error, hasta que consigue aprender, por lo que no es conveniente realizar todas estas iteraciones en un modelo real.

Por estas razones se necesita de un entorno de simulación para poder validar los algoritmos y poder generar modelos entrenados en simulación sin deteriorar el equipo real. Además el entorno en simulación te permite entrenar distintos agentes simultáneamente y reducir los tiempos de entrenamiento.

Para el entorno de simulación nos hemos basado en Gym citar,una librería escrita en Python y desarrollada por la compañía OpenAI, que permite desarrollar y comparar algoritmos de aprendizaje por refuerzo. Esta librería te permite generar un entorno con el cual interactúe el agente, sin importar la implementación de éste, permitiendo comparar el rendimiento de distintos agentes sobre el mismo entorno. IMAGEN GYM

Como entorno se ha partido del entorno GymFC ha partido del entorno de simulación GymFC, desarrollado por William Koch et al. [5]. Este es un entorno de simulación utiliza Gazebo 9, un entorno de simulación 3D de código abierto ampliamente utilizado en el campo de la robótica. En este entorno se simula el comportamiento de un multirrotor, concretamente el de un modelo del cuadricóptero IRIS. imagen GYMFC

Para acercar la simulación a la configuración de los experimentos que se realizarían posteriormente en la plataforma real, se ha situado una articulación en su centro de gravedad que fija la posición de su centro pero permite rotar el numero de grados de libertad que se predefina.

Para probar distintos agentes con distintos algoritmos se han empleado la implementación de los algoritmos *state of the art*, del repositorio de GitHub *Stable-Baselines* [9]. Inicialmente se comenzó empleando las *Baselines* de OpenAI [10] pero se decidió cambiar a las *Stable Baselines* por la limpieza del código y la facilidad para realizar experimentos con diversos algoritmos e hiperparámetros.

¿hablo un poco sobre transfer learning?

5.3. Pruebas Reales

Para las pruebas reales se han empleado

Ademas se ha empleado ROS Melodic para conectar este entorno con el dron real.

5.4. Descripción del equipo

Para el desarrollo del trabajo se ha empleado un portatil MSI-GE62 empleando Windows 10 para el desarrollo CAD y Ubuntu 18.04 LTS para el resto de las tareas. El equipo cuenta con 16GB de RAM DDR4, un procesador Intel i7-6700HQ de 8 núcleos a 2.60GHz y una GPU GeForce GTX 970M con 3GB de memoria dedicada.

Metodología (Problem Formulation)

El objetivo del trabajo es estabilizar un UAV usando algoritmos de control basados en una red neuronal entrenada empleando algoritmos de aprendizaje automático. Los principales componentes que intervienen en el agente son el estado, las acciones y la recompensa, para cada problema hay un conjunto que estados, acciones y funciones de recompensa que pueden llevar a que el agente aprenda.

6.1. Diseño del estado

El autopiloto cuenta con 2 IMUs para poder obtener datos sobre su estado. Se quiere estabilizar el dron en una orientación concreta, por lo tanto el estado que se ha diseñado consta de 6 parámetros:

$$S = (\varphi, \theta, \psi, \dot{\varphi}, \dot{\theta}, \dot{\psi}) \qquad \varphi, \theta, \psi, \dot{\varphi}, \dot{\theta}, \dot{\psi} \in [-1, 1] \tag{6.1}$$

Siendo φ, θ y ψ los ángulos de alabeo (*roll*), cabezeo (*pitch*) y guiñada (*yaw*) del dron y $\dot{\varphi}, \dot{\theta}$ y $\dot{\psi}$ sus respectivas velocidades. Para favorecer la convergencia del aprendizaje, se ha normalizado el estado para que todas sus componentes estén comprendidas dentro del intervalo $[-1, 1]$.

Los ángulos proporcionan información sobre el estado actual y la velocidad angular sobre los estados pasados, es decir, proporciona cierta información temporal.

Para obtener la estimación de orientación se han fusionado las medidas de las 2 IMUs del autopiloto utilizando un filtro complementario, para así conseguir una buena estimación estática junto con una buena respuesta dinámica.

6.2. Diseño de las acciones

Al trabajar con un quadricóptero podemos actuar sobre la potencia que se le entrega a los motores, por lo que cada acción que realice el agente constará de 4 campos:

$$A = (T_1, T_2, T_3, T_4) \qquad T_i \in [-1, 1] \tag{6.2}$$

Siendo T_i la potencia *(Thrust)* normalizada entregada a cada motor. Un valor de $T_1 = -1$ significa que el motor 1 estaría girando a la mínima potencia permitida y un valor de $T_1 = 1$ corresponde a que el motor 1 estaría girando a la máxima potencia.

completar con la transformacion del mundo -1,1 al mundo 1000,2200 μS

6.3. Diseño de la función de recompensa

La función de recompensa rige la forma en la que la red va a configurar sus pesos, por lo tanto, cómo se va a comportar el agente en un estado determinado. Para conseguir que el agente responda de la forma deseada se han probado una gran variedad de funciones de *reward*, optando finalmente por:

$$R_t = \left(1 - \frac{|\varphi| + |\theta| + |\psi|}{3}\right)^3 \tag{6.3}$$

Con esta funcion de recompensa BLA BLA BLA

Experimentos

Pa

Discusión

Conclusiones y trabajo futuro

Índice de figuras

Índice de tablas

Bibliografía

[1] H. J. Kim, M. I. Jordan, S. Sastry, and A. Y. Ng, "Autonomous helicopter flight via reinforcement learning," in *Advances in neural information processing systems*, 2004, pp. 799–806.

[2] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental robotics IX*. Springer, 2006, pp. 363–372.

[3] T. Dierks and S. Jagannathan, "Output feedback control of a quadrotor uav using neural networks," *IEEE transactions on neural networks*, vol. 21, no. 1, pp. 50–66, 2010.

[4] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[5] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, p. 22, 2019.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[7] "Esp32 technical reference manual," Espressif Systems, 2018. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[8] "Esp32 datasheet," Espressif Systems, 2019. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[9] A. Hill, A. Raffin, M. Ernestus, A. Gleave, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[10] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.