Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería Eléctrica
IEE2913 - Diseño Eléctrico 1-2019

Informe Final

Cristóbal Ferrer y Felipe Neut

1. Descripción del proyecto

El proyecto trata de un vehículo aéreo no tripulado (UAV). El UAV puede mantener una altura fija, envidada desde el computador a este, junto con la estabilidad ante perturbaciones externas. Para lograr todo esto cuenta con un conjunto de sensores que son capaces de medir y compensar los puntos mencionados anteriormente. Como es de imaginarse se cuenta con ciertas medidas de seguridad ya que se trata de objetos (aspas) girando a gran velocidad por lo que existe un riesgo asociado. Por último la alimentación del cuadricóptero es independiente, en este caso mediante una batería que es capaz de alimentar a los motores que consumen corrientes de hasta 20A.

2. Requerimientos y especificaciones

La definición de vehículo aéreo no tripulado (UAV) es bastante general, por lo que comenzaremos acotando las opciones. Primero que todo, dadas las especificaciones, tomaremos la primera decisión de construir un VTOL o Vertical Take-Off and Landing que se refiere a un vehículo aéreo con la capacidad de despegar y aterrizar de forma vertical como cuadricópteros, helicópteros y otros. Dentro de los dispositivos anteriormente mencionados, nosotros decidimos utilizar una topología tipo cuadricóptero. Esto debido a su estabilidad en los tres ejes y la gran cantidad de documentación que existe al respecto.
Como es de suponerse es necesario definir varios aspectos a cumplir, es decir requerimientos. Dentro de las especificaciones de los proyectos no se explicitaron muchos por lo que esto se hará aquí.
El vehículo debe poder mantener una altura fija, envidada desde el computador a este. Junto con esta altura este se debe mantener quieto sobre un lugar específico que corresponde al lugar de despegue. El drone obviamente debe ser capaz de levantar su propio peso, si no que también debe ser capaz de soportar ciertas cargas y perturbaciones sin perder la orientación y ubicación. Para lograr todo esto debe contar con un conjunto de sensores que sean capaces de medir y compensar los puntos mencionados anteriormente. Como es de imaginarse se debe contar con ciertas medidas de seguridad ya que se trata de objetos (aspas) girando a gran velocidad por lo que existe un riesgo asociado. Por último la alimentación del cuadricóptero debe ser independiente, es decir mediante una batería que sea capaz no solo de alimentar al controlador, si no que a los motores que suelen consumir una gran parte de la corriente.

Las especificaciones del vehículo son:

- Material de construcción: Aluminio
- Cantidad de motores: 4
- Tipo de motores: BLDC
- Modo de control de los motores: PWM
- Alimentación: Batería LiPo 4S (16.8V), 1800 mAh.
- Controlador: STM32F407
    - Utiliza un Cortex^{TM}-M4 core + 168 MHz.
    - Programación en lenguaje C.
    - Unidad de punto flotante.
    - 17 timers de 16 - y 32 bits.
    - 15 interfaces de comunicación: 6x USARTs, 3x SPI, 3x I2C, 2x CAN, SDIO.

- Comunicaciones inalámbrica:

  Para efectuar la comunicación inalámbrica entre el computador y el drone, se utilizó el módulo **HC-12**, que es un puerto serial inalámbrico con un alcance de 1000 metros y que posee una antena con una frecuencia de funcionamiento entre $433,4 - 473$ MHz.

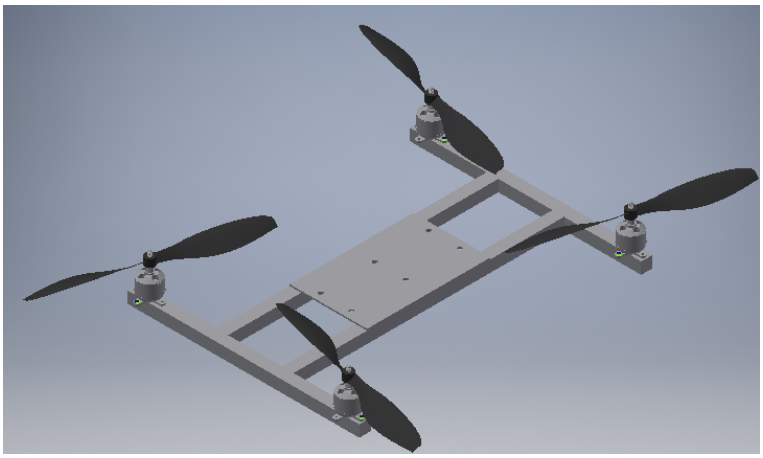
- Sensores

  1. **Sensor Time of flight VL53L0X**: Este sensor tiene relativamente la misma velocidad que el anterior pero un mayor rango (200cm) y robustez. El funcionamiento se basa en la emisión de un pulso de luz infrarroja y el tiempo que demora en 'volar' hasta el objeto ida y vuelta. Este cuenta con un protocolo de comunicación I^2C.
  2. **MPU-9250**: Incluye un acelerómetro de 3 ejes, un giroscópio de 3 ejes y un magnetómetro de 3 ejes. CUenta con dos protocolos de comuncación, SPI e I^2C.

3. Prototipo final

3.1. Presentación del prototipo

- En la Figura 3.1 se puede ver el bosquejo del prototipo. Se decidió utilizar una armadura o frame tipo H, debido a que simplifica el método de unión y el torque que se ejerce en estas. Para la unión de las piezas queremos utilizar placas de aluminio con forma de T que irán en cada una de las uniones. Las dimensiones son de 300x200mm, estas fueron elegidas para proporcionar estabilidad en base al empuje de los motores. En términos de el centro de masa es sencillo determinar que este se encuentra en el centro, ya que la construcción es simétrica.



Ahora, el diseño final se puede ver en las siguientes imágenes,

Figura 1: Diseño final

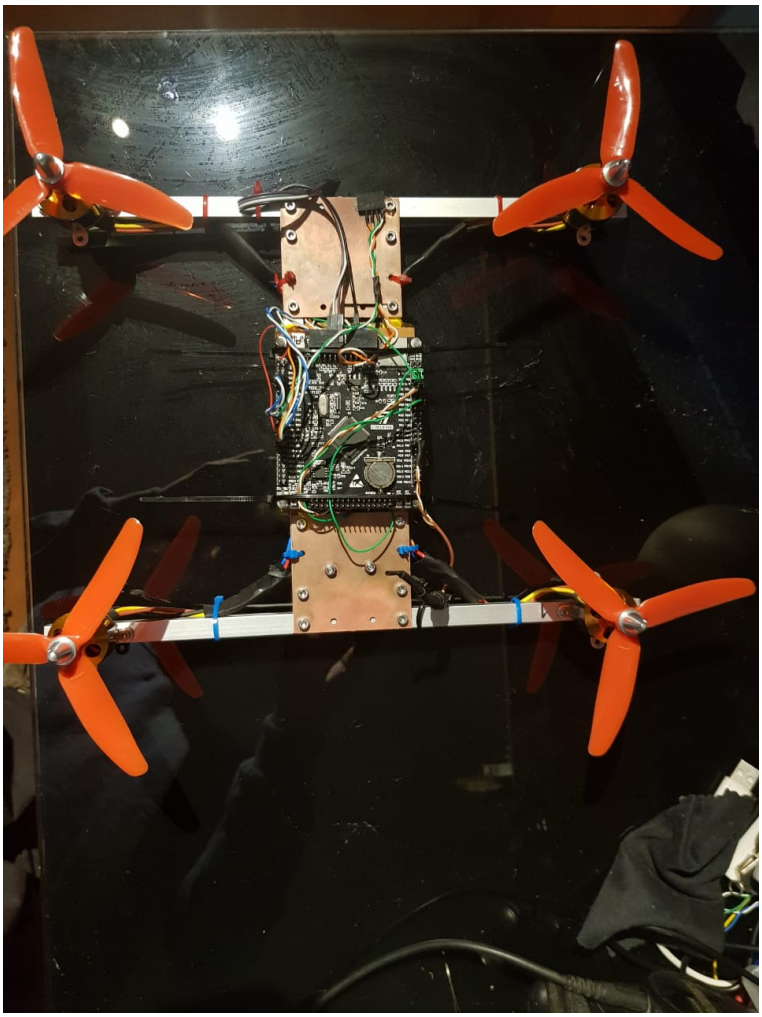


Figura 2: Diseño final

De la Figura 2 se logra apreciar el drone por arriba, detectando en color negro el microcontrolador STM32F4 nucleo.

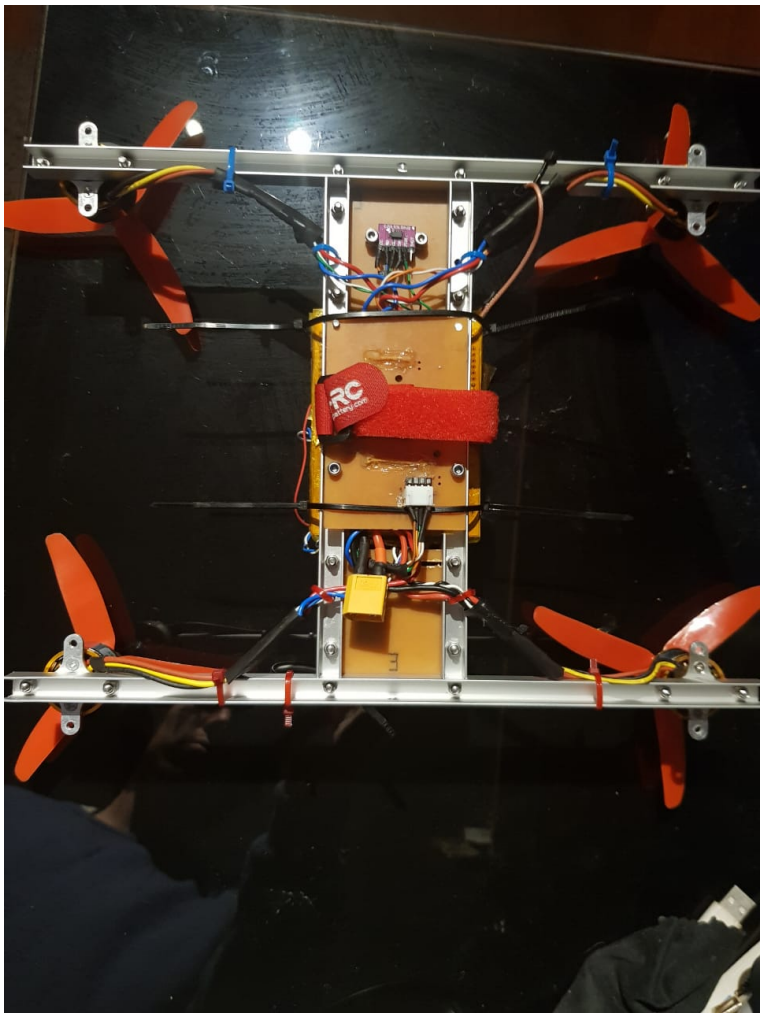


Figura 3: Diseño final

De la Figura 3 se logra apreciar el drone por abajo, detectando en color morado el sensor V53LOX.

- **Circuitos impresos**

  En las siguientes figuras es posible ver los circuitos impresos de potencia (PDB) y de control.

  El primero, PDB se encarga de recibir la alimentación de la batería, convertir esta a dos rieles de voltaje de 5V y 3.3V para el microcontrolador y periféricos, alimentar los motores mediante los Mosfet de seguridad y proporcionar un punto de montaje para la batería. La primera imagen corresponde a la placa de distribución de potencia, a la que se le tuvo que agregar soldadura sobre las pistas de potencia para que fueran capaces de manejar la corriente que circula por estas.

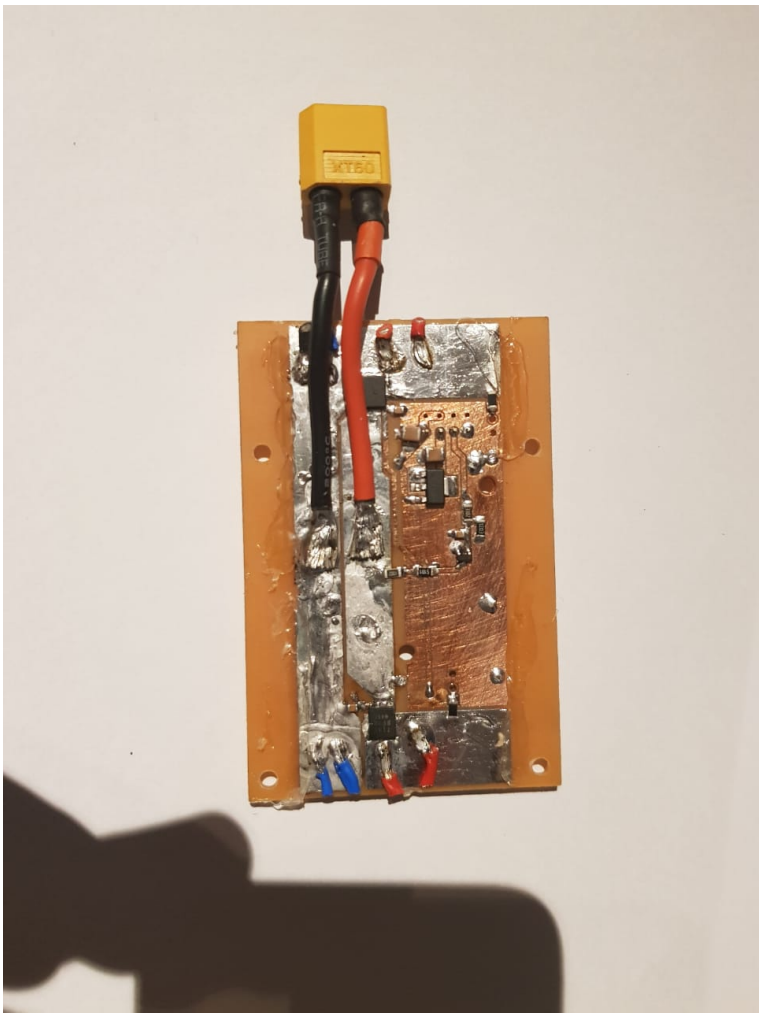


Figura 4: Circuito impreso de potencia

Por su parte la placa de control alberga la mayoría de los periféricos, microcontrolador y todas las conexiones del circuito. Se realizo un montaje en Stack, en el que bajo el microcontrolador se encontraban las conexiones y periféricos. Se tuvo especial cuidado de proteger la unidad de medición inercial (MPU-9250) de interferencias externas, por lo que se blindó (Shielding) esta de EMI y otras interferencias magéticas.

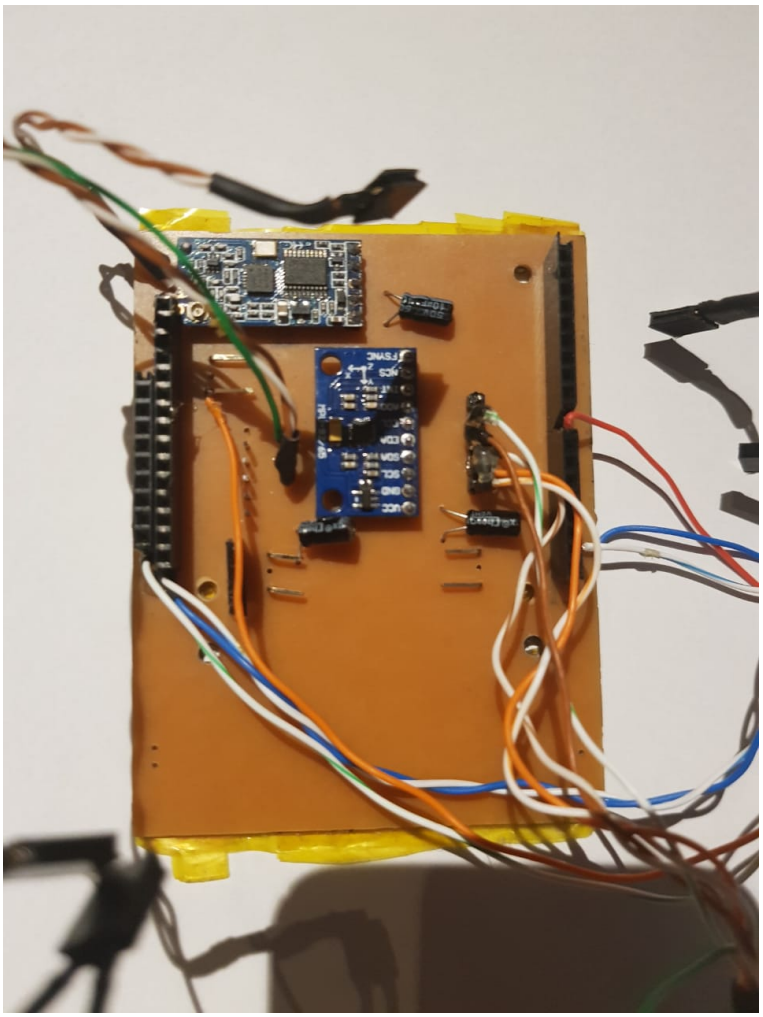


Figura 5: Circuito impreso de control

En ambas placas destaca el uso de componentes SMD, que nos permitieron integrar todo el sistema de potencia más control en 1.5cm de altura.

3.2. Análisis de cumplimiento de especificaciones

El prototipo desarrollado tiene la capacidad de controlar con una gran velocidad de respuesta y estabilidad los ejes de *pitch* y *roll* frente a distintas perturbaciones. Duratnte la fase de *Tunning* se realizaron diferentes pruebas, tales como escalones en la referencia del ángulo, perturbaciones externas, etc. Gracias al método utilizado para encontrar los valores de las constantes, mencionado más adelante, se logró encontrar valores que mantenían el UAV sumamente estable. Es por lo anterior que se lograron las especificaciones mínimas de estabilidad.

A su vez, se podía controlar la altura del mismo, manteniendo la estabilidad, por lo

que se cumplieron con las especificaciones mínimas de control de altura. Esto representó un problema en un principio, ya que a medida que los motores aceleraban para mantener la altura, su punto de operación cambiaba y por lo tanto los valores de las constantes de los controladores perdían validez. Por su parte a medida que la batería se iba descargando, los motores comenzaban a perder fuerza, por lo que las constantes mencionadas anteriormente perdían validez sobre el control.

No obstante, no se pudo controlar la posición (x,y) del drone, por lo tanto, siempre se tenía un *drift*. Esto provocaba que el drone tuvierá que estar amarrado de forma mínima a unos cordeles, que permitían detener al drone en caso de que el *drift* fuera muy alto. Se comenzó a desarrollar un controlador de posición absoluta. Esto se lograba mediante la plataforma OpenCv de reconocimiento de imágenes y un celular que iba montado en la parte inferior del vehículo. Mediante un patrón impreso en una hoja de $1m^2$ se lograba reconocer la posición, que posteriormente era enviada a un microcontrolador (Arduino) que la procesaba y enviaba al controlador principal (STM32F4). Finalmente este control no se implementó, por lo que en este sentido no se lograron las especificaciones mínimas de control de posición.

3.3. Costos

Presupuesto

Item	**Precio**	**Cantidad**	**Total**	**Referencia**
Motores+ESC	5756	4	23024	Link
Stm32f407	13000	1	13000	
Lipo	29700	1	29700	Link
aluminio	5000	1	5000	
MPU9250	4990	1	4990	Link
VL53L0X	6990	1	6990	Link
HC-12	4990	2	9980	Link
10 % materiales menores	9268	1	9268	
Total		1	**101952**	

4. Diagrama de bloques de alto nivel

En la Figura 6 se encuentra el diagrama de bloques de alto nivel.

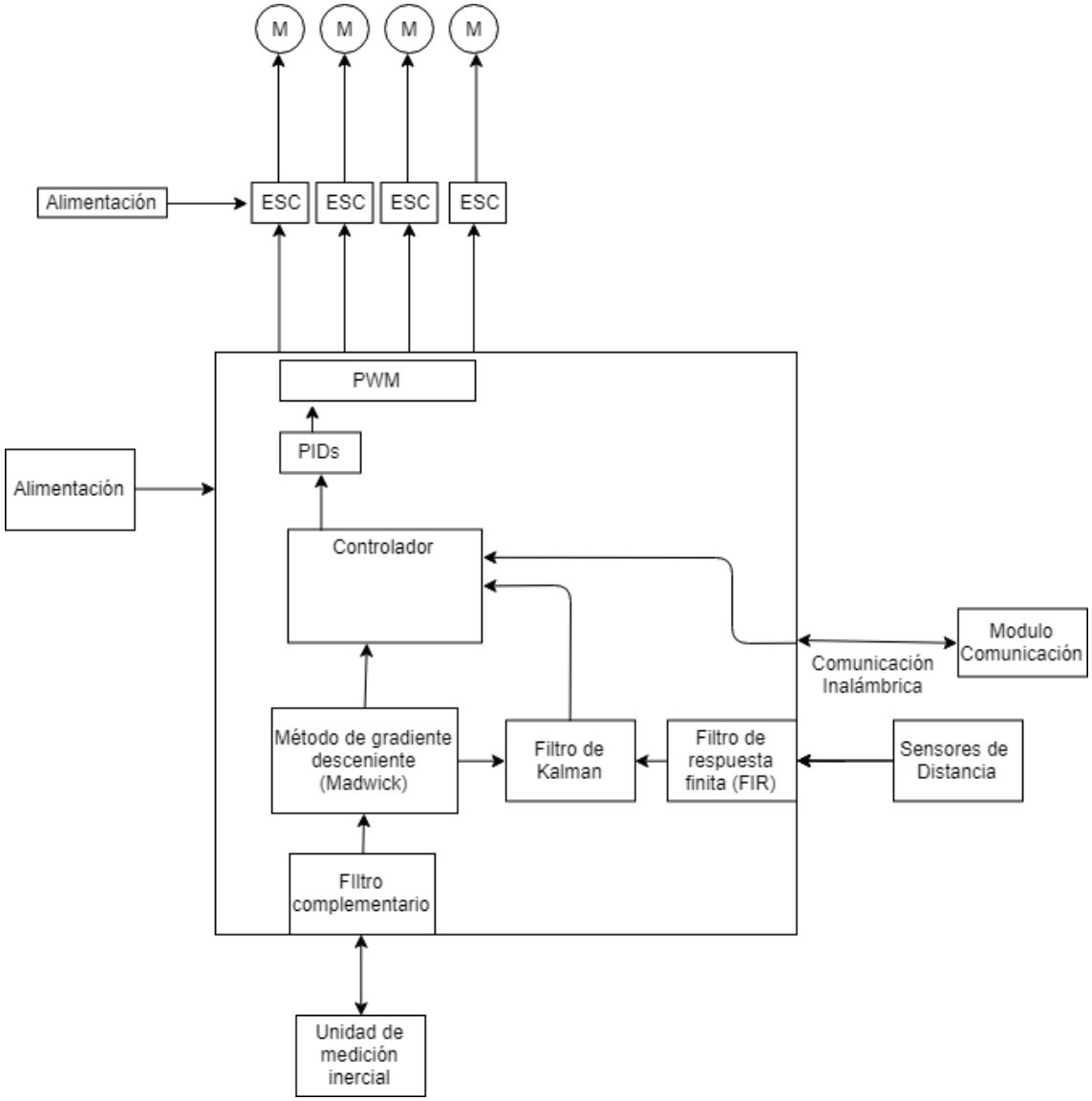


Figura 6: Diagrama de bloques de alto nivel.

Lo que se hace es obtener los resultados del magnetómetro, giroscopio y acelerómetros mediante el módulo MPU-9250. Dichos valores pasan a través de un filtro complementario, que complementa (...) los valores del acelerómetro y giroscopio mediante un filtro pasa altos y pasa bajos respectivamente. Como resultado de esto se reciben quaterniones. Los quaterniones ingresan al filtro de Madgwick, que consiste en un método de gradiente descendiente, que converge a un resultado ara la posición absoluta. Luego los quaterniones son transformados a grados Euler. Ya con la orientación estimada del drone en grados Euler, estos ingresan

entonces al controlador en formato de *pitch*, *roll* y *yaw*.

Finalmente, el controlador entrega los valores estimados de *pitch*, *roll* y *yaw* al bloque de PIDs que genera los valores de los ciclos de trabajo de las PWM necesarios para estabilizar el cuadricóptero.
Sumado a lo anterior, se implementó un filtro de Kalman, que tenía como objetivo compensar la baja velocidad del controlador de altura, ya que este controlador se encuentra condicionado por el sensor de altura (VL53LOX) a 33 Hz, en comparación del loop de control que corría a 1kHz. Luego, lo que se hacía era estimar la altura en base a la medición real de altura, y a las aceleraciones que entregaba el filtro de Madgwick.
También se implementó un filtro de respuesta finita, el cual fue utilizado como filtro pasabajos para eliminar las componentes de alta frecuencia de la medición del sensor de distancia.
El módulo de comunicación empleado fue el HC-12, que permitía una comunicación inalámbrica, de tal modo de manipular el encendido, apagado, referencia de altura, entre otros, en el controlador.

5. Diagrama de bloques de bajo nivel

En la imagen 7 se puede el diagrama de bloques de lo que sería la placa de distribución de potencia. Comenzamos con la batería que tiene un rango de voltaje de 16.8-12.8V, este voltaje es regulado a 5V por un conversor DC-DC (SMPS) para alimentar los periféricos y STM32F4. Es importante dimensionar correctamente este conversor ya que todo el sistema depende de él. Para asegurar márgenes de seguridad se utilizará un conversor conocido, Lm2596 que tiene una corriente máxima de 3A.

Luego se puede ver un regulador lineal que entregará el voltaje para los periféricos que utilizan 3.3V, en este caso Ld1117. Este tiene una corriente máxima de 800mA, considerando que se estiman unos 50mA de consumo en el riel de 3.3V, nuevamente queda con un generoso margen.

Finalmente cabe destacar el circuito de Mosfet de seguridad y receptor RF independiente. La seguridad es una de las consideraciones más importantes al diseñar un prototipo, especialmente cuando hay elementos mecánicos girando a altas velocidades. Utilizando un simple receptor RF como el xy-mk se puede obtener a la salida una señal lógica que indica si se energizan los motores, los cuales quedan inmediatamente des energizados al quitar la señal. De esta manera se puede apagar toda la estructura de potencia en caso de que sea necesario. Nuevamente es importante tomar en consideración las capacidades de los componentes, en este caso se opto por un Mosfet IRF540NS capaz de manejar 40A nominalmente y peaks de hasta 100A.

Finalmente se puede ver la salida de potencia hacia cada uno de los motores, pasando por el sistema de seguridad.

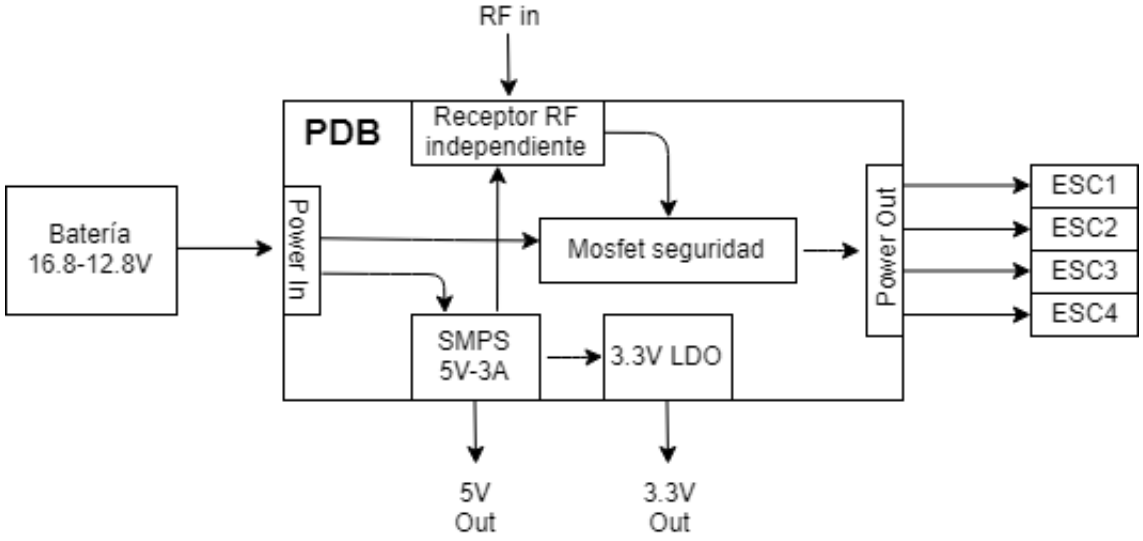


Figura 7: Power Distribution Board.

En la imagen 8 se puede ver el diagrama del cerebro detrás de la operación. A pesar de que conceptualmente es muy simple, la complejidad en este bloque es a nivel de código. Es

difícil representar en forma visual la complejidad del código y tareas que se llevan a cabo, pero en la figura 8 se hace un intento. El corazón de este bloque está en el bloque de PIDs. En un comienzo se procesa toda la información proveniente de los sensores y se adapta para poder ser entregada al control, luego de que el control realiza los cálculos pertinentes se aplican distintas señales PWM dependiendo del control a realizar.
En la imagen 8 se destaca el módulo HC-12 necesario para la comunicación inalámbrica, como también el sensor de altura VL53LOX, y el IMU MPU-9250.

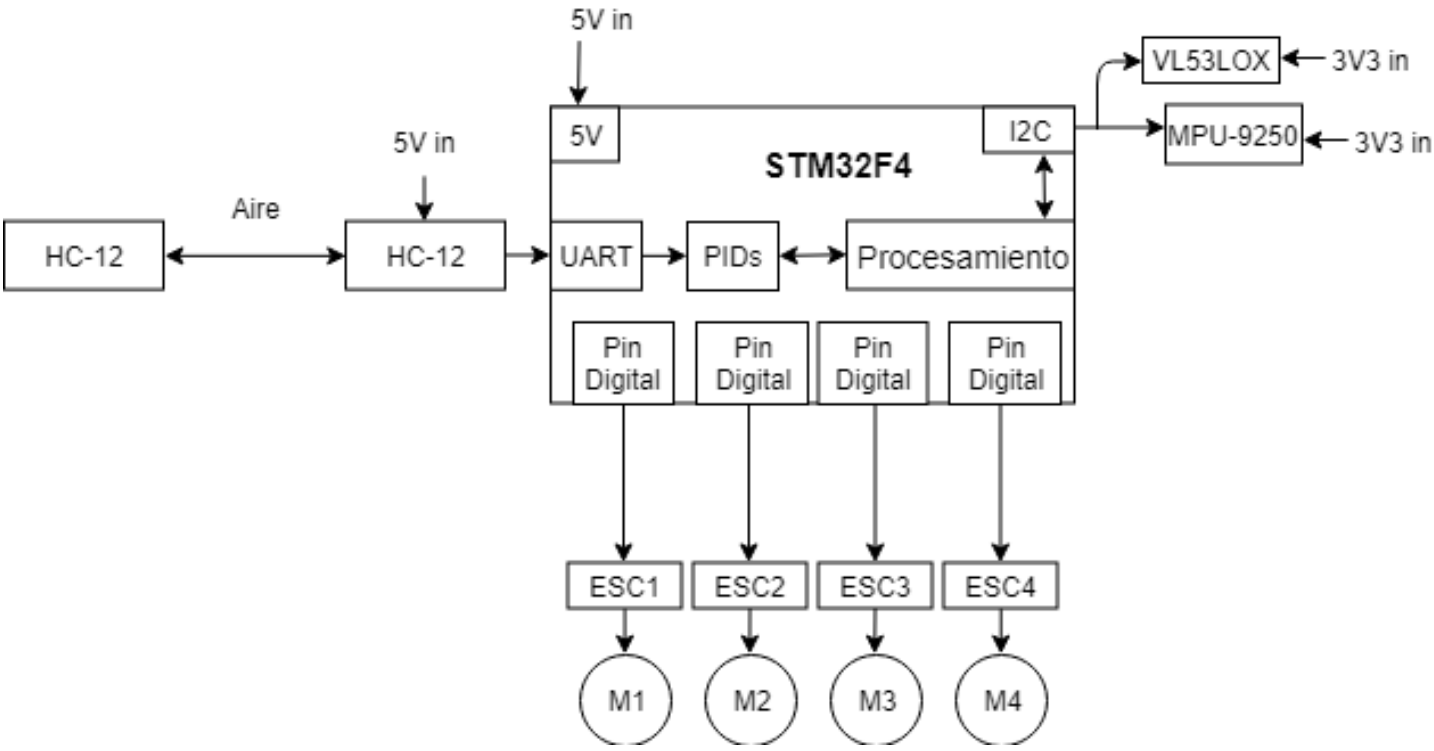


Figura 8: Sistema de control principal.

En la figura 9 se puede ver el diagrama conceptual y de conexiones de el inversor con el motor. Este es relativamente sencillo, por una parte entra la alimentación de la batería, la cual es DC, que posteriormente es convertida a AC y trifásica mediante un puente H para alimentar el motor. La PWM viene de pines digitales de la STM32F4, que generan una señal proporcional a la velocidad del motor. Es importante destacar que se deben separar las tierras de potencia y señal ya que si no las altas corrientes que circulan por la parte de potencia podrían afectar la señal.

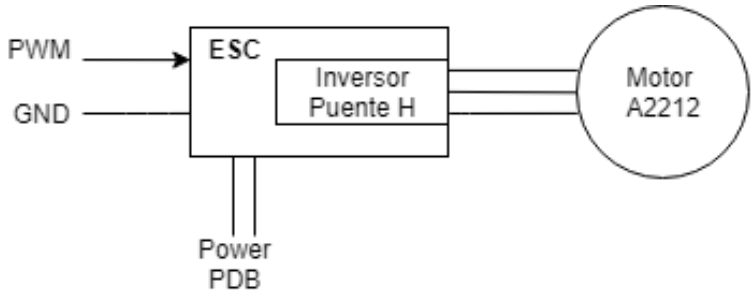


Figura 9: ESC con sus respectivas conexiones.

A continuación se puede ver el diagrama de bloques de el controlador PID mencionado objetivo que se intentó implementar.

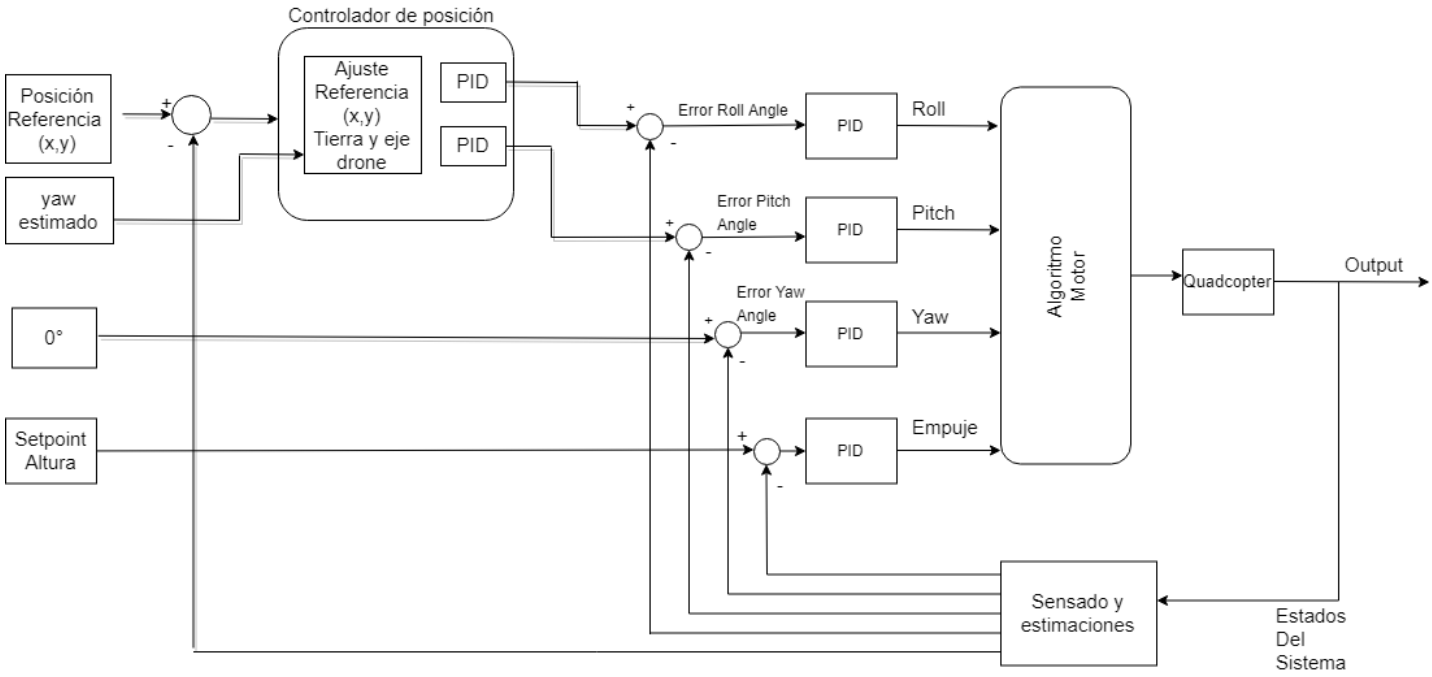


Figura 10: Bloque de control objetivo

La explicación de este diagrama con gran detalle se encuentra en la sección de implementación, no obstante, está basado en 6 PID's para controlar la orientación y posición absoluta del drone, de tal forma de dejarlo estático en una posición de referencia (x,y) mediante el controlador de posición, como a su vez elevarlo a algún setpoint fijado mediante el PID de altura. Finalmente, la idea es la implementación de un control en cascada, que en base a la posición de referencia (x,y) fijada, el ouput del controlador de posición son las referencias de pitch y roll, que entran como referencia a sus 2 PID respectivos.
No obstante, como se vió en la presentación el controlador de posición no pudo ser implementado en su totalidad. A pesar de que se logró estimar la posición (x,y) mediante una

aplicación de Android denominada OpenCV Bot, no se pudo ajustar de forma correcta el controlador, por lo tanto el diagrama de bloques de bajo nivel que realmente mostró funcionalidad fue,

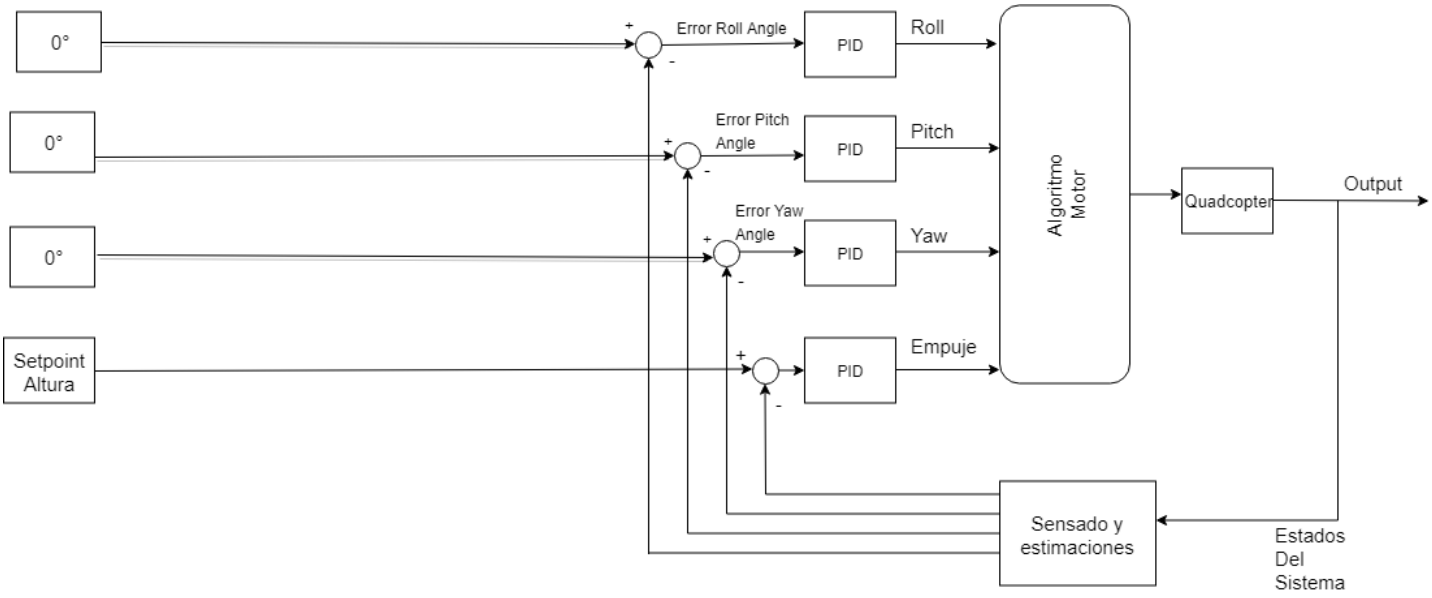


Figura 11: Bloque de control implementado

6. Implementación y resultados

6.1. Prototipo final

En la siguiente tabla se puede ver el peso de cada uno de los componentes y por consecuencia el peso total. Dentro de la tabla se agregó un peso de componentes menores del 20 % que simula el peso del resto de los componentes, junto con errores en las mediciones.

Item	**Peso[g]**
Bateria	185
ESC	100
Motores	240
Frame	165
STM32F4	65
20 %	100
Total	**855**

Tabla 1: Peso de los componentes

6.2. Distribución de potencia

La distribución de potencia y alimentaciones se realiza mediante la PDB.

En esta se recibe el voltaje de la batería (16.8V) que es convertido en 5V mediante un SMPS Buck converter. El riel de 5V alimenta el microcontrolador (STM32F4), el módulo de comunicación inalámbrica y un regulador lineal de 3.3V.

Por su parte el regulador lineal de 3.3V alimenta los periféricos, tales como Vl53l0x y MPU9250.

La otra función de la PDB es proporcionar el apagado por Hardware de los motores mediante Mosfet. Como los ESCs comparten tierra con el microcontrolador, por el sistema de comunicación, se tuvo que elegir un Mosfet de canal P para controlar la alimentación positiva de los ESC. El Mosfet elegido es el Si7137DP que soporta corrientes continuas de 60A. Para controlarlo se utilizó un level shifter que proporcionaba el voltaje de source al momento de apagarlo, y era controlado por el microcontrolador.

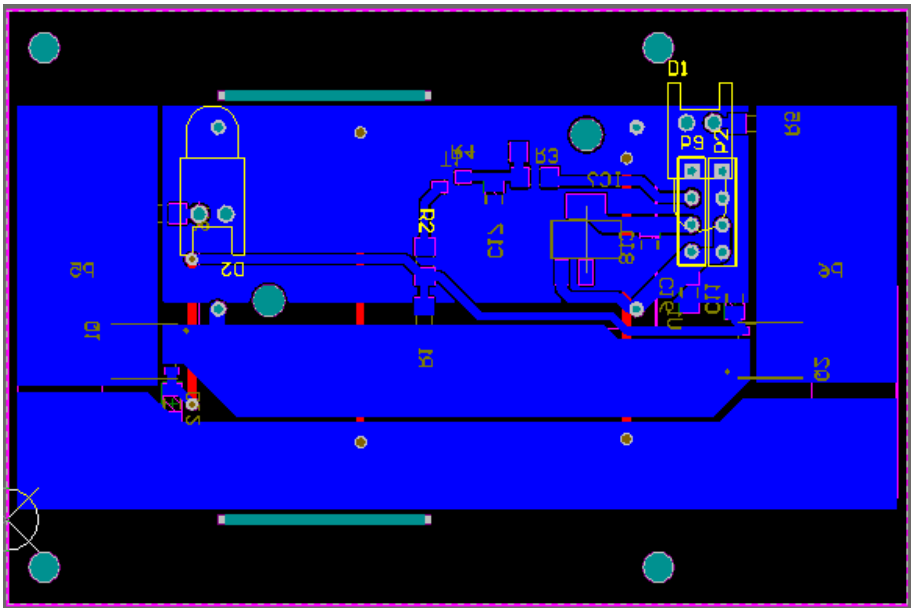


Figura 12: Circuito impreso PDB

6.3. Placa de control

En la siguiente figura es posible ver el circuito impreso utilizado para albergar todos los sistemas de control y medición. En el centro de la placa y también del cuadricóptero, es posible ver el sensor de medición inercial. Este se comunica mediante I^2C con el microcontrolador. En la esquina derecha inferior es posible ver el módulo de comunicación inalambrica HC-12 junto con su comunicación serial correspondiente.

Para esta placa se tuvieron dos consideraciones importantes, la primera es el ruteo de las lineas de alta velocidad de I^2C que tenían una frecuencia de hasta 400kHz. Estas se hicieron lo más cortas posibles, teniendo en consideración que debían conectar dos sensores ubicados en diferentes lugares, al controlador. La otra consideración fue la de decoupling ya que la alimentación provenía de la placa de distribución, por lo que se agregaron capacitores de $10\mu F$ y $1\mu F$ en todos los módulos.

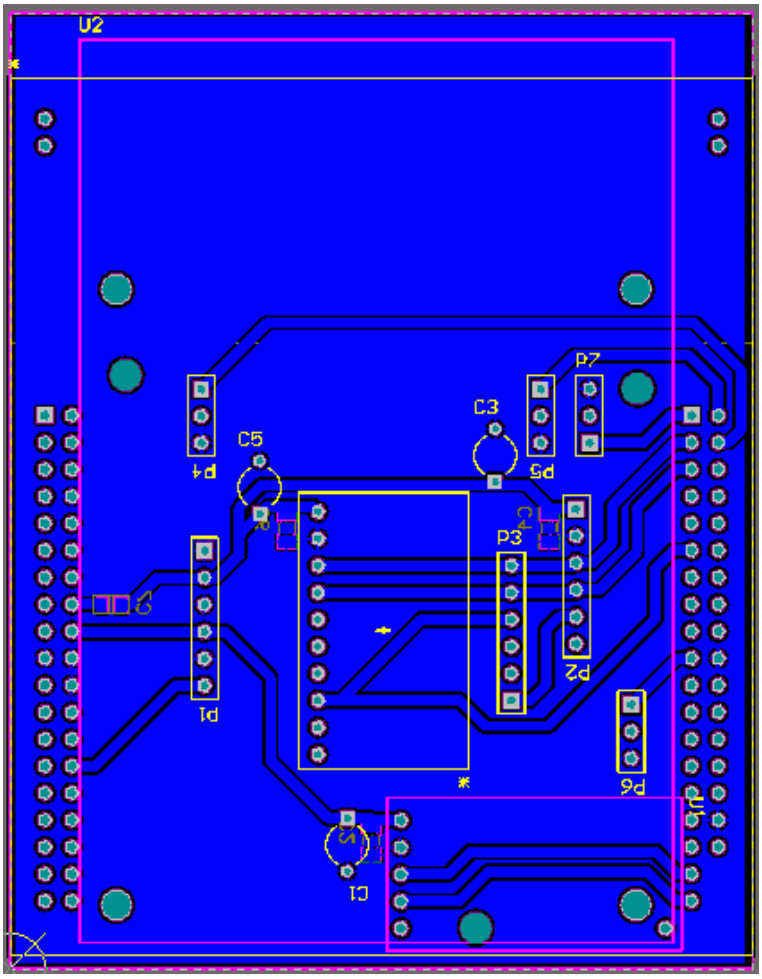


Figura 13: Circuito impreso de control

6.4. Sistema de propulsión

Motor A2212 1000 kV:

- Kv $= 1000\frac{RPM}{V}$
- Corriente sin carga 0.5A.
- Resistencia en Delta $R_{\Delta} = 0.09\Omega$.
- Corriente máxima $I_{max} = 13A$.
- Numero de polos $N_p = 14$

Y para la hélice:

Hélice: **Modelo 5040.**

- Largo: 4”(25.4cm)
- Paso: 4”(11.43cm)
- Peso: 2g

En la figura 14 es posible ver la curva de empuje para la hélice anterior.

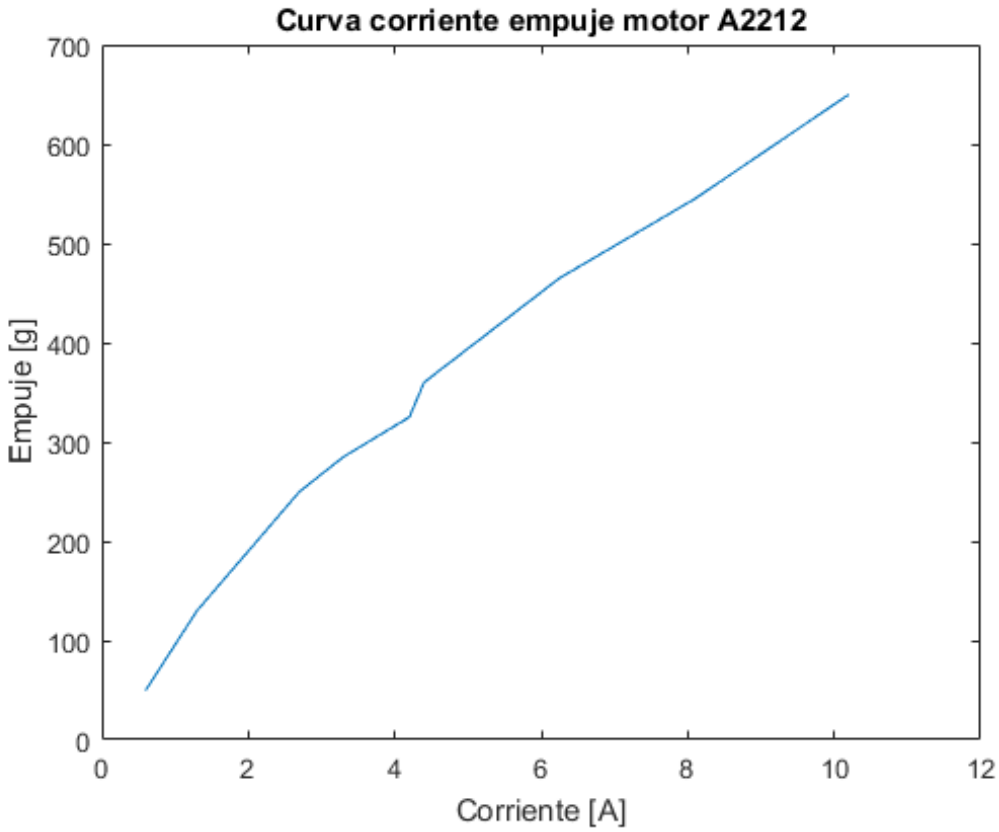


Figura 14: Curva corriente empuje.

De la figura 14 y tabla de pesos es posible inferir la corriente necesaria para obtener u empuje suficiente para sustentar el vuelo. El Drone tiene un peso aproximado de 900g por lo que se necesitan al rededor de 225g de empuje por motor. De la figura podemos ver que los motores utilizados junto con las hélices entregan ese empuje al rededor de los 3A, por lo que la corriente consumida durante el 'planeo' es al rededor de 9A en total.

Una vez que se determinó el valor de las corrientes necesarias para elevar el dron, es necesario determinar esta misma relación, pero ahora con respecto al valor del ciclo de trabajo de la PWM. La vez anterior se realizaron pruebas para un motor y se extrapolaron al cuadricóptero completo. Estos resultados, a pesar de que son una buena representación del comportamiento, no son la realidad. Es por esto que se decidió realizar una prueba con el drone completo y ver en que porcentaje del acelerador este puede mantener su propio peso.

Ayudando con las manos, se logró determinar que para un porcentaje del acelerador de 70 % este lograba mantener su propio peso. A pesar de que este porcentaje pareciera ser muy alto, una vez que logra superar su propio peso todo el rango de potencia restante esta disponible para acelerar. Es importante recordar que se busca estar suspendido en el aire, por lo que teóricamente la aceleración es cero una vez que se llega al equilibrio.

6.5. Control/Modelo de control automático

Esta sección se haya basada en las Figuras 11 y 10.

- **Controlador de altura**: Es el primer controlador que surge a la mente, en el que se controlará la altura del drone en base al empuje de los propulsores. Lo que se hará será sensar la altura y en base a un *setpoint* determinado se calculará el error. Supongamos que la altura del drone es menor que el *setpoint*, por lo tanto el error será positivo. Este error entrará a un controlador PID que hará que el empuje aumente, haciendo que los 4 motores incrementen su velocidad al mismo tiempo, generando con ello que el drone se eleve. No obstante, el escenario no es tan simple debido a que hay una serie de factores externos que inducen en el plano del drone *roll angle*, *yaw angle* y *pitch angle*, los cuales provocarán que el drone se encontrará "ladeadeado"(eje z desviado con respecto a su origen), por lo que además de movimientos verticales, existirán movimientos horizontales, los cuales se ven claramente al hacer la descomposición de fuerzas del empuje. Estos movimientos horizontales sacaran al drone de la referencia (x,y) deseada, por lo tanto, se hace necesario el controlador del siguiente punto.

- **Controlador de posición**: Lamentablemente no pudo ser desarrollado, no obstante se entregará una idea sobre éste. Este controlador nos entrega la facultad de mantener al drone en la posición de referencia (x,y). Hagamos un ejemplo para explicar este controlador. Supongamos la existencia de un viento en dirección oeste que hace que el drone se mueva un poco en dicha dirección. Luego, para hacer frente y que el drone pueda volver a su posición de referencia (x,y), es necesario que se aplique al drone cierto *roll angle* y/o *pitch angle* (en este ejemplo es necesario la existencia de un *roll* hacia la derecha). La presencia de estos ángulos en el eje del drone provocarán que se induzcan ciertos movimientos horizontales debido al empuje que se está aplicando en el eje z. Estos movimientos horizontales inducidos a la derecha permitirán que el drone vuelva a su posición de equilibrio. Es por ello, que este controlador nos permitirá mantener la posición de referencia en el eje (x,y) mediante ciertos *roll angles* y/o *pitch angles* que me permitirán realizar movimientos horizontales en la dirección deseada. La idea de este controlador es la de un *loop cascada*, en el cual en lugar de nosotros entregar la referencia de ángulos, el mismo controlador lo hace.
  Antes de finalizar, resta un detalle en este controlador. Sabemos que la posición (x,y) (sea x norte-sur, y oeste-este) es relativa a a la referencia de la Tierra, mientras que el *roll angle* y *pitch angle* son relativos al eje del drone. Esto provoca que cambios en el *roll angle* no siempre mueve al drone en el 'eje y' referenciado a la Tierra, mientras

que cambios en el *pitch angle* no siempre mueve al drone en el 'eje x' referenciado a la Tierra, si no que depende de que tanto el drone se encuentre rotado en el plano (x,y), o dicho con otras palabras, depende de que tanto desfase exista del *yaw angle* con respecto a la referencia de la Tierra. Es por eso, que si necesitamos mover el drone a un punto específico de una pieza, es necesario saber que tanto desfase existe en términos de *yaw angle* para determinar si el drone debe hacer una variación del *roll angle* y/o *pitch angle* para llegar a esa posición.
Como se detalló previamente, la posición del drone estaba siendo estimada mediante la aplicación OpenCV Boat, que permitía el tracking de un objeto en base a su posición (x,y). De esta forma, se estaba dejando una imagen en el piso, y con un celular android en el drone se iba siguiendo dicha imagen. El problema es que este seguimiento no pudo ser tuneado de la forma que se esperaba para poder presentarlo, por lo que finalmente se decidió descartarlo de la presentación.

- **Controlador yaw**: Recibe como referencia 0° pues no queremos una rotación o giro con respecto al eje z.
- **Controlador pitch y Controlador roll**: Reciben como *setpoint* el valor a la salida del controlador de posición para regular los movimientos horizontantales. No obstante como finalmente el control en cascada no pudo ser implementado, la referencia de de *roll* y *pitch* está fijada en 0°

Todos los errores se realimentan a sus respectivos controladores PID que determinarán la velocidad a aplicar a cada uno de los 4 motores. La implementación utilizada se mostrará a continuación, que es la misma tanto para el *pitch*, como para el roll, yaw y altura.

$$error = setpoint - medicion_{actual}$$

Luego, se tendrá la realimentación del integrador para el caso *pitch* y caso *roll* siguiendo la siguiente lógica:

$$error_{integrador} += (error_{anterior} + error_{actual}) \cdot K_i$$

El derivativo a su vez, será:

$$error_{derivativo} = (error_{actual} - error_{anterior}) \cdot K_d$$

Mientras que el proporcional simplemente corresponde a:

$$error_{proporcional} = (error_{actual}) \cdot K_p$$

Luego simplemente se realiza la suma de los errores anteriores,

$$error_{total} = error_{proporcional} + error_{derivativo} + error_{integrador}$$

Es así como se tendrá un $error_{total_{pitch}}$ y un $error_{total_{roll}}$, los cuales entran al **algoritmo del motor**, el cual consiste en las siguientes ecuaciones,

$$motor_{adelantederecha} = error_{total_{roll}} + error_{total_{pitch}} + error_{total_{yaw}} + empuje_{altura}$$

$$motor_{adelanteizquierda} = error_{total_{roll}} - error_{total_{pitch}} - error_{total_{yaw}} + empuje_{altura}$$

$$motor_{atrasderecha} = -error_{total_{roll}} + error_{total_{pitch}} - error_{total_{yaw}} + empuje_{altura}$$

$$motor_{atrasizquierda} = -error_{total_{roll}} - error_{total_{pitch}} + error_{total_{yaw}} + empuje_{altura}$$

Hablando un poco más sobre el **algoritmo de motores**, un primer detalle a recalcar es la dirección de giro de los motores del drone. Lo que se hará es que ambos motores en un eje (ya sea eje 'x' o eje 'y') giran en la misma dirección, pero opuesta a la dirección de giro del par de motores del otro eje. Esto es necesario para que el empuje, *roll angle*, *yaw angle* y *pitch angle* sean independientes unos de otros, permitiéndonos modificar un valor de alguno de ellos sin alterar al resto.
El algoritmo de motores no es tan directo porque solo contamos con 4 actuadores, pero tenemos 6 grados de libertad. El hecho de que no tengamos un actuador para cada movimiento complica la situación. Por ejemplo, el drone no se puede mover a la izquierda sin primero rotar una cierta cantidad de grados en esa dirección. Esto se ve al hacer la descomposición de fuerzas del empuje en el centro de gravedad cuando el drone se encuentra rotado, y ver como esta tiene componente vertical y horizontal.

Con respecto al ajuste de los valores de las constantes utilizados para realizar el *tunning* de los controladores PID, lo que se hizo fue aplicar un método basado en el método de Ziegler Nichols, no obstante ajustado para el uso en drones. Lo primero que se hace es inestabilizar algún eje (por ejemplo *pitch*), y realimentar exclusivamente con realimentación proporcional hasta inestabilizar la planta, y con dicho valor proporcional utilizar aproximadamente un 0.6 de este valor. Luego, se hace lo mismo con la realimentación integral y derivativa respectivamente, encontrando de este modo valores de referencia para cierto eje. Así sucesivamente se aplica para todos los ejes del drone. Finalmente, dichos valores son ajustados levemente acorde a prueba y error, y condiciones requeridas de velocidad de respuesta, *overshoot*, entre otras. Al valor que se llegó para cada constante, se encuentra a continuación,

Controlador	K_p	K_d	K_i
Pitch	0.35	0.5	0.0001
Roll	0.3	0.5	0.0001
Yaw	0.1	0	0
Altura	1.5	1	0

Mediante los valores especificados previamente, se pudo controlar con una gran velocidad de respuesta y estabilidad los ejes de *pitch* y *roll* frente a distintas perturbaciones. A su vez, se podía controlar la altura del mismo, manteniendo la estabilidad. No obstante, no se pudo controlar la posición (x,y) del drone, por lo tanto, siempre se tenía un *drift*. Esto provocaba

que el drone tuvierá que estar amarrado de forma mínima a unos cordeles, que permitían detener al drone en caso de que el *drift* fuera muy alto.

6.6. Otros resultados

Se procederá a especificar de forma un poco más formal las velocidades de los lazos de control que se están obteniendo actualmante. Es importante destacar que el loop de control de altura funciona a una menor velocidad, con respecto al loop de control de los ángulos del drone. Esto se hizo para intentar acelerar este último loop descrito.

Lectura y procesamiento de componente	**Velocidad del loop de control**
VL530OX (lectura)	33 Hz
Filtro de Kalman (loop control altura)	1kHz
MPU9250 (lectura)	1kHz
Loop de control pitch,roll,yaw	10kHz

Tabla 2: Velocidades de lectura

7. Cumplimiento de Estándares ECMA-287

La fuente de energía presente en el prototipo en base al estándar ECMA-287 es la batería LiPo. La batería LiPo escogida tiene 4 celdas en serie (16.8V) y un valor nominal de corriente de 1800 mAh. Realizando simples cálculos, se puede estimar la corriente promedio que se le pedirá a la batería. Si consideramos que la corriente consumida por los motores es:

$$I_m = 4 \cdot I_p = 12A$$

A lo anterior se le añade la corriente consumida por la electrónica, que será considerado como la corriente consumida por la STM32F4 y otros periféricos:

$$I_e = I_{STM32F4} + I_{otros} = 330mA + 200mA = 0.53A$$

Luego la corriente total será:

$$I_T = I_e + I_m = 12.53A$$

Al estudiar el archivo que presenta los estándares ECMA-287 se pudo concluir que la batería LiPo cumple los requerimientos para ser una *Power Source DC PS3* debido a que excede los límites LPS presentados en el archivo, y tampoco cumple internamente ninguna de las otras condiciones restantes para ser PS2, las cuales se pueden encontrar en el archivo.
A partir de la conclusión anterior en el archivo se enuncian una serie de requerimientos como protección ante una PS3. Entre ellos, el circuito debe contar con la presencia de un *open switch*, conmutadores y *relay contacts*. La experiencia nos ha entregado esta noción de seguridad, así que antes de leer los estándares, ya habíamos diseñado un circuito de apagado por software mediante el uso de Mosfets, cuyo objetivo no es más que la seguridad y protección, como se especificó previamente.
En las normas ECMA-287 se enuncian también una serie de otras medidas de seguridad, como por ejemplo de proveer una distancia suficiente a materiales u objetos inflamables, prevenir la ignición , usar dispositivos de protección, prevenir la entrada de objetos externos, usar materiales, componentes y una construcción adecuada para prevenir la ignición o la expansión de fuego
Se buscaron más estándares, y se encontraron los siguientes estándares:

- DO-160G: Características medioambientales, que permite asegurar el funcionamiento del sistema frente a determinados rangos de temperatura, altitud, humedad etc.
- DO-254: Hardware.
- DO-178: Software, que se relaciona con el impacto a la seguridad que se puede generar por una falla de nuestro prototipo. En nuestro caso sería DAL-C.

8. Análisis de impacto de las soluciones y uso de material al día

Los drones han hecho aportes muy valiosos a la sociedad. Un uso reciente fue de realizar búsquedas de personas extraviadas, pudiendo encontrar coordenadas exactas de la persona en específico para su rescate.
También, los drones han sido utilizados para detectar riesgos climáticos, como son tormentas o huracanes.
Otro aporte significativo de los drones ha tenido que ver en el área de la agricultura, los cuales monitorizan constantemente los campos para así ayudar en el riego de los cultivos y ayudar a encontrar si existe alguna plaga o planta que pudiera afectar a los cultivos.
Con respecto a los riesgos, dada la creciente demanda de drones, la congestión área y la posibilidad de choques será cada vez mayor, pudiendo poner en riesgo la salud de personas que se encuentren en la superficie. Otro detalle a considerar, dependiendo del uso que le den, es como los drones podrían alterar la privacidad de personas, dada la alta capacidad y tecnología de procesamiento de imágenes con la que cuentan actualmente.
Por nuestra parte, existen ciertos riesgos al construir el drone que son relativamente obvios. Durante las pruebas de vuelo, existe la posibilidad que por un control que no esté bien probado en la práctica, el drone pueda salir "disparado.ª una pared y romperse varias piezas o peor aún lesionar a alguien. Es por esto que es clave utilizar un sistema de armado/desarmado que sea independiente al control cuyo labor es que en caso de ser necesario apagar completamente el cuadricóptero.

Referencias

[1] http://www.ti.com/lit/ds/symlink/lm2596.pdf

[2] https://www.st.com/resource/en/datasheet/ld1117.pdf

[3] http://www.mantech.co.za/Datasheets/Products/433Mhz_RF-TX&RX.pdf

[4] http://www.irf.com/product-info/datasheets/data/irf540ns.pdf

[5] http://www.imavs.org/papers/2017/321_imav2017_proceedings.pdf

[6] https://etd.auburn.edu/bitstream/handle/10415/3158/David_Wall_Thesis.pdf?sequence=2

[7] https://www.wired.com/2014/05/modeling-the-thrust-from-a-quadcopter/

[8] Douglas, Brian (2018). Matlab Tech Talks.

[9] Selby, Wil (2017). Simulation Environment. https://www.wilselby.com/research/arducopter/simulation environment/

[10] Kotarski.D, Benic, Z & Krznar. M (2016). Control Design For Unmanned Aerial Vehicles With Four Rotors.

[11] Ji. A & Turkoglu. K. (2015). Development of a Low-Cost Experimental Quadcopter Testbed Using an Arduino Controller and Software.

[12] Naduvilakandy. G (2016). Dynamics and Control of a Quadcopter.

[13] Musa, Sumaila. (2018). Techniques for Quadcopter Modelling Design: A Review. 5. 10.21535/just.v5i3.981.