

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE
TELECOMUNICACIÓN



Biblioteca

Diseño de un Dron e implementación de
un sistema auxiliar basado en tecnología
LORA

TRABAJO FIN DE GRADO

Junio – 2021

AUTOR: José Alberto Juan Segura

DIRECTOR: Pablo Corral González

Resumen

Con este proyecto se pretende programar y construir un dron controlado mediante el software de Arduino y además dotarlo de un sistema de emergencia para situaciones extremas.

Dicho sistema de emergencias, cuya finalidad será detener por completo el dron para evitar posibles accidentes por un funcionamiento inadecuado del sistema, se implementará mediante la tecnología Lora la cual se comunicará con arduino.

Nuestro proyecto Dron irá montado sobre un chasis de fibra de carbono, cuatro motores, cuatro ESC (Electronic Speed Controller), una placa distribuidora de potencia, un sensor MPU6050, una batería Lipo, una placa Arduino Nano para controlar todas las ordenes de nuestro dron, dos arduino mini para la recepción del lora, dos Lora, dos antenas, un receptor de radiofrecuencia y un mando a distancia.

El dron una vez montado y configurado la placa Arduino Nano deberá poder ser controlado mediante el mando a distancia. Todas las palancas de control del dron se comunicarán por radiofrecuencia utilizando la banda de 2.4 GHz excepto el del sistema de emergencias que utilizara la tecnología Lora que emplea la banda de 868 MHz. La cámara funcionará a 5 GHz.

Palabras clave: dron, sistema de emergencia, Arduino, Lora, emisor, receptor.

Índice

| | |
|---|----|
| Resumen..... | 3 |
| 1. INTRODUCCIÓN Y OBJETIVOS..... | 9 |
| 1.1 MOTIVACIÓN..... | 9 |
| 1.2 DISEÑO..... | 9 |
| 1.3 ALCANCE..... | 10 |
| 1.4 ESTADO DEL ARTE..... | 11 |
| 1.4.1 CALIBRACIÓN DE LOS PIDS..... | 15 |
| 1.5 OBJETIVOS..... | 17 |
| 2. INTRODUCCION TEORICA..... | 18 |
| 2.1 LA CAPA FÍSICA DE LORA..... | 19 |
| 2.2 PARAMETROS DE LA CAPA FÍSICA DE LORA..... | 20 |
| 2.3 FORMATO DE TRAMA FÍSICA LORA..... | 22 |
| 3. MATERIAL..... | 23 |
| 3.1 LORA..... | 23 |
| 3.2 ARDUINO PRO MINI..... | 24 |
| 3.3 ARDUINO MINI..... | 25 |
| 3.4 ARDUINO NANO..... | 26 |
| 3.5 BUZZER..... | 27 |
| 3.6 CHASIS DRON..... | 28 |
| 3.7 ESC..... | 29 |
| 3.8 BATERIA..... | 30 |
| 3.9 MOTORES..... | 32 |
| 3.10 HÉLICES..... | 33 |
| 3.11 PLACA DISTRIBUIDORA..... | 33 |
| 3.12 CONECTORES XT60..... | 34 |
| 3.13 MPU6050..... | 34 |
| 3.14 EMISOR..... | 36 |
| 3.15 RECEPTOR..... | 37 |
| 4. MÉTODOS..... | 37 |
| 4.1 CIRCUITO ELECTRONICO DEL DRON..... | 37 |
| 4.2 CONEXIÓN ARDUINO-MPU6050..... | 38 |

| | |
|--|----|
| 4.3 CALIBRADO-MPU6050..... | 39 |
| 4.4 CONEXIÓN ARDUINO-ESC-MOTORES..... | 50 |
| 4.5 CONEXIÓN ARDUINO-RECEPTOR | 52 |
| 4.6 Circuito FPV(First Person View)..... | 53 |
| 5. MONTAJE..... | 57 |
| 6. PROGRAMACIÓN..... | 69 |
| 6.1 MAIN PRINCIPAL:..... | 69 |
| 6.2 CALIBRACIÓN MPU6050..... | 76 |
| 7. MODOS DE VUELO..... | 78 |
| 7.1 MODO ACROBÁTICO | 79 |
| 7.2 MODO HORIZON | 83 |
| 8. CONCLUSIÓN | 88 |
| 9. PRESUPUESTO | 89 |
| 10. BIBLIOGRAFIA..... | 90 |



Índice de Figuras

| | |
|--|----|
| Figura 1: Multirotor..... | 11 |
| Figura 2: Clase de multirrotores..... | 12 |
| Figura 3: Helicóptero..... | 13 |
| Figura 4: Ala Fija..... | 13 |
| Figura 5: Actuación de los PIDS..... | 17 |
| Figura 6: Variación de la frecuencia con respecto al tiempo de una señal emitida por LoRa..... | 20 |
| Figura 7: Estructura de una trama de LoRa..... | 22 |
| Figura 8: Modulo RF-LoRa-868-SO..... | 23 |
| Figura 9: Arduino Pro Mini..... | 24 |
| Figura 10: Arduino Mini..... | 25 |
| Figura 11: Arduino Mini..... | 26 |
| Figura 12: Arduino NANO..... | 26 |
| Figura 13: Arduino Nano..... | 27 |
| Figura 14: Buzzer..... | 28 |
| Figura 15: Chasis..... | 28 |
| Figura 16: ESC..... | 29 |
| Figura 17: Batería LiPo..... | 30 |
| Figura 18: Motores..... | 32 |
| Figura 19: Hélices..... | 33 |
| Figura 20: PDB Matek con BEC..... | 34 |
| Figura 21: MPU6050..... | 35 |
| Figura 22: Ejes de movimiento de un dron..... | 36 |
| Figura 23: Emisor..... | 36 |
| Figura 24: Receptor..... | 37 |
| Figura 25: Arduino-MPU6050..... | 38 |
| Figura 26: Orientacion-MPU6050..... | 38 |
| Figura 27: Ejes Tait-Brayan o Euler..... | 39 |
| Figura 28: Calibrado del MPU6050..... | 43 |
| Figura 29: Escalar valores de Acele/Grados..... | 46 |
| Figura 30: grados IMU..... | 47 |
| Figura 31: Sentido de giros de los motores..... | 50 |

| | |
|--|----|
| Figura 32: Conexión Motor-ESC..... | 50 |
| Figura 33: Arduino-ESC-Motores..... | 51 |
| Figura 34: Esquema Angular..... | 52 |
| Figura 35: Arduino-Receptor..... | 52 |
| Figura 37: Gafas FPV..... | 54 |
| Figura 38: Gafas FPV..... | 55 |
| Figura 39: Arduino-Placa Micro perforada..... | 58 |
| Figura 40: Montaje Motor-ESC..... | 58 |
| Figura 41: Posición Arduino NANO..... | 59 |
| Figura 42: Posición MPU6050..... | 59 |
| Figura 43: Soldaduras Arduino NANO-MPU6050..... | 60 |
| Figura 44: Soldaduras Arduino NANO-MPU6050..... | 60 |
| Figura 45: Zócalos Arduino NANO-MPU6050..... | 61 |
| Figura 46: Zócalos Arduino NANO-MPU6050..... | 61 |
| Figura 47: Posición PDB..... | 62 |
| Figura 48: Radiación de un dipolo..... | 63 |
| Figura 49: Antena dipolo receptor..... | 63 |
| Figura 50: Antena dipolo receptor con funda termo retráctil..... | 64 |
| Figura 51: Integración lora..... | 64 |
| Figura 52: Integración lora y conexionado..... | 65 |
| Figura 53: Interruptor dron..... | 65 |
| Figura 54: Conexión interruptor a PDB..... | 66 |
| Figura 55: Esquema RF-LoRa-868-SO..... | 66 |
| Figura 56: Esquema RF-LoRa-868-SO..... | 67 |
| Figura 57: Placa receptora parte delantera..... | 68 |
| Figura 58: Placa receptora parte trasera..... | 68 |
| Figura 59: Diagrama de flujo modo acrobático..... | 80 |
| Figura 60: Generar PID..... | 81 |
| Figura 61: Corrección de error velocidad..... | 82 |
| Figura 62: Ordenes modo acrobático..... | 83 |
| Figura 63: Calculo Err.1..... | 85 |
| Figura 65: Calculo Err.2..... | 85 |
| Figura 66: Ordenes modo estable..... | 86 |

Índice de Tablas

| | |
|---|----|
| Tabla 0: Ventajas y desventajas según el tipo de dron. | 14 |
| Tabla 1: Sensibilidad en dBm del dispositivo LoRa en función del ancho de banda y del factor de propagación. | 21 |
| Tabla 2: Arduino Pro Mini. | 25 |
| Tabla 3: Rango de frecuencias Emisor. | 54 |
| Tabla 4: Rango de frecuencias Receptor. | 55 |
| Tabla 5: Especificaciones Eachine EV800. | 57 |
| Tabla 6: Presupuesto del Proyecto. | 89 |



1. INTRODUCCIÓN Y OBJETIVOS

1.1 MOTIVACIÓN

En la actualidad el uso de drones o vehículo aéreo no tripulado está cogiendo cada día más fuerza, ya se están realizando las primeras pruebas de drones de reparto de mercancía, búsqueda de personas, control de plagas, replanteo de obra, competiciones de drones de carrera o freestyle, etc... en el uso agrícola, donde son utilizados para controlar los cultivos mediante varios sensores que son capaces de detectar la temperatura, la hidratación además de lanzar fertilizantes y pesticidas.

Otro de los usos más comunes es la captura de fotos y videos aéreos. Ya sea con fines de vigilancia como, por ejemplo, los que utiliza la DGT para el control de vías.

Este número elevado de utilidades impresionantes que puede realizar un pequeño aparato manejado a distancia es lo que me causa tanto misterio y provoca que cada día me sumerja más y más en saber cómo podemos fabricar y mejorar uno desde la programación hasta el diseño final.

El desarrollo de este TFG consta del desarrollo software y hardware de un dron. Por un lado, el código software almacenado en Arduino que permita el estudio de estabilidad del dron. Por otro lado, el diseño de una PCB donde se incluyen los dispositivos y conexiones necesarias para el correcto funcionamiento del dron y posibles incorporaciones futuras.

En este proyecto lo dotaremos de un sistema de emergencia que será un extra debido a la peligrosidad que supondría que un dron perdiera el control en una zona transitada por personas.

1.2 DISEÑO

Según el tipo de Dron que quieras diseñar tendrás que tener en cuenta varios factores, en nuestro caso vamos a realizar un mix entre dron de competición y dron de fotos y videos más orientados a la actualidad. El requisito más importante a tener en cuenta a la hora de diseñar un dron es el tiempo de vuelo. Para calcular este tiempo deberemos tener en cuenta dos cosas la potencia de los motores y el peso total de nuestro dron.

Respecto a la potencia de vuelo tenemos dos variables muy importantes que son los motores y el tipo de hélice, debemos tener en cuenta que para hacer despegar nuestro dron aparte de tener un motor potente debemos escoger bien la pala, ya que un mismo motor puede ejercer diferente presión hacia el suelo dependiendo la pala que estemos utilizando.

Respecto al peso total del dron donde tenemos que hacer más hincapié es en la batería ya que esta suele ser el 40% del peso total del dron, eligiendo una buena batería adecuada a nuestras necesidades ahorraremos en tiempo.

Una vez valorado estas premisas, vamos a diseñar un dron que tendrá el tiempo de vuelo cercano a los 10-15 minutos, que es el tiempo que nos suelen vender.

1.3 ALCANCE

El objetivo del proyecto es construir un dron y desarrollar un software que se ejecute en él. El software consiste en el manejo de la aeronave no tripulada de forma que pueda corregir errores y mantenerse horizontal en todo momento a parte de un sistema de emergencia controlado de forma inalámbrica.

Este proyecto debe servir de guía para que cualquier usuario con un conocimiento base de estudios pueda realizar el montaje de una aeronave no tripulada, añadirle mejoras y funcionalidades.

El sistema que controla el dron utilizará la radiofrecuencia en la banda de 2.4 GHz para comunicarse con el mando a distancia y el sistema de emergencias trabajará en la banda de 868 MHz para poder tener los dos sistemas independientes sin que surja ningún tipo de interferencia.

1.4 ESTADO DEL ARTE

En este apartado se presentan los modelos más importantes que existen en el campo de los drones, se enumeran sus características y se hace una pre-selección de los componentes a utilizar.

Encontrar una plataforma de vuelo óptima para trabajar, en ocasiones puede ser un quebradero de cabeza. En este punto vamos a tratar de diferenciar cada uno de ellos.

Nos encontramos básicamente con dos tipos de plataformas: Ala fija y Ala rotatoria. El ala fija tiene el aspecto y forma de un avión de aerodelismo de toda la vida y el ala rotatoria tiene dos tipos de drones incluidos, los helicópteros y los multirrotores.

1. Multirrotores:



Figura 1: Multirrotor.

Los multirrotores son la herramienta más extendida actualmente y la que todos al pensar en drones tenemos en nuestra mente. Proporciona una gran versatilidad y eficacia en las operaciones por su simpleza a la hora de ser pilotados y por la velocidad de montaje. Es una plataforma estable por naturaleza, debido a que los motores se encuentran a la misma distancia del centro de gravedad de la aeronave.

Según la cantidad de motores los clasificamos en tricópteros (3 motores), **cuadricópteros** (4 motores), **hexacópteros** (6 motores) y **octocópteros** (8 motores). Y según la configuración de los brazos los clasificamos en “Y” (i griega), “Y invertida” (i griega invertida), “X” (equis), “+” (cruz).

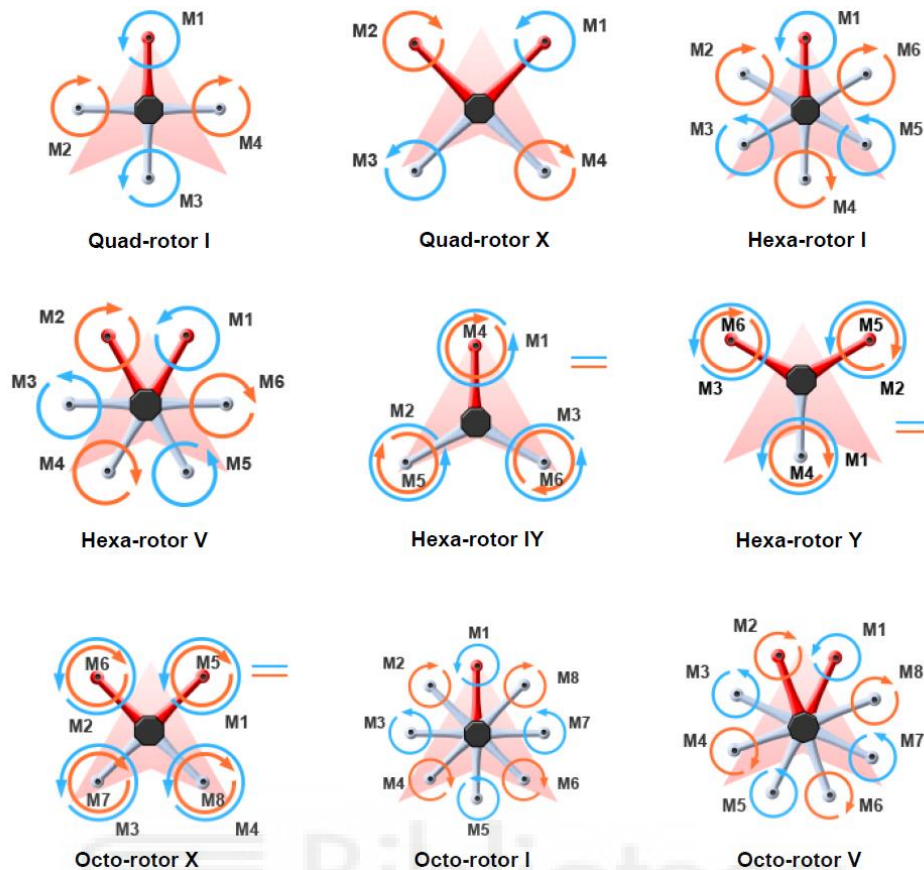


Figura 2: Clase de multirrotores.

Es importante destacar que pese a las muchas ventajas que ofrece un multirroto, nos encontramos con una desventaja nada despreciable, la autonomía. Un multirroto, de media no suele superar los 15 minutos de vuelo, lo que representa un gran impedimento para muchas operaciones.

2. Helicópteros:

Los helicópteros son la herramienta más polivalente a la hora de realizar todo tipo de operaciones. Poseen una gran capacidad de carga y autonomía. Esto es gracias a que sólo posee un motor y una hélice de gran tamaño. Si lo comparamos con un cuadricóptero, estamos reduciendo a $\frac{1}{4}$ el consumo de energía. El helicóptero es mucho más eficiente aerodinámicamente que un multirroto, ya que el helicóptero funciona a revoluciones fijas de motor gracias al paso variable de las hélices, mientras que el multirroto varía las revoluciones del motor para mantenerse estable. Otra ventaja muy destacable es que si lo

dotamos de un motor de explosión, podemos permanecer en el aire alrededor de 1 hora, lo que es perfecto para operaciones como la fotogrametría.

Sin embargo, los helicópteros son bastante complejos a nivel mecánico, lo que nos obliga a tener que estar constantemente ajustándolo para que nos ofrezca un vuelo óptimo. También es bastante complicado a la hora de ser pilotado, y dominarlos suele llevar bastantes años de práctica.

Actualmente, el mercado nos ofrece soluciones al tema del control del helicóptero. Esta solución pasa por la controladora. Existen algunos tipos de controladoras específicas para helicópteros, lo que nos ofrecería un vuelo bastante semejante a un multirroto. El inconveniente es que éstas suelen ser bastante caras y en ocasiones difíciles de configurar.



Figura 3: Helicóptero.

3. Ala Fija:



Figura 4: Ala Fija.

El ala fija es el claro ganador en lo que a autonomía se refiere. Según esté equipado con motor eléctrico o de explosión, puede permanecer en el aire varias horas. Es la plataforma perfecta para trabajos que abarquen una gran extensión de terreno. Por otra parte, es el más eficiente aerodinámicamente hablando, ya que, con la configuración adecuada, puede permanecer bastante tiempo sin necesidad de utilizar el motor gracias al planeo. Por otra parte, el hecho de poder planear hace que sea una plataforma mucho más segura, ya que en un supuesto fallo de motor puede planear hasta llegar al punto de aterrizaje.

Sin embargo, el ala fija, está preparado para unos fines muy específicos, lo que le resta versatilidad a la hora de ser utilizado. Su principal desventaja es el tema del aterrizaje y el despegue. El no poder aterrizar y despegar verticalmente nos obliga a tener que acotar una extensión bastante grande de terreno (unos 60m), y que ésta sea plana y sin obstáculos. Por otra parte, un drone de ala fija no permite hacer un vuelo estacionario, lo que nos impide poder realizar un sinnúmero de operaciones. También posee una reducida capacidad de carga de peso respecto a su tamaño.

Por tanto, el ala fija es una herramienta indispensable para algunas operaciones concretas, pero no es la herramienta ideal si lo que buscamos es versatilidad. A modo de conclusión hay que decir que cada aeronave está destinada a un fin concreto y unas son más polivalentes que otras. En la siguiente tabla veremos qué plataforma es la más idónea para cada tipo de operación:




























| |  |  |  |
|--------------------------|---|--|---|
| Fotogrametría |  |  |  |
| Visual |  |  |  |
| Cartografía |  |  |  |
| Vigilancia/Seguridad |  |  |  |
| Inspección |  |  |  |
| Agricultura de precisión |  |  |  |
| Cobertura |  |  |  |
| Tiempo de vuelo |  |  |  |

Tabla 0: Ventajas y desventajas según el tipo de dron.

Para nuestro proyecto hemos elegido la opción de multirrotor porque pueden despegar y aterrizar de manera vertical, asimismo es una ventaja ya que no necesitamos de grandes áreas para poner en vuelo a nuestro dron. Por otra parte, gracias a sus potentes motores son capaces de detenerse en el aire por completo manteniendo unas coordenadas fijas, esto le permite hacer grandes maniobras en espacios realmente pequeños siendo muy versátil para trabajos de rescate, inspecciones, modelamiento y mapeo de lugares. Estas son las razones por las que he decidido construir el dron.

1.4.1 CALIBRACIÓN DE LOS PIDS

En este proyecto tiene como objeto la calibración de los PIDS tanto en la forma de vuelo acrobática como horizon todo esto respaldado por la tecnología LoRa, haremos múltiples pruebas hasta que consigamos una buena respuesta de estabilización.

Pero que son los PIDS y por qué son tan importantes en este proyecto. Las siglas PID son las siglas para Proporcional Integral Derivativo. Estos tres parámetros son los utilizados en el algoritmo de cálculo para la estabilización del dron.

Es un mecanismo de control que calcula la desviación o error entre los valores obtenidos por nuestro sensor y los valores que realmente debería tener el sensor en esa posición para estar en la posición correcta.

Proporcional

La función de P es el encargado de la estabilidad y control es el valor más importante e indica el nivel de corrección necesario a aplicar. Mientras más grande sea el valor de P, más intentara estabilizarse, pero si nos excedemos se volverá demasiado sensible y provocara oscilaciones.

Si encontramos el valor correcto para las características físicas de nuestro dron los valores de I y D los podríamos dejar a 0 y el dron haría hover sin problemas.

- Si el valor de P es muy bajo será muy difícil controlar el dron porque será muy fácil sobre-correr las maniobras cosa que hará imposible mantenerlo estable.

- Si el valor de P es correcto nos será fácil mantener la estabilidad y acelerará correctamente al aplicar o dejar gas.
- Si el valor de P es demasiado alto el dron oscilará rápidamente o los motores emitirán un sonido oscilante y agudo. También ganará altura fácilmente (a veces como a saltos) y será difícil mantenerla.

Video demostrativo constante $kP= 1,8$:

https://drive.google.com/file/d/1giBisinIHQRud9_cxp5krgQugsFpmFet/view?usp=sharing

Video demostrativo constante $kP= 2,8$:

<https://drive.google.com/file/d/1D3syYUOORBNkHjiSsXQOkFp-6WeUAQWe/view?usp=sharing>

Video demostrativo constante $kP= 4,8$:

<https://drive.google.com/file/d/1KKda6s0uhukLrj8ywx3bnfrmEBOgpxDL/view?usp=sharing>

Integral

La función de I es la encargada de la velocidad con la que se repite la acción Proporcional, si el valor de $I=0$ notaremos que el movimiento de oscilación es muy seco como si fuese un robot.

- Si el valor de I es muy bajo el dron tenderá a subir el morro al cambiar de dirección y derivará.
- Si el valor de I es correcto mantendrá el ángulo de forma mucho más precisa.
- Si el valor de I es demasiado alto el dron oscilará lentamente y tenderá a bajar el morro cuando cambiamos de dirección. También se volverán muy perezosas las reacciones.

Video demostrativo constante $kI= 0,5$:

<https://drive.google.com/file/d/1AEhROoffMF33NKYEtBpbbTEFNI2wyUK2/view?usp=sharing>

Video demostrativo constante $kI= 0,2$:

<https://drive.google.com/file/d/1S847KnMLp6rmulrawKFdDTWwyzXtUMhR/view?usp=sharing>

Derivativo

La función de D es suavizar todos los movimientos.

- Un valor de D muy bajo hará el dron perezoso y en movimiento puede asemejarse a tener la P muy baja.
- Un valor de D bajo suavizará las reacciones
- Un valor de D más elevado hará que las reacciones sean más nerviosas.
- Un valor de D demasiado elevado provocará oscilaciones rápidas.

Video demostrativo constante $k_D= 0,3$:

https://drive.google.com/file/d/1m_ZWh0Z_eb8NskuEaZ5aD1DpRgH_sW9j/view?usp=sharing

Video demostrativo constante $k_D= 0,8$:

<https://drive.google.com/file/d/1NRvwYfrRUNehKFL0EXXnY4OxaJhY74aH/view?usp=sharing>

Video demostrativo constante $k_P= 0$, $k_I=0$, $k_D=0$: <https://drive.google.com/file/d/12wac6jx6-HnLFrRBwvs-NI2FpVUVzuTe/view?usp=sharing>

A continuación, vemos una representación gráfica de la actuación de los PIDS

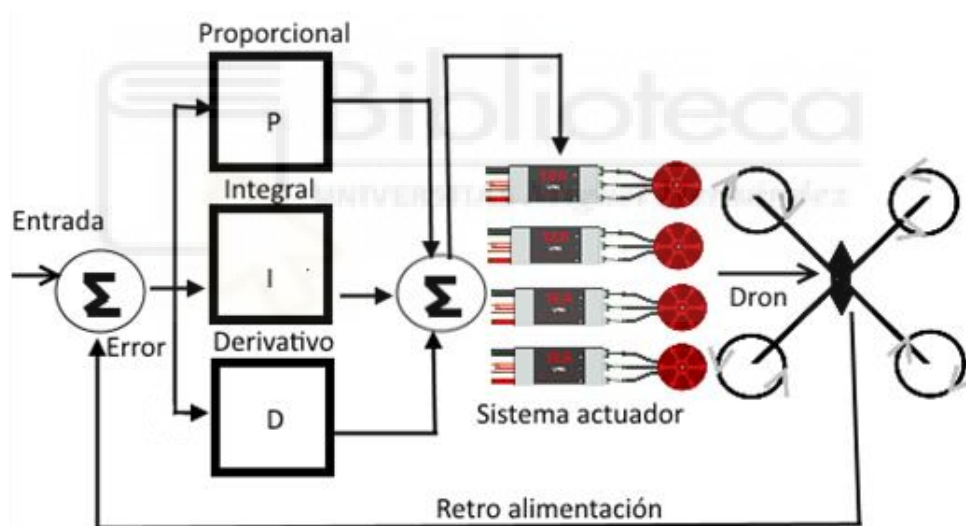


Figura 5: Actuación de los PIDS.

1.5 OBJETIVOS

Los objetivos que se pretenden conseguir con el proyecto son los siguientes:

- Realizar el diseño de una aeronave no tripulada de forma sencilla, destacando los aspectos más importantes a tener en cuenta a la hora de elegir los componentes para su diseño.
- El proyecto se va a realizar dejando abierta la integración de distintas funcionalidades en un futuro, por lo que el desarrollo no se va a encasillar en una aplicación concreta. En el

desarrollo de la PCB, en el que se incluye toda la electrónica, se ha previsto la inclusión en el futuro de dos elementos más para ampliar las capacidades del dispositivo, un sensor de presión para controlar la altura y un receptor de video.

- Implementación de un sistema de emergencia independiente en el Dron mediante tecnología LoRa, detallando la configuración tanto en el emisor como en el receptor.

2. INTRODUCCION TEORICA

LoRa (Long Range/largo alcance), es una tecnología inalámbrica de largo alcance, desarrollada por Semtech pero que actualmente está administrada por la LoRa Alliance. Utilizada en comunicaciones militares desde hace décadas

LoRa es el tipo de modulación en radiofrecuencia patentado que entre sus principales ventajas se encuentra:

- Alta tolerancia a las interferencias
- Alta sensibilidad para recibir datos (-168 dB)
- Basado en modulación chirp
- Bajo Consumo (hasta 10 años con una batería*)
- Largo alcance 10 a 20 km
- Baja transferencia de datos (hasta 255 bytes)
- Conexión punto a punto
- Frecuencias de trabajo: 915 MHz América, 868 MHz Europa, 433 MHz Asia

La capa física de LoRa utiliza el espectro sin licencia como parte de las bandas ISM (Industrial, Scientific and Medical) de 433 MHz, 868 MHz o 915 MHz, dependiendo de en qué zona nos encontremos.

La carga útil de cada transmisión fluctúa ente los 2-255 octetos, y la tasa de bits puede alcanzar hasta los 50 kbps.

2.1 LA CAPA FÍSICA DE LORA

LoRa opera en la banda ISM por debajo de 1 GHz, utiliza como esquema de modulación una variación de la modulación DSS de tipo Spread Spectrum (espectro ensanchado) denominada “Chirp Spread Spectrum” (CSS) en la que se utilizan los pulsos chirp con una variación lineal de la frecuencia en el tiempo para codificar la información. La linealidad de los pulsos chirp provoca que la desincronización de la frecuencia entre el receptor y el emisor sea equivalente a la del tiempo, permitiendo que se elimine fácilmente en el decodificador. Gracias a esto la modulación “Chirp Spread Spectrum” es resistente al efecto Doppler.

La desincronización en frecuencia entre el emisor y el receptor puede llegar a ser un 20% del ancho de banda sin tener consecuencias en el rendimiento del decodificador. Esto es una de las claves para reducir el precio de los emisores de LoRa, puesto que los cristales embebidos en los transmisores no necesitan ser fabricados con una precisión extrema. Los receptores LoRa son capaces de acoplarse a los pulsos chirp, permitiendo una sensibilidad de aproximadamente -137 dBm para un Spreading factor de 12 y -123 dBm para un Spreading factor de 7.

LoRa utiliza un método de ajuste dinámico de la potencia de emisión y la tasa de transferencia denominado ADR (Adaptative Data Rate) que principalmente permite al dispositivo receptor el ajuste dinámico de los parámetros antes citados en función de la distancia que separa a los dispositivos y del tamaño del mensaje para conseguir que la comunicación sea a la máxima velocidad posible con el menor gasto de energía.

En LoRa la duración de los símbolos es mayor que las típicas interferencias de AM que se producen en sistemas Frequency Hopping Spread Spectrum (FHSS), los errores producidos por estas interferencias son corregidos en LoRa mediante Forward Error Correction Codes (FECs) permitiendo la corrección de los errores en el receptor sin tener que realizar retransmisiones de la información original lo cual es de ayuda para que sea una tecnología de bajo consumo.

La típica selectividad entre otros canales es decir la máxima diferencia de potencia entre una interferencia en una banda vecina y la señal LoRa es de 90 dB y el rechazo en el mismo

canal (la máxima diferencia de potencia entre una interferencia en el mismo canal y la señal de LoRa) de los receptores de LoRa es 20 dB.

2.2 PARAMETROS DE LA CAPA FÍSICA DE LORA

LoRa nos permite variar varios parámetros en la modulación según las necesidades de la aplicación para la que se aplique: ancho de banda (Bandwidth, BW), factor de propagación (Spreading factor, SF) y tasa de código (Code Rate, CR). LoRa utiliza una definición del factor de propagación poco común, esto es, el logaritmo en base 2 del número de chirps por símbolo.

Estos parámetros influyen en la tasa de bits efectiva de la modulación, la resistencia a las interferencias y la facilidad de decodificación. Un símbolo de LoRa está formado por 2^{SF} chirps, que ocupan por completo la banda de frecuencia. Comienza en la mínima frecuencia de la banda enviando chirps de manera ascendente hasta que se alcanza la máxima frecuencia de la banda entonces vuelve al valor mínimo de la frecuencia y empieza a repetir el proceso anterior. Esto se aprecia en la Figura 2. La posición de esta discontinuidad en frecuencia es lo que codifica la información transmitida. La cantidad de bits que se pueden codificar por símbolo viene dada por el Spreading Factor (SF) que puede tener un valor de entre 7 y 12 incluyendo ambos, por lo cual, si este tiene un valor de N, el símbolo representa N bits y puede tener 2 elevado a la N posibles valores de frecuencia a los que puede ir saltando.

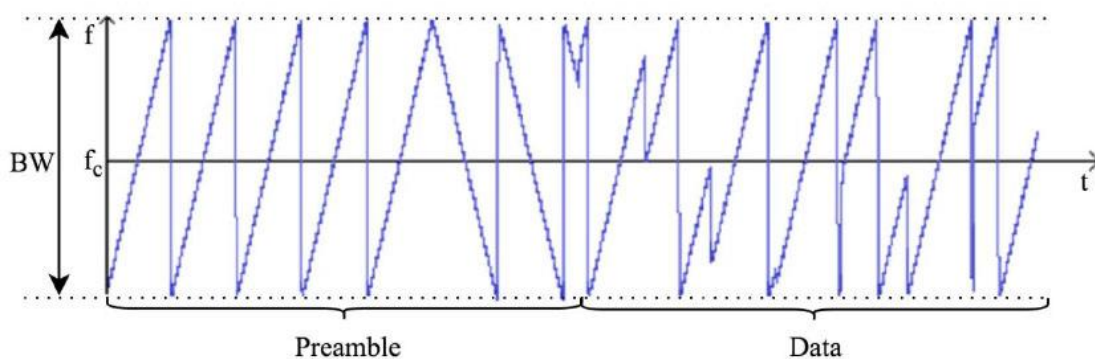


Figura 6: Variación de la frecuencia con respecto al tiempo de una señal emitida por LoRa.

El ancho de banda es uno de los parámetros más importantes en la modulación LoRa ya que la tasa de chirp es igual al ancho de banda (un chirp por segundo por Hertzio del ancho de

banda). Este hecho provoca consecuencias en la modulación: si incrementamos el factor de propagación en uno implicará la división del intervalo de un chirp en dos y multiplicará la duración del símbolo por dos. Sin embargo, no dividirá la tasa de bits por dos porque como he comentado anteriormente en un símbolo se codifican SF bits de información por lo que la tasa de bits aumentará en uno puesto que se envía un bit más por símbolo.

Además, la tasa de símbolos y la tasa de bits para un factor de propagación dado será proporcional al ancho de banda de la frecuencia, por lo que doblar el ancho de banda doblará la tasa de transmisión ya que aumentaríamos el número de frecuencias para saltar. Esto se puede observar en la siguiente ecuación (1).

$$T_s = 2^{SF}/B \quad (1)$$

LoRa también incluye un forward error correction code. La tasa de código CR es igual a $4/(4+n)$, con $n \in \{1, 2, 3, 4\}$. Teniendo esto en cuenta, así como el hecho de que SF bits de información son transmitidos por símbolo, la ecuación (2) permite calcular la tasa de bits (R_b).

$$R_b = SF \times B / 2^{SF} \times CR \quad (2)$$

Todos estos parámetros influyen en la sensibilidad del decodificador. En concreto un incremento del ancho de banda reduce la sensibilidad del receptor y aumenta la velocidad de la comunicación, mientras que un incremento del factor de propagación incrementa la sensibilidad del receptor y ralentiza la comunicación.

| BW/SF | 7 | 8 | 9 | 10 | 11 | 12 |
|---------|------|------|------|------|------|------|
| 125 KHz | -123 | -126 | -129 | -132 | -133 | -136 |
| 250 KHz | -120 | -123 | -125 | -128 | -130 | -133 |
| 500 KHz | -116 | -119 | -122 | -125 | -128 | -130 |

Tabla 1: Sensibilidad en dBm del dispositivo LoRa en función del ancho de banda y del factor de propagación.

2.3 FORMATO DE TRAMA FÍSICA LORA

La modulación LoRa puede emplearse para transmitir tramas arbitrarias, pero existe un formato de trama física especificado e implementado en los dispositivos Semtech. El ancho de banda y el factor de propagación son constantes para una misma trama.

La trama LoRa comienza con un preámbulo de sincronización en el que se define el esquema de modulación. Esto permite definir los parámetros de modulación de forma sencilla en función de nuestras necesidades además de codificar en los dos últimos chirps del preámbulo la palabra de sincronización o “sync word”.

Esta palabra de sincronización de un byte es un valor de un byte que se usa para diferenciar redes LoRa que emplean las mismas bandas de frecuencia. Un dispositivo configurado con una determinada palabra de sincronización dejará de escuchar las transmisiones cuyas palabras de sincronización decodificadas sean distintas a la suya. La palabra de sincronización va seguida de dos chirps y medio descendentes, que duran 2.5 símbolos. Este preámbulo puede tener como duración entre 10.25 y 65539.25 símbolos.

Después del campo preámbulo, hay una cabecera opcional que en caso de transmitirse se hará con un CR de 4/8. Esta indica el CR empleado en el final de la transmisión, el tamaño de la carga útil en bytes y si se ha utilizado o no un CRC (Cyclic Redundancy Check) al final de la trama. El tamaño de la carga útil se almacena usando un byte, siendo el máximo tamaño de la carga útil a 255 bytes. La cabecera también incluye un CRC para permitir al receptor descartar paquetes con cabeceras no válidas.

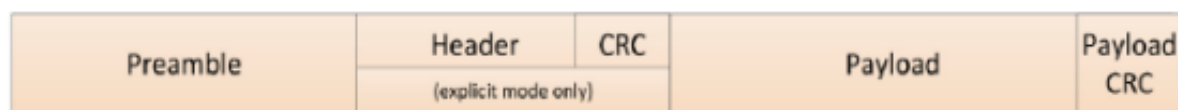


Figura 7: Estructura de una trama de LoRa.

Cada parte de la trama tiene su propia función

- **Preámbulo**, esta parte de la trama tiene como función principal la de sincronismo en el esquema de modulación de la trama o mensaje que se envía.

- **Cabecera**, este es un elemento opcional a configurar, su función es la de añadir más información sobre la trama y control de errores.

- **Carga Útil**, es el elemento donde se encuentra los datos proporcionados por los sensores y que se pretenden enviar hacia el servidor.

3. MATERIAL

3.1 LORA

Se trata del componente más importante del hardware necesario para realizar el sistema de emergencia. Se han utilizado dos dispositivos RF-LoRa-868-SO de la empresa británica RF SOLUTIONS Ltd. Estos dispositivos implementan el chip SX1272 desarrollado por Semtech, conectado de tal forma que se simplifican las conexiones a la hora de acceder a los pines claves del chip SX1272.



Figura 8: Modulo RF-LoRa-868-SO.

Las características principales que tiene este modelo de LoRa son:

- Integra un módulo SX1272 Semtech.
- Alta sensibilidad, alrededor de -130 dBm.
- Potencia configurable, hasta 20 dBm.
- Tensión de alimentación entre 1.8 y 3.6 V.
- Consumo de corriente entre 10 y 125 mA, en función del modo de operación y de la potencia configurada.

- Posibilidad de conexión de antena de forma directa con el dispositivo
- gracias a la simplificación del patillaje.
- Tasa de bits programable hasta los 300 kbps.
- Posibilidad de medir de forma directa SNR y RSSI.
- Puede comportarse como receptor y como emisor utilizando el mismo sistema de conexiones.
- Las funcionalidades que tiene este modelo de LoRa dentro del proyecto son:
- Responsable del envío y la recepción de mensajes para la comunicación inalámbrica.
- Configurar la potencia de envío, para adaptarse al escenario.
- Configurar los parámetros de ancho de banda, factor de propagación, tasa de código, etc., para alcanzar el máximo alcance posible.

3.2 ARDUINO PRO MINI

Es el microcontrolador que controlará y se comunicará con el módulo LoRa así podremos configurar los pulsadores para activar nuestro sistema de emergencia. Los principales motivos por los que se ha escogido este dispositivo es la facilidad de uso, la disponibilidad de código y librerías que implementaban el uso del chip SX1272, su pequeño tamaño ideal para colocarlo en el mando y que tan solo necesitábamos un modelo que nos suministrara 3.3 V para alimentar el módulo LoRa.

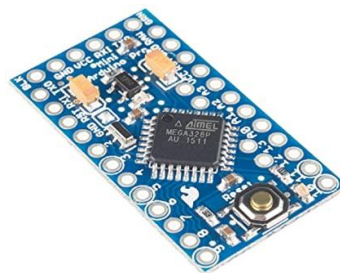


Figura 9: Arduino Pro Mini.

| | |
|--|--|
| Microcontrolador | ATmega328P * |
| Fuente de alimentación de la placa | 3,35-12 V |
| Voltaje de funcionamiento del circuito | 3,3 V |
| Pines de E / S digitales | 14 |
| Pines PWM | 6 |
| UART | 1 |
| SPI | 1 |
| I2C | 1 |
| Pines de entrada analógica | 6 |
| Interrupciones externas | 2 |
| Corriente CC por pin de E / S | 40 mA |
| Memoria flash | 32 KB de los cuales 2 KB utiliza el gestor de arranque * |
| SRAM | 2 KB * |
| EEPROM | 1 KB * |
| Velocidad de reloj | 8 MHz |

Tabla 2: Arduino Pro Mini.

3.3 ARDUINO MINI

Es el microcontrolador responsable de comunicarse con el módulo LoRa utilizado como receptor colocado en el dron. Hemos escogido principalmente este Arduino por que funciona como toda la electrónica que tenemos en el dron y así podemos alimentar con 5 V al relé para que se accione ya que si utilizáramos el PRO MINI (emisor) con 3.3 V no conseguimos generar el campo magnético suficiente para que se activen las bobinas del relé y conmute. A parte de su pequeño tamaño y facilidad de implementarlo en cualquier sitio del dron.



Figura 10: Arduino Mini.

| | |
|--|--|
| Microcontrolador | ATmega328 |
| Fuente de alimentación de la placa | 5-12 V |
| Voltaje de funcionamiento del circuito | 5 V |
| Pines de E / S digitales | 14 |
| Pines PWM | 6 |
| UART | 1 |
| SPI | 1 |
| I2C | 1 |
| Pines de entrada analógica | 8 |
| Interrupciones externas | 2 |
| Corriente CC por pin de E / S | 40 mA |
| Memoria flash | 32 KB de los cuales 2 KB utiliza el gestor de arranque * |
| SRAM | 2 KB * |
| EEPROM | 1 KB * |
| Velocidad de reloj | 16 MHz |

Figura 11: Arduino Mini.

3.4 ARDUINO NANO

Como microcontrolador principal para coordinar con los distintos módulos que integran el dron hemos utilizado el modelo NANO de la gama de Arduino. Entre los diferentes motivos de esta elección destaco el pequeño tamaño de esta placa y el gran número de pines digitales.

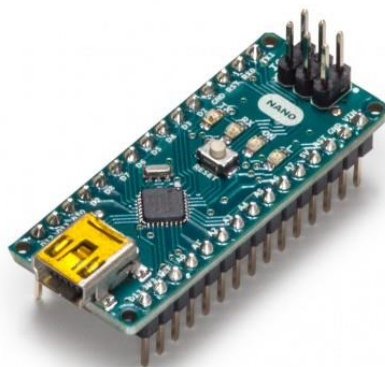


Figura 12: Arduino NANO.

| | |
|----------------------------------|--|
| Microcontrolador | ATmega328 |
| Tensión de funcionamiento | 5V |
| Voltaje de entrada (recomendado) | 7-12V |
| Pines de E / S digitales | 22(6 de los cuales son PWM) |
| Canales de entrada analógica | 8 |
| Corriente CC por pin de E / S | 40 mA |
| Memoria flash | 32 KB de los cuales 2 KB utiliza el gestor de arranque |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Velocidad de reloj | 16 MHz |
| Longitud | 45 mm |
| Anchura | 18 mm |
| Peso | 7 g |

Figura 13: Arduino Nano.

Arduino NANO se encargará de gestionar los módulos como el MPU6050, Magnetómetro, Receptor de mando, ESC(Electronic Speed Controller), las señales de emergencia que mandaremos desde nuestro mando que las recibirá Arduino Mini estas señales de emergencia son el descenso progresivo y el corte de pánico el cual dejara sin alimentación a los motores.

3.5 BUZZER

Buzzer (zumbador) es un transductor electroacústico que produce un zumbido continuo o intermitente.



Figura 14: Buzzer.

Este irá en el dron de forma que emita sonidos cuando se accione el sistema de emergencia con el objetivo de que, si el dron cae en una zona de difícil visión o simplemente no tenemos una buena referencia de donde cayó lo podamos localizar mediante el sonido.

3.6 CHASIS DRON

El chasis del dron es un elemento muy importante a la hora de diseñar un dron ya que tenemos de varias formas y tipos y según sea tendrán unas peculiaridades



Figura 15: Chasis.

El diseño de nuestro dron se asemeja a un dron de carrera o freestyle, no es nuestra finalidad, pero si que buscamos un dron que sea ligero, rápido y maniobrable para tener unas buenas tomas de fotografía y video.

El chasis del dron está fabricado en fibra de carbono lo cual lo hace muy ligero y resistente, combinado con unos pilares de aluminio los cuales nos brindara un refuerzo y firmeza a grandes velocidades sin que este se deforme, a parte hemos diseñado unas piezas en 3D para distribuir mejor nuestra electrónica.

3.7 ESC

Los ESC o Electronic Speed Controller, nos encontramos ante un controlador de velocidad electrónico. Un sistema capaz de definir la velocidad de giro de un motor brushless mediante la generación de pulsos PWM de control que apliquemos, estos son necesarios ya que los motores de los drones son trifásicos de corriente alterna, los también llamados variadores convierten la tensión en continua DC en tensión alterna AC.



Figura 16: ESC

Dos cables: generalmente uno rojo y otro negro se corresponden a la entrada de alimentación de la batería.

Tres cables: suelen ser los 3 negros se conectan al motor para proveerlo de alimentación mediante pulsos.

Un conector con dos o tres cables: va conectado a la controladora, de la que recibe los datos para mover el motor. El número de cables dependerá de si el ESC lleva incluido BEC (3 cables) o no (2 cables).

¿Qué tipo de ESC tengo que elegir para mi dron? para cada tipo de dron en general dependerá del tipo de empuje que necesitemos en nuestro dron, un factor que está muy relacionado con el tipo de batería utilizado y su configuración de motores y hélices.

Si utilizamos una batería 4S, es conveniente utilizar un ESC de al menos 20A.

Características:

- Corriente : 36A
- Corriente de explosión (10S): 42A
- Celdas LiPo: 3-5S
- Peso: 8.4g
- Tamaño: 35x11x7mm.
- Aplicaciones típicas: 170-330 mm multirrotores.

3.8 BATERIA



Figura 17: Batería LiPo.

El encargado de suministrar energía eléctrica a todo el Dron. Las más utilizadas en drones son las LiPo ya que son capaces de acumular bastante energía siendo ligeras. En nuestro caso utilizaremos una LiPo de 4 celdas y 1550 mAh capaz de proporcionar 14.8 V.

Las baterías LiPo usada en RC están compuestas de **celdas individuales** conectadas en serie. Cada celda tiene un **voltaje nominal de 3.7 V**, por lo tanto el voltaje se define simplemente como la cantidad de celdas de la batería, también conocido como su número "S". así :

1S = 1 celda = 3.7V

2S = 2 celdas = 7.4V

3S = 3 celdas = 11.1V

4S = 4 celdas = 14.8V

5S = 5 celdas = 18.5V

6S = 6 celdas = 22.2V

El **voltaje** afecta directamente la **velocidad de los motores sin escobillas**, así que usando baterías con más S puedes aumentar la velocidad del dron si es que los ESC y el resto de la electrónica soporta este voltaje más alto.

La **capacidad de una batería LiPo** se mide en **mAh (mili amperios hora)**. mAh es básicamente una indicación de cuanta corriente puedes extraer de la batería en una hora hasta que está vacía.

Por ejemplo, una LiPo de 1550 mAh se descargaría completamente en una hora si se extraen 1,55 A de ella. Si la corriente extraída se doblase (3.1 A), el tiempo se reduciría a la mitad. Si se sacasen 36 A de corriente de forma continua, la batería solo duraría dos minutos y medio ($1,55A/36A=0.043$ horas) $\rightarrow 0.043h*60Min=$ **2,58 Min**

Incrementar **el tamaño de la batería** podría incrementar el tiempo de vuelo, pero también aumenta el tamaño y peso de la misma, que a su vez disminuye el tiempo de vuelo.

Hay por tanto un compromiso por tanto entre la capacidad y el peso para definir cual es el tamaño óptimo de la batería que afecta a la agilidad del vuelo.

C Rating es un indicador de la velocidad de descarga continua de un LiPo. Está diseñado para permitir a los usuarios calcular fácilmente la corriente constante máxima que puede dar una batería LiPo de forma segura sin dañarse.

Consumo máximo de corriente = Capacidad x Clasificación C

Por ejemplo, en nuestro caso tenemos una batería LiPo 4S 1550mah 100C, su consumo de corriente máximo seguro sería de $1550ma \times 100C = 155A$.

Características:

- Capacidad mínima: 1550mAh
- Configuración: 4S1P / 14.8V / 4 Cells
- Velocidad de descarga: 100C
- Max Burst tasa de descarga: 200C
- Peso neto (± 20 g): 178 g / unidad
- Dimensiones: 72 mm Longitud 35 mm x anchura x 36 mm Altura
- Enchufe de carga: JST-XHR
- Enchufe de descarga: XT-60

3.9 MOTORES

Los motores y palas que se vayan a utilizar dependen del peso que se desee levantar, es decir, del peso del conjunto-dron. El empuje necesario se consigue con una correcta combinación motor-pala. Los fabricantes de motores especifican las palas adecuadas para conseguir potencias determinadas, lo cual simplifica la elección: se elige un motor de suficiente potencia, y se selecciona la pala que, según el fabricante, da el empuje necesario.



Figura 18: Motores.

Dependiendo del peso del dron, se utilizan motores de continua (menor potencia), o motores trifásicos con variadores de frecuencia, que transforman la corriente continua de la batería en una señal trifásica. Como el dron que se pretende construir en este proyecto es

relativamente grande, se utilizarán motores trifásicos sin escobillas, más conocidos como motores “brushless”, los cuales generan más potencia que los motores de corriente continua.

3.10 HÉLICES

En cuanto a las palas, lo único que hay que considerar es que, a mayor superficie, mayor empuje y menor velocidad de giro. Los motores brushless tienen un mayor rendimiento a revoluciones bajas, por lo que la mejor elección será unas palas del mayor radio posible.

Para aumentar aún la superficie de sustentación, pueden utilizarse hélices tripala en lugar de las bipala (ver Figura12).



Figura 19: Hélices.

3.11 PLACA DISTRIBUIDORA

Una PDB (Power Distribution Board) es un elemento esencial en el mundo de los drones.

Facilita el montaje del dron y la ampliación del mismo con nuevos componentes. Básicamente es un circuito impreso que permite realizar la soldadura de todos los componentes con relativa facilidad.

No es un elemento obligatorio, podemos prescindir de usar una PDB, pero el manejo de cables que resultaría del experimento seguramente daría bastante grima. Además de complicar cualquier reparación o modificación sobre el dron.

El excesivo voltaje de una batería 4S hará que necesitemos un BEC que se encargue de brindarnos dos tipos de voltaje distintos, compatibles con el resto de los componentes. Los voltajes habituales que nos proporciona este tipo de PDB son 5 V (la tensión necesaria para alimentar la controladora), y 12v (para LEDs, ESC y sistema de vídeo).



Figura 20: PDB Matek con BEC.

También encontraremos PDBs que disponen de un filtro LC, este filtro compuesto esencialmente por ferrita y condensadores, nos da una línea de tensión desparasitada de interferencias electromagnéticas. Conectar el sistema de vídeo a una línea limpia es garantía de buena calidad de imagen.

3.12 CONECTORES XT60

Conectores XT60 para baterías LiPo. Hechos de nylon de gran calidad y conectores chapados en oro 4.5mm, proporcionan conexiones sólidas para amperajes altos (+100A).

3.13 MPU6050

El acelerómetro es un sensor que va a medir la aceleración estática, y también la aceleración dinámica. La primera en el eje vertical, como la gravedad, y la segunda, en el eje horizontal, en el plano XY. Es utilizado para determinar la posición y la orientación del dron durante el

vuelo. Por otro lado, está el giroscopio, encargado de medir los ángulos de ubicación del dron cuando este se encuentra en el aire. Generalmente este sensor, se ubica en la misma unidad, en la que se encuentra el acelerómetro de tres ejes, así trabajan en conjunto. Por una parte, el acelerómetro calculará la posición, por otra, el giroscopio calculará el ángulo en el que se encuentra. Esta unidad que integra los dos elementos se denomina Unidad de Medición Inercial o IMU (Inertial Measurement Unit).



Figura 21: MPU6050.

Esta unidad electrónica que se encarga de realizar las mediciones que determinan las actuaciones del dron y, por tanto, le permite conocer cuál es su posición relativa con respecto al estado inmediatamente anterior. Por medio de la IMU el aeromodelo conoce sus aceleraciones, sus desplazamientos y sus posiciones en cada momento.

En el caso de un dron, esto es utilizado para saber la actual tasa de aceleración usando uno o más acelerómetros, y detecta los cambios en atributos rotacionales tales como cabeceo (pitch), alabeo (roll) y guiñada (yaw) usando uno o más giróscopos

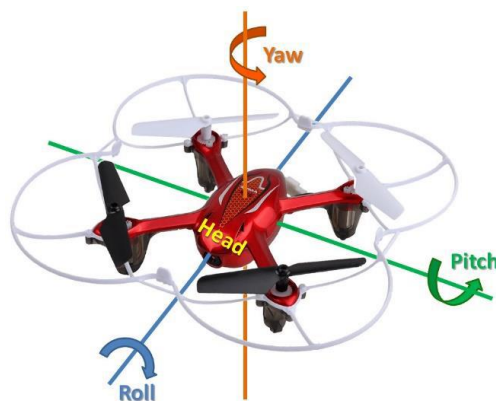


Figura 22: Ejes de movimiento de un dron.

Para la comunicación entre Arduino y MPU6050 se utiliza el bus I2C (Inter – Integrated Circuit). La conexión I2C se realiza mediante los pines A4 a A5 , también denominado TWI (Two Wire Interface) es un bus creado por Phillips en 1970 para reducir la utilización de cables en el conexionado de sus circuitos.

Tan solo dispone de dos conexiones, SDA por donde se transmiten los datos, y SCL por el que se transmite una señal de reloj. Se pueden conectar un total de 112 dispositivos con una distancia total máxima de 1 metro.

3.14 EMISOR

El emisor será el encargado de hacer de interlocutor con nuestro receptor que estará integrado en nuestro dron, este será capaz de emitir la ordenes correspondientes a una frecuencia de 2.408-2.475 GHz con un ancho de banda de 500 KHz y codificación GFSK.



Figura 23: Emisor.

El mando pose 8 canales de transmisión los cuales no sobrepasaran una potencia máxima de 20 dBm.

En la parte posterior del mando incluiremos el sistema de emergencia que constará de dos botones uno de corte suministro de corriente lo cual hará que el dron caiga a peso y el otro

donde el dron descenderá lentamente hasta aterrizar, dentro de esta caja ira acoplado el arduino mini y lora que se comunicaran con el dron.

3.15 RECEPTOR

El receptor es el encargado de recibir las señales del emisor, este se conectará al Arduino micro mediante cableado, este también posee 8 canales y funciona a la misma frecuencia que el emisor 2.408-2.475 GHz con un ancho de banda de 500 KHz y modulación GFSK. La potencia del receptor será de -105 dBm y constará de 8 canales.



Figura 24: Receptor.

4. MÉTODOS

A continuación, se va a enumerar los pasos seguidos para elaborar al completo nuestro proyecto dron.

4.1 CIRCUITO ELECTRONICO DEL DRON

En este apartado se detallará toda la electrónica que lleva montado nuestro dron todos los esquemas eléctricos como la descripción de cada uno de ellos cuál es su principal funcionamiento y que mejoras se podrían implantar en un futuro, lo dividiremos en 3 esquemas principales: Arduino-MPU6050, Arduino-Variadores-Motores, Arduino-Receptor.

4.2 CONEXIÓN ARDUINO-MPU6050

A continuación, mostramos el esquema detallado de las conexiones del Arduino-MPU6050.

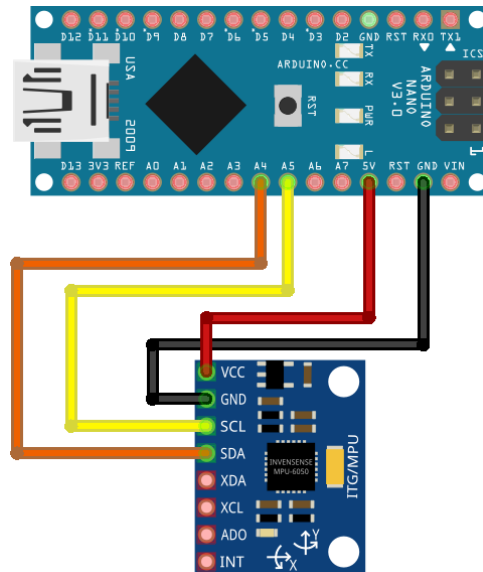


Figura 25: Arduino-MPU6050.

- Pin VCC al pin 5 V de Arduino.
- Pin GND al pin GND de Arduino.
- Pin SCL al pin A5 de Arduino.
- Pin SDA al pin A4 de Arduino.

El Pin SDA es el encargado de mandar datos por cada ciclo que le manda el SCL, el pin SCL es un reloj que cada vez da un pulso manda un dato y el SDA se encarga de pasar el dato a arduino para que este lo procese, gracias a estos 2 pines nos ahorramos muchos cablecitos ya que es capaz que con solo 2 pines podamos manejar tanto el acelerómetro y el giroscopio del MPU6050.

Colocaremos nuestro sensor MPU6050 lo más centrado y plano posible dentro del cuerpo del dron, en la dirección frontal de nuestro dron la cual nos marca con una flecha Y.

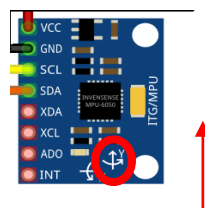


Figura 26: Orientacion-MPU6050.

Ya que la posición de este es crucial para tener la referencia 0 en todos los ejes, una vez situado en su posición final cargaremos unas librerías desarrolladas por **Jeff Rowberg** para calibrar el MPU6050 esto hará que pequeños offset que podamos tener físicamente o electrónicamente los podamos corregir mediante programación y así asegurarnos de que no haya ningún error en cada componente.

4.3 CALIBRADO-MPU6050

Si queremos tener un dron bien calibrado y que tenga una respuesta fina no debemos dejar pasar este paso del calibrado y asegurarnos de cuando nuestro dron esté en reposo cada componente se acerque a 0.

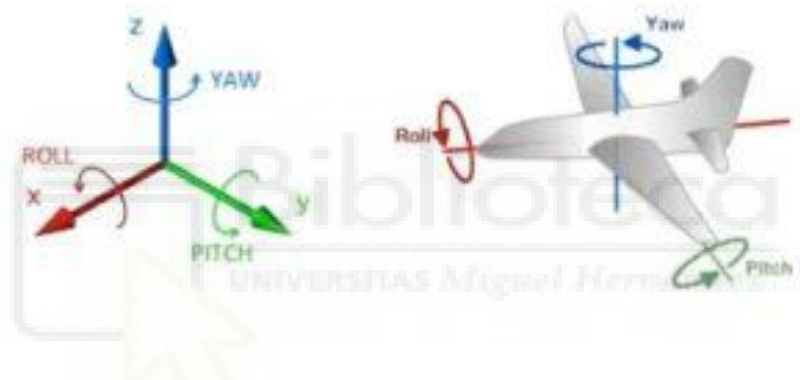


Figura 27: Ejes Tait-Brayan o Euler.

Para solucionar estos problemas debemos subir el siguiente código para compensar dichos offsets.

```
// calibrar_mpu6050.ino
// Librerías I2C para controlar el mpu6050 con Arduino,

#include <I2Cdev.h>
#include <MPU6050.h>
#include <Wire.h>

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implícito
MPU6050 sensor;
```

```

// Valores RAW (sin procesar) del acelerometro y giroscopio en los ejes x,y,z
int ax, ay, az;
int gx, gy, gz;

//Variables usadas por el filtro pasa bajos
long f_ax,f_ay, f_az;
int p_ax, p_ay, p_az;
long f_gx,f_gy, f_gz;
int p_gx, p_gy, p_gz;
int counter=0;

//Valor de los offsets
int ax_o,ay_o,az_o;
int gx_o,gy_o,gz_o;

void setup() {
  Serial.begin(57600); //Iniciando puerto serial
  Wire.begin(); //Iniciando I2C
  sensor.initialize(); //Iniciando el sensor

  if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");

  // Leer los offset los offsets anteriores
  ax_o=sensor.getXAccelOffset();
  ay_o=sensor.getYAccelOffset();
  az_o=sensor.getZAccelOffset();
  gx_o=sensor.getXGyroOffset();
  gy_o=sensor.getYGyroOffset();
  gz_o=sensor.getZGyroOffset();

  Serial.println("Offsets:");
  Serial.print(ax_o); Serial.print("\t");
  Serial.print(ay_o); Serial.print("\t");
  Serial.print(az_o); Serial.print("\t");

```



```

Serial.print(gx_o); Serial.print("\t");
Serial.print(gy_o); Serial.print("\t");
Serial.print(gz_o); Serial.println("\t");

Serial.println("\n\nEnvie cualquier caracter para empezar la calibracionnn");

// Espera un caracter para empezar a calibrar
while (true){if (Serial.available()) break;}
Serial.println("Calibrando, no mover IMU");
}
void loop() {
// Leer las aceleraciones y velocidades angulares
sensor.getAcceleration(&ax, &ay, &az);
sensor.getRotation(&gx, &gy, &gz);

// Filtrar las lecturas
f_ax = f_ax-(f_ax>>5)+ax;
p_ax = f_ax>>5;

f_ay = f_ay-(f_ay>>5)+ay;
p_ay = f_ay>>5;

f_az = f_az-(f_az>>5)+az;
p_az = f_az>>5;

f_gx = f_gx-(f_gx>>3)+gx;
p_gx = f_gx>>3;

f_gy = f_gy-(f_gy>>3)+gy;
p_gy = f_gy>>3;

f_gz = f_gz-(f_gz>>3)+gz;
p_gz = f_gz>>3;

//Cada 100 lecturas corregir el offset
if (counter==100){

```

```

//Mostrar las lecturas separadas por un [tab]
Serial.print("promedio:"); Serial.print("\t");
Serial.print(p_ax); Serial.print("\t");
Serial.print(p_ay); Serial.print("\t");
Serial.print(p_az); Serial.print("\t");
Serial.print(p_gx); Serial.print("\t");
Serial.print(p_gy); Serial.print("\t");
Serial.println(p_gz);

//Calibrar el acelerometro a 1g en el eje z (ajustar el offset)
if (p_ax>0) ax_o--;
else {ax_o++;}
if (p_ay>0) ay_o--;
else {ay_o++;}
if (p_az-16384>0) az_o--;
else {az_o++;}

sensor.setXAccelOffset(ax_o);
sensor.setYAccelOffset(ay_o);
sensor.setZAccelOffset(az_o);

//Calibrar el giroscopio a 0°/s en todos los ejes (ajustar el offset)
if (p_gx>0) gx_o--;
else {gx_o++;}
if (p_gy>0) gy_o--;
else {gy_o++;}
if (p_gz>0) gz_o--;
else {gz_o++;}

sensor.setXGyroOffset(gx_o);
sensor.setYGyroOffset(gy_o);
sensor.setZGyroOffset(gz_o);

```



```

counter=0;
}
counter++;
}

```

Para calibrarlo debemos mantenerlo en su posición final y le cargaremos un programa que empiece a leer los offsets usando un filtro y cada 100 lecturas comprueba los valores si se acercan a los que deseamos leer, aumentando o disminuyendo los offsets. Esto hará que las lecturas filtradas se acerquen a:

-aceleración: p_ax=0, p_ay=0, p_az=+16384

-velocidad angular: p_gx=0 , p_gy=0 , p_gz=0

```

COM10
Offsets:
0 0 0 116 -6 -15
nnEnvie cualquier caracter para empezar la calibracionnn
Calibrando, no mover IMU
promedio: -18 3 17210 467 3 1
promedio: -25 7 17920 470 -3 -3
promedio: -26 4 17936 457 -4 3
promedio: -26 -5 17938 459 5 -13
promedio: -26 2 17942 445 0 1
promedio: -26 7 17962 449 2 -1
promedio: -20 3 17934 449 4 1
promedio: -35 -17 17910 438 -7 -1
promedio: -28 -2 17931 440 2 -2
promedio: -31 0 17916 426 -4 3
promedio: -28 8 17934 429 -7 -3

```

Figura 28: Calibrado del MPU6050.

Ahora anotamos las compensaciones obtenidas usando la función *mpu.setXaccelOffset()*.

Este paso solo es necesario hacerlo una vez, ya que el MPU6050 siempre va estar en la misma posición y con esta parte de calibrado lo que queremos es afinar al máximo cualquier desviación que podamos tener.

Esto sería genial si solo utilizáramos el acelerómetro para determinar el ángulo, cualquier aceleración generada por un desplazamiento generaría errores en el ángulo. En cambio sí solo usamos el giroscopio vamos a obtener un error acumulativo por causa de la integración de w (velocidad angular). Este filtro se utiliza cuando, queremos censar el ángulo pero el MPU está en constante movimiento (Drones, robots móviles, etc).

ESCALADO DE LECTURAS

Ahora es el paso de escalar las lecturas captadas por la aceleración y velocidad angular, cargamos el siguiente código que usando una ecuación determinamos los valores leídos en valores de aceleración o velocidad angular.

```
// escalar_valores.ino
// Librerías I2C para controlar el mpu6050 con Arduino,
// la librería MPU6050.h necesita I2Cdev.h y la librería I2Cdev.h necesita Wire.h
/*
Conociendo los rangos con los que está configurado nuestro MPU6050,
dichos rangos pueden ser 2g/4g/8g/16g para el acelerómetro y
250/500/1000/2000(°/s) para el giroscopio.
Los rangos por defecto (2g y 250°/s)

Variable      valor mínimo  valor central  valor máximo
Lectura MPU6050  -32768      0              +32767
Aceleración     -2g         0g             +2g
Velocidad angular -250°/s     0°/s          +250°/s
*/

#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implícito
MPU6050 sensor;

// Valores RAW (sin procesar) del acelerómetro y giroscopio en los ejes x,y,z
int ax, ay, az;
int gx, gy, gz;

void setup() {
  Serial.begin(57600); //Iniciando puerto serial
  Wire.begin();       //Iniciando I2C
```

```

sensor.initialize(); //Iniciando el sensor

if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");
else Serial.println("Error al iniciar el sensor");

}

void loop() {

// Leer las aceleraciones y velocidades angulares

sensor.getAcceleration(&ax, &ay, &az);

sensor.getRotation(&gx, &gy, &gz);

float ax_m_s2 = ax * (9.81/16384.0);

float ay_m_s2 = ay * (9.81/16384.0);

float az_m_s2 = az * (9.81/16384.0);

float gx_deg_s = gx * (250.0/32768.0);
float gy_deg_s = gy * (250.0/32768.0);
float gz_deg_s = gz * (250.0/32768.0);

//Mostrar las lecturas separadas por un [tab]
Serial.print("a[x y z](m/s2) g[x y z](deg/s):\t");
Serial.print(ax_m_s2); Serial.print("\t");
Serial.print(ay_m_s2); Serial.print("\t");
Serial.print(az_m_s2); Serial.print("\t");
Serial.print(gx_deg_s); Serial.print("\t");
Serial.print(gy_deg_s); Serial.print("\t");
Serial.println(gz_deg_s);

delay(100);
}

```

Con el MPU6050 solo podemos obtener los ángulos X e Y, puesto que una rotación en el eje Z del acelerómetro no lo detectaría ya que usamos la gravedad para determinar el ángulo,

para este ángulo lo mejor es usar un Magnetómetro como por ejemplo el HMC58883L, esto es una de las mejoras a implementar.

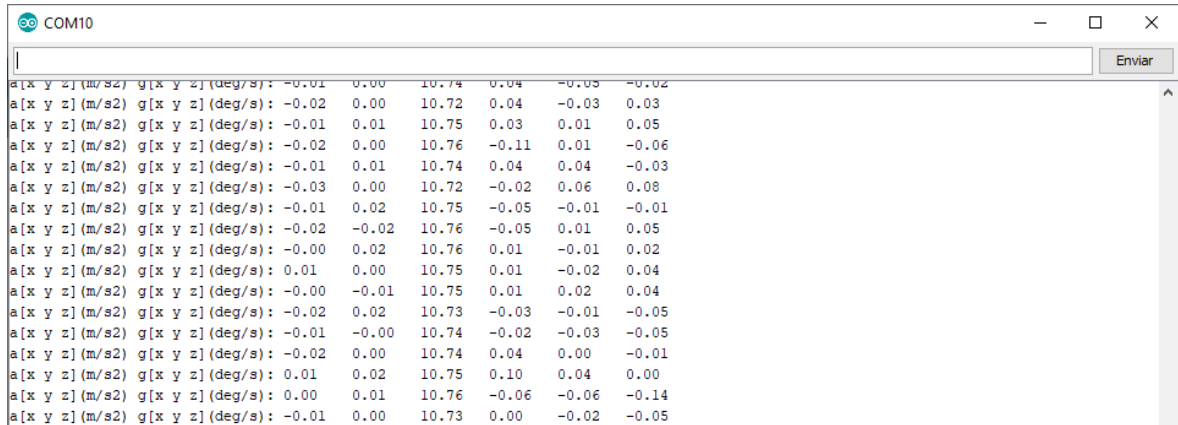
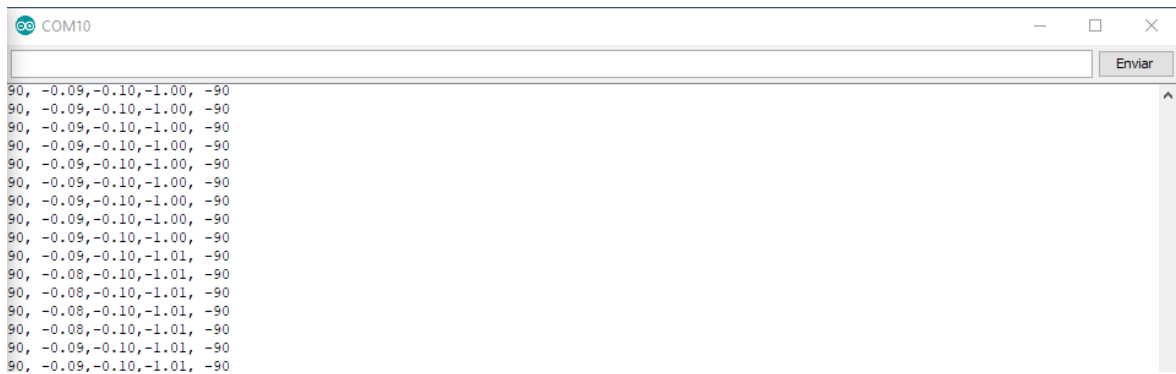


Figura 29: Escalar valores de Acele/Grados.

Los valores obtenidos ya están escalados a unidades de aceleración y velocidad angular, hemos convertido la aceleración a valores en m/s² por lo que se reemplazó el valor de g=9.81 si el sensor se mantiene en posición horizontal se deben obtener mediciones cercanas a 9.8 m/s² (aceleración de la gravedad terrestre) es la componente z de la aceleración.

SIMULACION GRADOS SENSOR MPU6050

Vamos a hacer una pequeña demostración para comprender mejor lo que estamos haciendo y hacer un pequeño punto de control en este punto del proyecto, montamos todo en el dron y vamos a simular grados a ver si responde los parámetros Yaw, Roll y Pitch a medida que se generan al cambiar la posición del IMU.



```
COM10
90, -0.09,-0.10,-1.00, -90
90, -0.09,-0.10,-1.00, -90
90, -0.09,-0.10,-1.00, -90
90, -0.09,-0.10,-1.00, -90
90, -0.09,-0.10,-1.00, -90
90, -0.09,-0.10,-1.00, -90
90, -0.09,-0.10,-1.00, -90
90, -0.09,-0.10,-1.00, -90
90, -0.09,-0.10,-1.00, -90
90, -0.09,-0.10,-1.00, -90
90, -0.09,-0.10,-1.01, -90
90, -0.08,-0.10,-1.01, -90
90, -0.08,-0.10,-1.01, -90
90, -0.08,-0.10,-1.01, -90
90, -0.08,-0.10,-1.01, -90
90, -0.08,-0.10,-1.01, -90
90, -0.08,-0.10,-1.01, -90
90, -0.08,-0.10,-1.01, -90
90, -0.09,-0.10,-1.01, -90
```

Figura 30: grados IMU.

OBTENER INCLINACION

El código **obtener inclinación** viene bastante bien descrito y no hará falta muchas aclaraciones.

* El factor de conversión: $w = \text{Lectura} * (250/32768) = \text{lectura} * (1/131)$

*/

```
#include "Wire.h"
```

```
//Direccion I2C de la IMU
```

```
#define MPU 0x68
```

```
//Ratios de conversion
```

```
#define A_R 16384.0 // 32768/2
```

```
#define G_R 131.0 // 32768/250
```

```
//Conversion de radianes a grados 180/PI
```

```
#define RAD_A_DEG = 57.295779
```

```
//MPU-6050 da los valores en enteros de 16 bits
```

```
//Valores RAW
```

```
int16_t AcX, AcY, AcZ, GyX, GyY, GyZ;
```

```

//Angulos
float Acc[2];
float Gy[3];
float Angle[3];

String valores;

long tiempo_prev;
float dt;

void setup()
{
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  Serial.begin(115200);
}
void loop()
{
  //Leer los valores del Acelerometro de la IMU
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,6,true); //A partir del 0x3B, se piden 6 registros
  AcX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros

  AcY=Wire.read()<<8|Wire.read();
  AcZ=Wire.read()<<8|Wire.read();

  //A partir de los valores del acelerometro, se calculan los angulos Y, X
  //respectivamente, con la formula de la tangente.
  Acc[1] = atan(-1*(AcX/A_R)/sqrt(pow((AcY/A_R),2) + pow((AcZ/A_R),2)))*RAD_TO_DEG;
  Acc[0] = atan((AcY/A_R)/sqrt(pow((AcX/A_R),2) + pow((AcZ/A_R),2)))*RAD_TO_DEG;

```



```

//Leer los valores del Giroscopio
Wire.beginTransmission(MPU);
Wire.write(0x43);
Wire.endTransmission(false);
Wire.requestFrom(MPU,6,true); //A partir del 0x43, se piden 6 registros
GyX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
GyY=Wire.read()<<8|Wire.read();
GyZ=Wire.read()<<8|Wire.read();

//Calculo del angulo del Giroscopio
Gy[0] = GyX/G_R;
Gy[1] = GyY/G_R;
Gy[2] = GyZ/G_R;

dt = (millis() - tiempo_prev) / 1000.0;
tiempo_prev = millis();

//Aplicar el Filtro Complementario
Angle[0] = 0.98 *(Angle[0]+Gy[0]*dt) + 0.02*Acc[0];
Angle[1] = 0.98 *(Angle[1]+Gy[1]*dt) + 0.02*Acc[1];

//Integración respecto del tiempo para calcular el YAW
Angle[2] = Angle[2]+Gy[2]*dt;

//Mostrar los valores por consola
valores = "90, " +String(Angle[0]) + "," + String(Angle[1]) + "," + String(Angle[2]) + ", -90";
Serial.println(valores);

```

4.4 CONEXIÓN ARDUINO-ESC-MOTORES

El conexionado de los motores es uno de los pasos a tener muy en cuenta, ya que de nada valdría tener una buena calibración del sensor para mandar órdenes a los motores si estos no giran en el sentido adecuado.

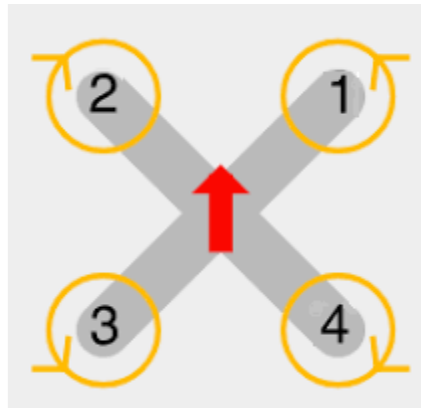


Figura 31: Sentido de giros de los motores.

Para que cada motor tenga un sentido de giro u otro debemos cruzar los cables de los motores a los ESC, en la Figura 25 tenemos un ejemplo de conexión Motor-ESC.

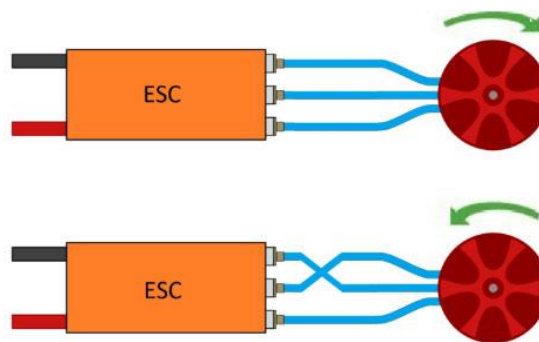


Figura 32: Conexión Motor-ESC.

En la figura 26 hemos detallado un esquema de conexiones Motor-ESC-Arduino-PDB., Desde el motor hacia el ESC tendremos 3 cables aquí es donde tenemos que hacer bien el conexionado para hacer que el giro del motor sea el correcto, como podemos observar en la siguiente imagen el Motor 2 y el Motor 4 giran en el sentido de las agujas del reloj ya que

su conexión es como el de la figura 26, el Motor 1 y Motor 3 giran en las dirección contraria a las agujas del reloj, en la otra parte del ESC tendremos dos cables de mayor sección normalmente uno rojo y otro negro y estos irán conectados directamente a la placa distribuidora de potencia(PDB), finalmente nos quedaran 2 o 3 cables, en nuestro caso solo 2, el tercero sería un cable llamado BEC que como sus siglas indican se refiere a “Battery Electronic Controler” y es un cable para gestionar la batería y la potencia entregada a los motores. De estos dos cables uno es masa color negro y el otro es el de señal el color suele cambiar según el fabricante, Según los pines que hayamos declarado en el programa Arduino es donde tendremos que hacer el conexionado normalmente este se suele hacer en los pines digitales que van desde el D2 hasta el D12.

- Motor 1 pin digital D3 del Arduino.
- Motor 2 pin digital D4 del Arduino.
- Motor 3 pin digital D5 del Arduino.
- Motor 4 pin digital D6 del Arduino.

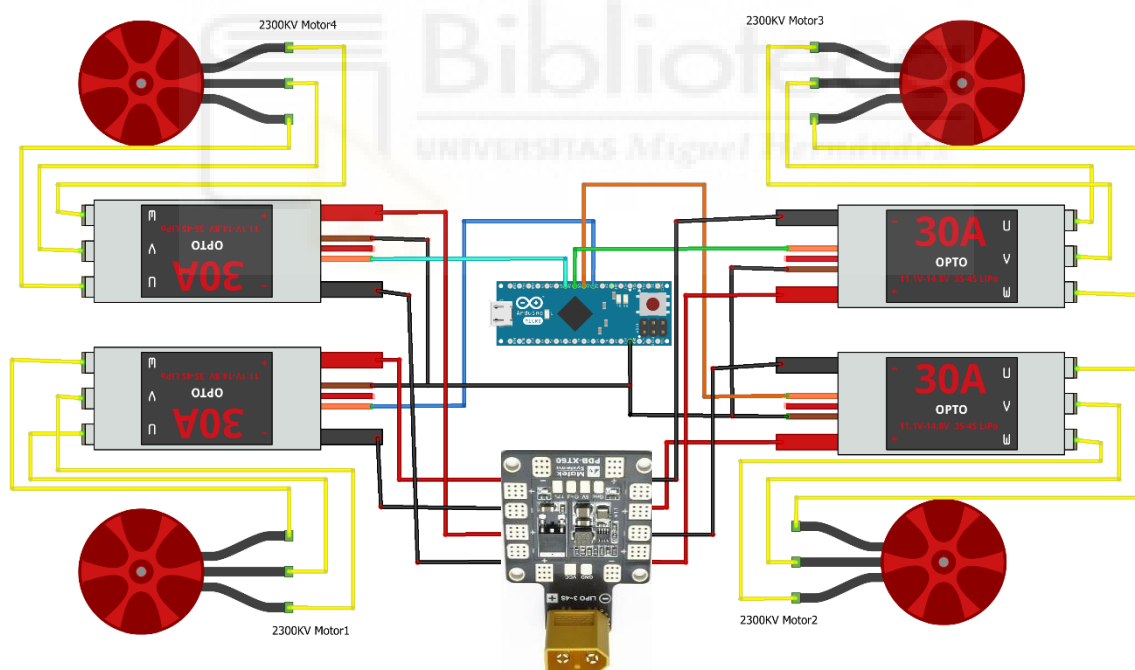


Figura 33: Arduino-ESC-Motores.

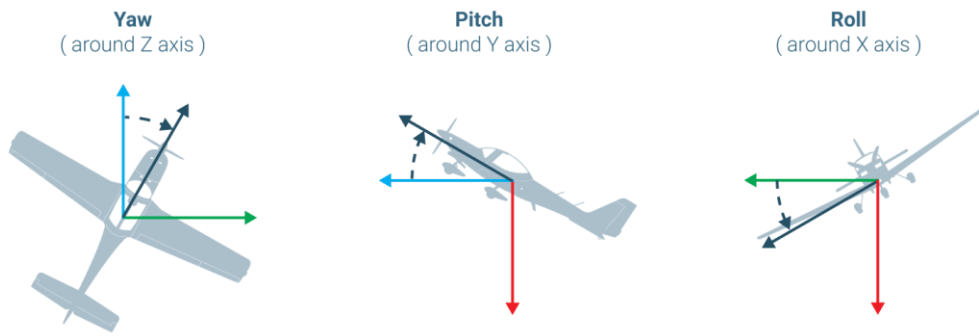


Figura 34: Esquema Angular.

4.5 CONEXIÓN ARDUINO-RECEPTOR

El receptor tiene una capacidad de hasta 8 canales de comunicación, en nuestro caso solo utilizaremos 5, el canal 1 para el Roll(alabeo), el canal 2 para el Throttle (potencia), el canal 3 para el Yaw (guiñada), el canal 4 para el Pitch (cabeceo), y el canal 5(modos de vuelo) estos controles los podemos ver gráficamente en la figura 27.

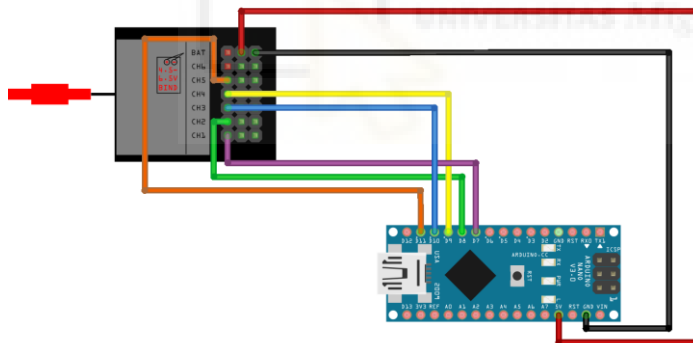


Figura 35: Arduino-Receptor.

El conexionado quedaría de la siguiente manera (Figura 35):

- Canal 1 con el pin digital 7 del Arduino.
- Canal 2 con el pin digital 8 del Arduino.
- Canal 3 con el pin digital 10 del Arduino.
- Canal 4 con el pin digital 9 del Arduino.
- Canal 5 con el pin digital 11 del Arduino
- VCC con VCC del Arduino.
- GND con GND.

4.6 Circuito FPV(First Person View)

El emisor de video de la marca Eachine modelo TX801 con una ganancia de antena de 3Dbi se conecta mediante el conector RP-SMA a un rango de frecuencias que van desde 5325-5945MHZ que permite 72 canales, cubren toda la frecuencia de 5.8G con potencias de transmisión de 0.01MW / 5MW / 25MW / 50MW / 100MW / 200MW / 400MW / 600MW conmutable.

Especificaciones:

- Formato de video: NTSC / PAL
- Corriente de trabajo: (MAX) 7V-550mA
- ; 12V-300mA
- ; 24V-160mA "
- Ancho de banda de audio: 6.5MHZ
- Ancho de banda de video: 18 MHZ
- Portador de audio: 1Vp-p (1KHZ)
- Impedancia de entrada AF: 10 K Ω
- Voltaje de funcionamiento: 7-24 V
- Voltaje de salida: 5 V
- Peso: 7,3 g
- Tamaño: 28 * 20 * 8MM



Figura 36: Emisor de video.

72 Channels Cover All 5.8G Frequency

| Band | CH1 | CH2 | CH3 | CH4 | CH5 | CH6 | CH7 | CH8 |
|--------|------|------|------|------|------|------|------|------|
| Band-A | 5865 | 5845 | 5825 | 5805 | 5785 | 5765 | 5745 | 5725 |
| Band-b | 5733 | 5752 | 5771 | 5790 | 5809 | 5828 | 5847 | 5866 |
| Band-E | 5705 | 5685 | 5665 | 5645 | 5885 | 5905 | 5925 | 5945 |
| Band-F | 5740 | 5760 | 5780 | 5800 | 5820 | 5840 | 5860 | 5880 |
| Band-r | 5658 | 5695 | 5732 | 5769 | 5806 | 5843 | 5880 | 5917 |
| Band-U | 5325 | 5348 | 5366 | 5384 | 5402 | 5420 | 5438 | 5456 |
| Band-o | 5474 | 5492 | 5510 | 5528 | 5546 | 5564 | 5582 | 5600 |
| Band-L | 5333 | 5373 | 5413 | 5453 | 5493 | 5533 | 5573 | 5613 |
| Band-H | 5653 | 5693 | 5733 | 5773 | 5813 | 5853 | 5893 | 5933 |

Tabla 3: Rango de frecuencias Emisor.

Para la obtención de información en tiempo real de vuelo vamos hacerlo mediante una cámara situada en el dron y una antena emisora la cual mandara las señales de video a nuestro receptor.



Figura 37: Gafas FPV.

Nuestro receptor serán una gafas FPV(First Person View) este tipo de recepción de video es muy popular entre los pilotos de drones de carreras o freestyle, con este método vamos a tener una mejor recepción de vuelo con la que podemos acercarnos más a nuestro objetivo y capturar momentos y planos únicos.



Figura 38: Gafas FPV.

En la Tabla 6 se observa una tabla con diferentes bandas, estas 5 bandas de frecuencias contienen 8 frecuencias cada una, gracias al autosearching nuestro receptor buscara rápidamente la banda y frecuencia donde se encuentra emitiendo nuestro emisor de video.

Especialty with RaceBand

Built-in super sensitiveness 5.8GHz 40CH receiver



Auto-Searching

Frequency Table

| | | | | | | | | |
|------------------|------|------|------|------|------|------|------|------|
| Frequency Band A | 5865 | 5845 | 5825 | 5805 | 5785 | 5765 | 5745 | 5725 |
| Frequency Band B | 5733 | 5752 | 5711 | 5790 | 5809 | 5828 | 5847 | 5866 |
| Frequency Band E | 5705 | 5685 | 5665 | 5645 | 5885 | 5905 | 5925 | 5945 |
| Frequency Band F | 5740 | 5760 | 5780 | 5800 | 5820 | 5840 | 5860 | 5880 |
| Frequency Band R | 5658 | 5695 | 5732 | 5769 | 5806 | 5843 | 5880 | 5917 |

Tabla 4: Rango de frecuencias Receptor.

Característica:

- Con esponja de cara y función de AV IN.
- Diseño separable, Pantalla de 5 pulgadas se puede utilizar como monitor pequeño en el controlador de radio o con el trípode.
- HD de alto brillo de 5 pulgadas LCD especialmente adaptado para las carreras de FPV.
- Incorporado Receptor 5.8GHz 40CH de súper sensibilidad, especialmente con raceband.
- Vienen con la función avanzada de búsqueda automática y la frecuencia de trabajo en la Pantalla.
- Incorporado en la batería de 3.7V / 2000mAh , Cada carga completa alrededor de 3.5 horas a revivir el tiempo de trabajo.
- Grande rango de voltaje de recarga de 5-12 V, bancos de la energía o 2S / 3S baterías pueden ser utilizadas como fuente de alimentación de repuesta, para prolongar el tiempo de trabajo; - Ángulo visible súper amplia de hasta 82 grados; - 92% de Lente transparente adoptado, sin distorsionar, hay falta de definición en el adagio de la Pantalla.
- Diseño ergonómico se ajusta a sus caras.

| Especificación | |
|--------------------|--|
| Nombre de la marca | Eachine |
| Artículo No. | EV800 |
| Pantalla | Tamaño de Pantalla: 5.0 pulgadas Resolución de la Pantalla: 800 * 480 (Sin desenfoco después agrandar por la Lente) Brillo de la Pantalla: 600cd / m2 con luz de fondo especial de alto brillo LED para FPV exterior Ángulo de visión: 140/120 grados (horizontal / vertical) |
| Cuaresma | 3x Impulso del vídeo para adquirir una intensa sensación-en-allí de vídeo; 92% De tasa transparente sin distorsionar la luz; |

| | | | | | | | | | |
|-----------------------|---|------|------|------|------|------|------|------|------|
| Batería Y recarga | Batería incorporada / 2000mAh 3.7V; Cada oferta del círculo de la batería alrededor de tiempo de trabajo 3.5h; 5-12 V amplia rango de voltaje de recarga, la tensión típica de carga es de 5V Acepta 5V bancos de la energía o 2S (7.4V) / 3S (11.1V) de la batería como energía de recambio para largo tiempo de trabajo. | | | | | | | | |
| Receptor | Construir en 40CH 5.8 g Raceband recibir | | | | | | | | |
| | Tabla de frecuencia | | | | | | | | |
| | Banda de frecuencia A | 5865 | 5845 | 5825 | 5805 | 5785 | 5765 | 5745 | 5725 |
| | Banda de frecuencia B | 5733 | 5752 | 5771 | 5790 | 5809 | 5828 | 5847 | 5866 |
| | Frecuencia de la banda E | 5705 | 5685 | 5665 | 5645 | 5885 | 5905 | 5925 | 5945 |
| | Banda de frecuencia F | 5740 | 5760 | 5780 | 5800 | 5820 | 5840 | 5860 | 5880 |
| Frecuencia de banda R | 5658 | 5695 | 5732 | 5769 | 5806 | 5843 | 5880 | 5917 | |
| Cuerpo | Dimensión: 180x145x82mm Peso: 349g con batería de 2000mAh 3.5h Cinturón: Cinturón ajustable de tres maneras | | | | | | | | |

Tabla 5: Especificaciones Eachine EV800.

5. MONTAJE

Lo primero que vamos a montar son los motores a los ESC conforme explicamos en la figura 31 teniendo en cuenta cual va ser nuestro frente del dron atendiendo a la figura 29 sabremos cuales son los Motor 1, Motor 2, Motor 3 y Motor 4 y su sentido de giro.

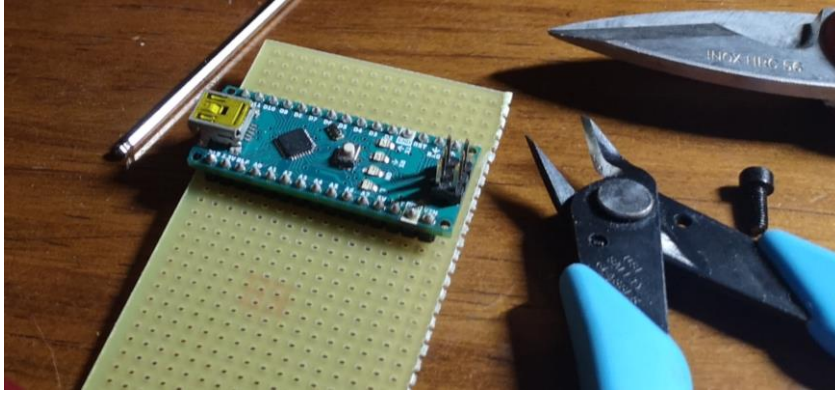


Figura 39: Arduino-Placa Micro perforada.

Como podemos observar en la Figura 40 montamos los motores en los extremos y colocamos los ESC dentro del cuerpo del dron, de esta forma nos aseguramos que en caso de impacto este más protegido ya que las zonas con mayor índice de choque son las patas.

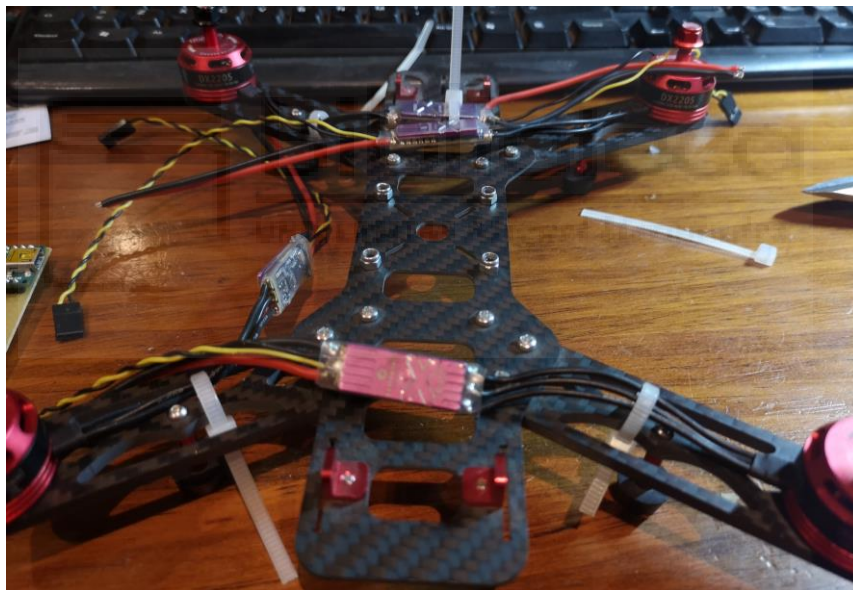


Figura 40: Montaje Motor-ESC.

Seguimos replanteado donde irán colocados todos los componentes, diseñamos en nuestra placa micro perforada la posición del arduino de forma lateral para tener un mejor acceso al USB para futuras actualizaciones o mejoras de código (Figura 42).

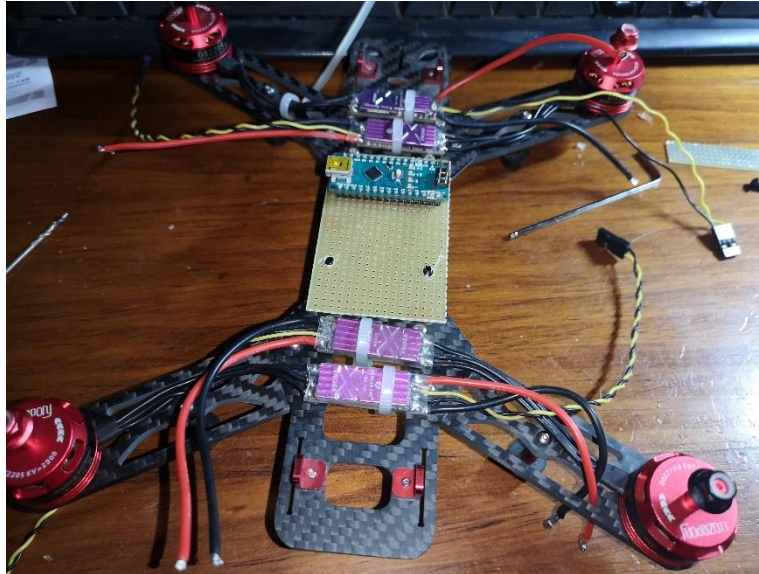


Figura 41: Posición Arduino NANO.

Es la hora de colocar el sensor MPU6050, el encargado de la estabilización del dron, este es uno de los elementos con menos movilidad dentro de nuestro dron, el arduino y los ESC podemos ponerlos en cualquier posición, pero el MPU6050 debe de estar lo más centrado y plano posible dentro del cuerpo del dron, pues es crucial para la lectura de datos. Con lecturas erróneas no podríamos saber con exactitud la posición para poder corregir estos valores.

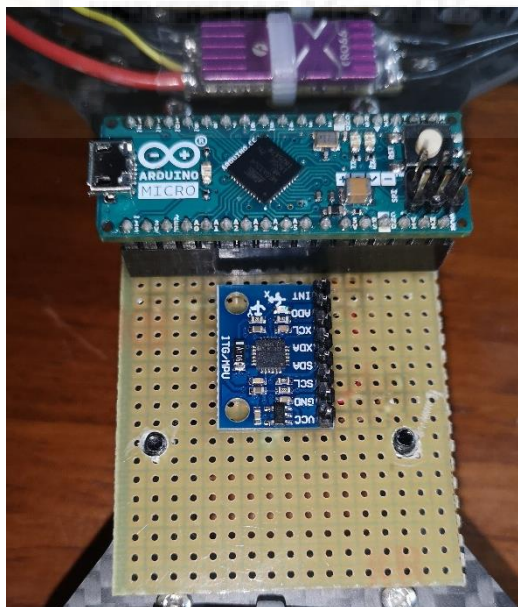


Figura 42: Posición MPU6050.

Soldamos por la parte trasera de nuestra placa micro perforada el MPU6050 al Arduino NANO, para esta soldadura es aconsejable que sea lo más corta y directa posible para evitar tener cables colgando, estas soldaduras están hechas en los zócalos donde irán insertada nuestra placa Arduino NANO y MPU6050, gracias a este sistema cualquier daño sufrido en

estos componentes será subsanado rápidamente simplemente quitando el dañado y sustituyéndolo por uno nuevo (ver figura 41 y 42).

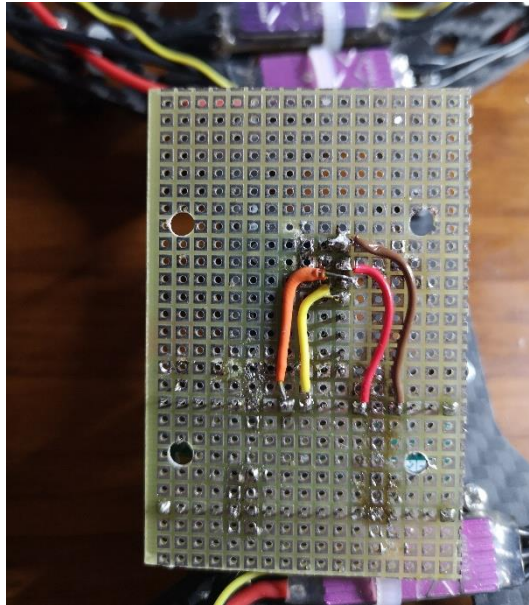


Figura 43: Soldaduras Arduino NANO-MPU6050.

Para las conexiones de los ESC a la placa arduino vamos a tener en cuenta darle unos cm de más al cableado para que seamos capaces de darle la vuelta a la placa una vez soldadas para futuras reparaciones.

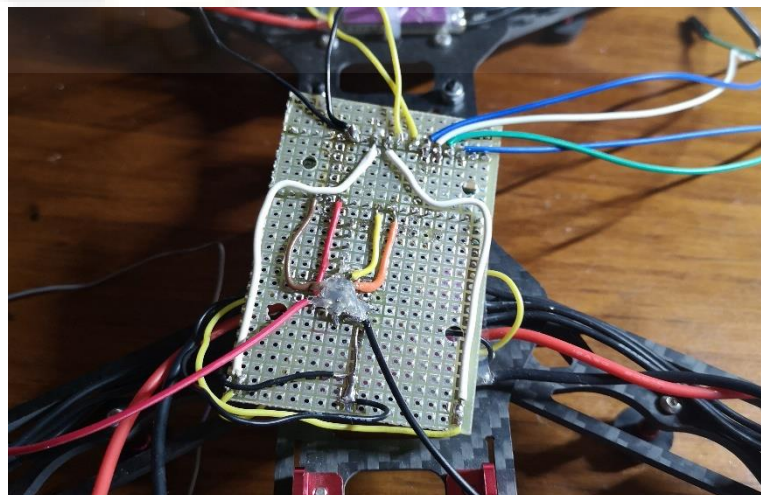


Figura 44: Soldaduras Arduino NANO-MPU6050.

En la siguiente figura 45 podemos observar los zócalos donde irán colocados nuestro micro controlador principal y sensor MPU6050.

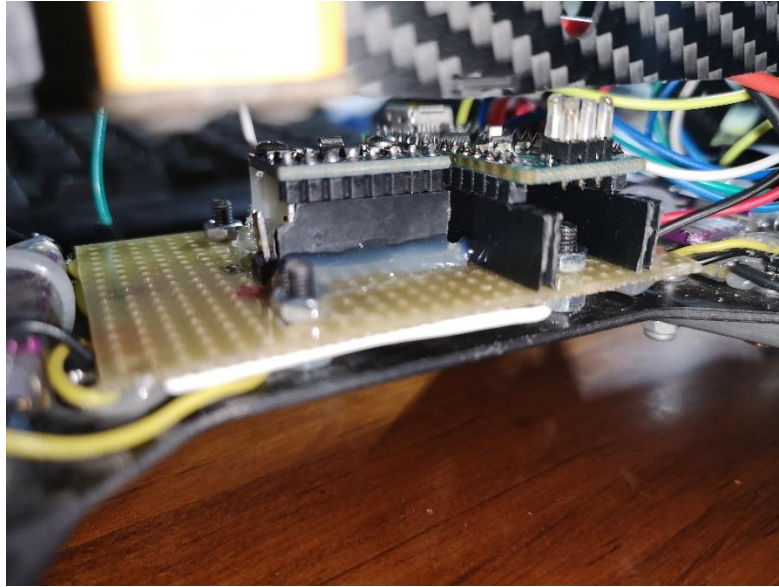


Figura 45: Zócalos Arduino NANO-MPU6050.

Un detalle a tener en cuenta es la posición del Arduino NANO, en la figura 46 vemos como la hemos colocado de forma que nos sea fácil la conexión via mini USB para poder actualizar el programa principal sin tener que desmontar nada, más adelante comentaremos el código de vuelo principal.

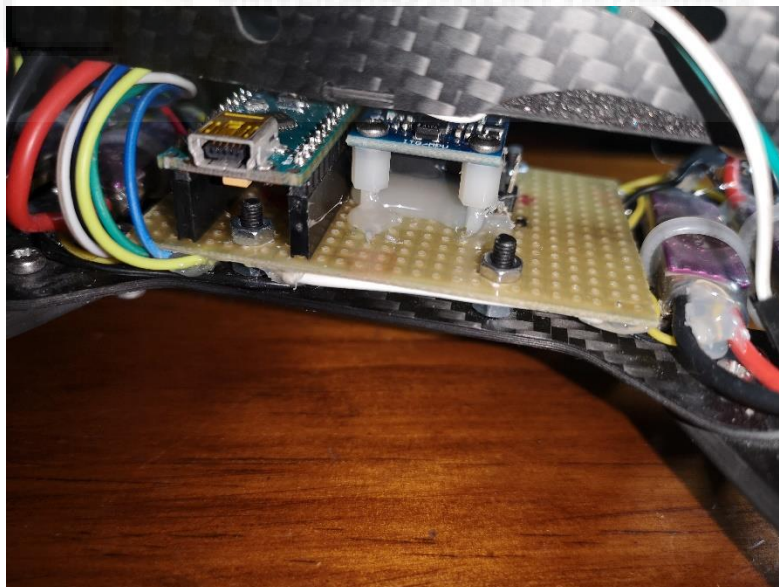


Figura 46: Zócalos Arduino NANO-MPU6050.

Ahora es el turno de colocar la PDB (Power Distribution Board) hemos decidido ponerla arriba de forma que para la conexión con la batería y los ESC sea lo más limpia posible (ver figura 42), tenemos que tener en cuenta que debemos de repartir el peso de todos los

componentes por todo el dron, esto es muy importante ya que de no ser así tendremos una descompensación de peso sobre una zona y esto hará que el MPU6050 este continuamente mandando pulsos para incrementar la potencia de algunos motores para corregir este fallo con lo que conlleva a un sobrecalentamiento de motores que podemos evitar con una buena re planificación.

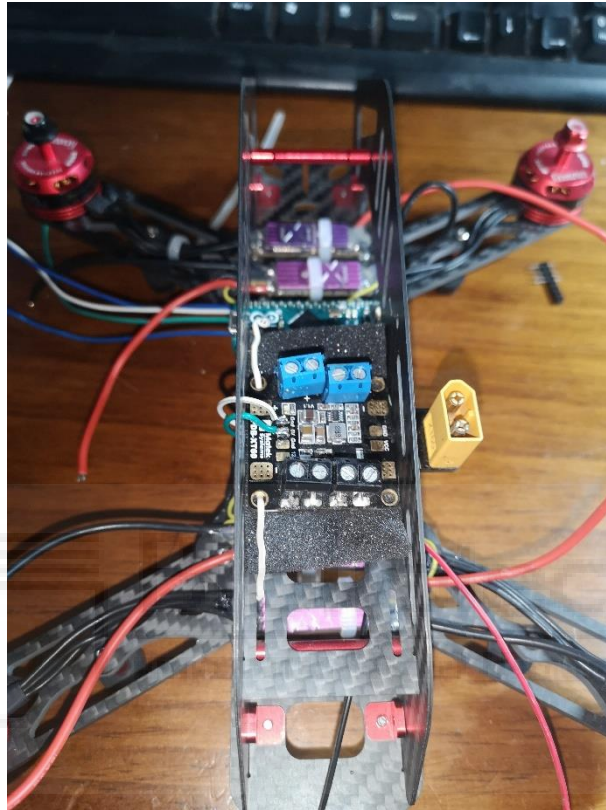


Figura 47: Posición PDB.

El receptor lo podemos situar en cualquier parte del dron siempre siendo conscientes de que tenemos una antena dipolo y atendiendo a su diagrama de radiación vamos a evitar en medida de lo posible orientar la antena hacia arriba o hacia abajo pensando que nosotros vamos a estar debajo del dron. Una buena orientación será cuando el cuerpo del dron y el dipolo sean paralelos al eje X(Roll) (ver figura 48: diagrama de radiación de un dipolo). Para conseguir que las antenas del receptor queden paralelas al suelo vamos a utilizar una brida enganchada al chasis del dron y sobre esta la colocaremos enfundándola con funda termo retráctil para darle rigidez y protección (ver figura 49). Hemos de tener en cuenta que en ningún momento las hélices puedan llegar a tocar la antena ya que esto sería un problema si llegase a cortar la antena, (ver figura 47).

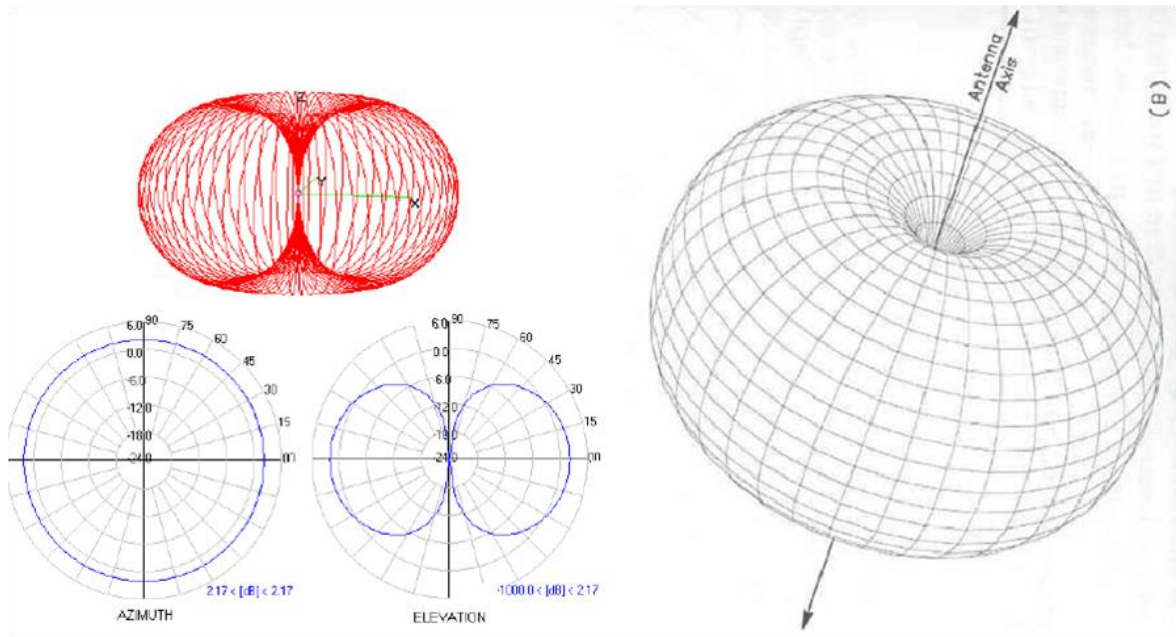


Figura 48: Radiación de un dipolo

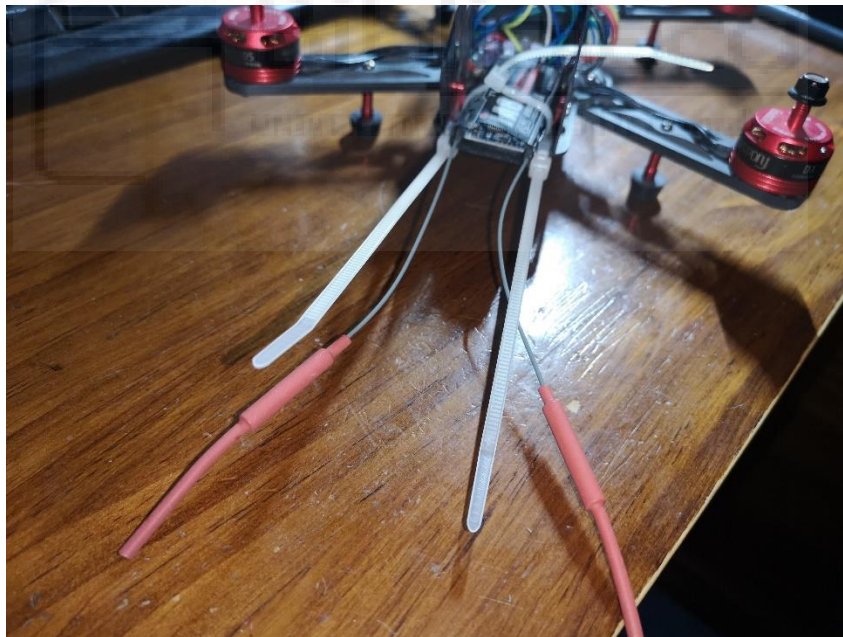


Figura 49: Antena dipolo receptor.

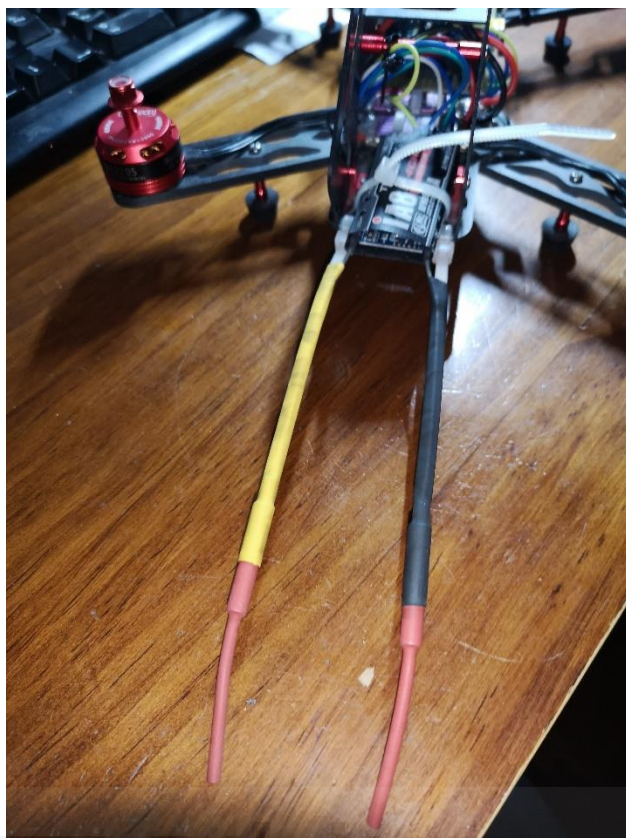


Figura 50: Antena dipolo receptor con funda termo retráctil.

En la siguiente imagen(Figura 51) podemos ver la localización de nuestro sistema lora, hemos adaptado toda la electrónica con un soporte fabricado en 3d de forma que quede lo más integrada con el dron, esta parte se podría mejorar siempre y cuando hagamos una placa de circuito impreso adaptada al dron, partiendo de esta placa ya montada la hemos adaptado de la forma que el dron no perdiese su línea de vuelo.

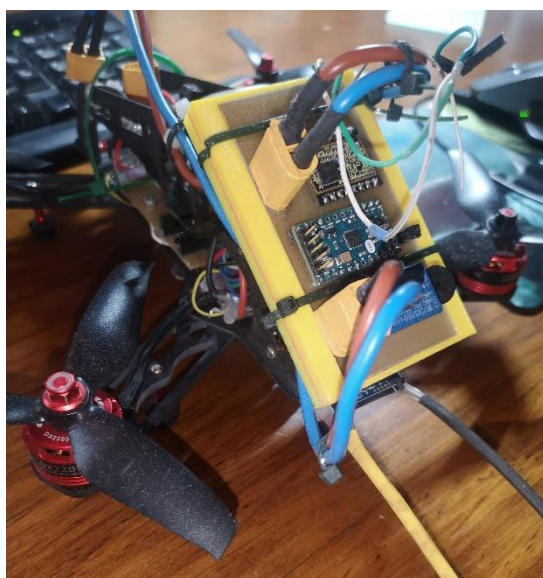


Figura 51: Integración lora.

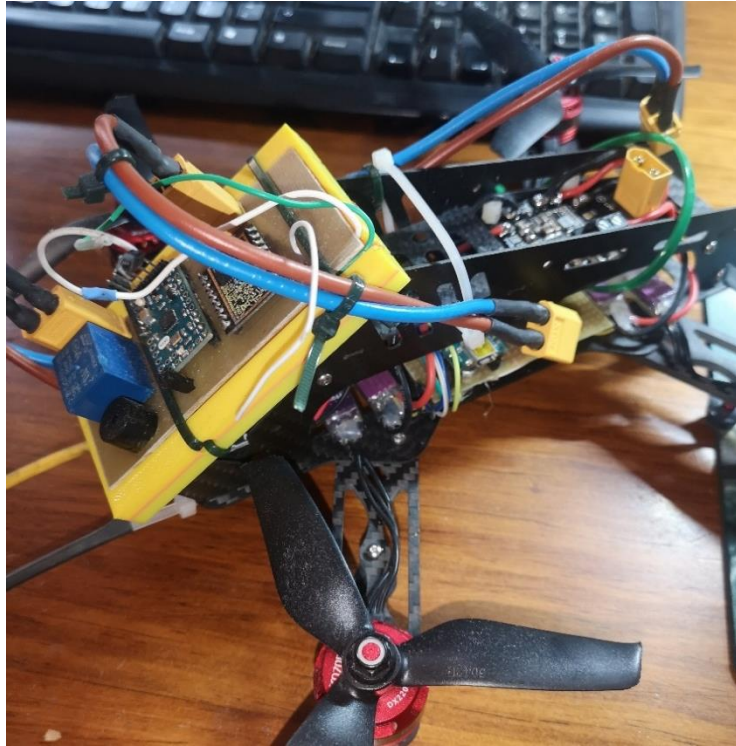


Figura 52: Integración lora y conexionado.

A continuación, se muestra unas imágenes detalladas de conexiones e interruptores del dron, este último lo hemos fijado como un enlace de seguridad entre la conexión de la batería y toda la electrónica ya que al conectar dicha batería se produce un arco voltaico que poco a poco va ennegreciendo los pines de conexionado, así nos evitamos estos desgastes.

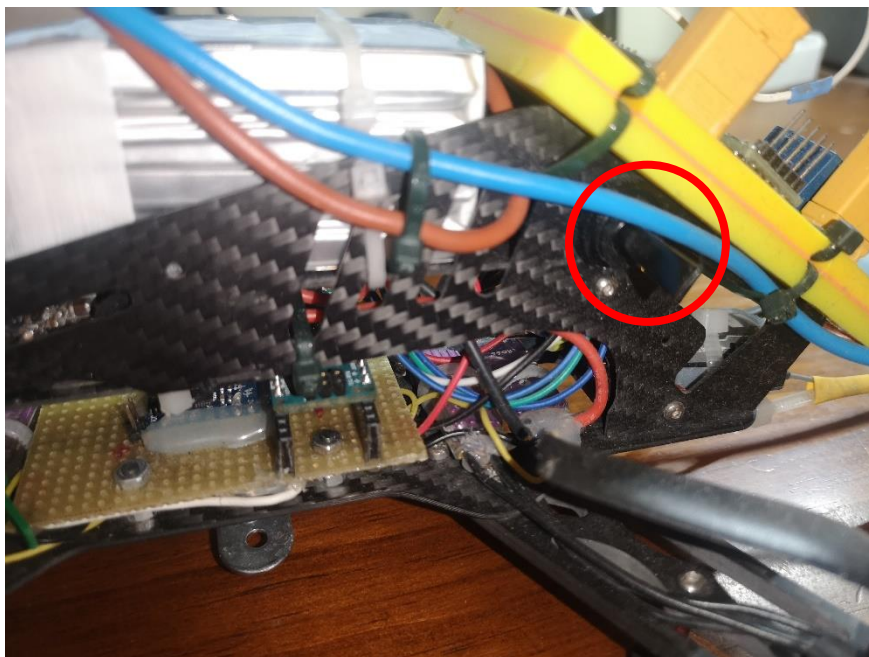


Figura 53: Interruptor dron.

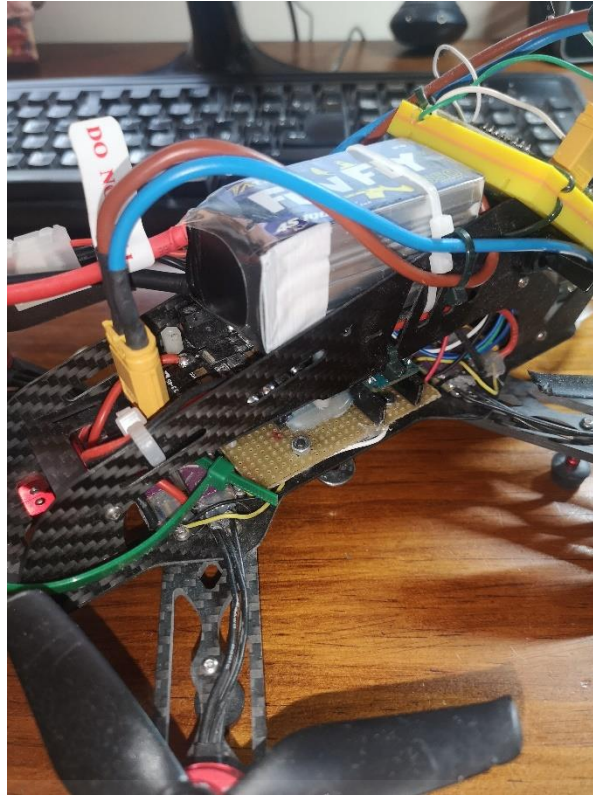


Figura 54: Conexión interruptor a PDB.

En la figura 50 podemos ver los cables que nos llegan del interruptor para alimentar la PDB y desde ahí alimentar toda la electrónica. A continuación, se muestra en la figura 55 el esquema detallado de lora receptor y arduino mini.

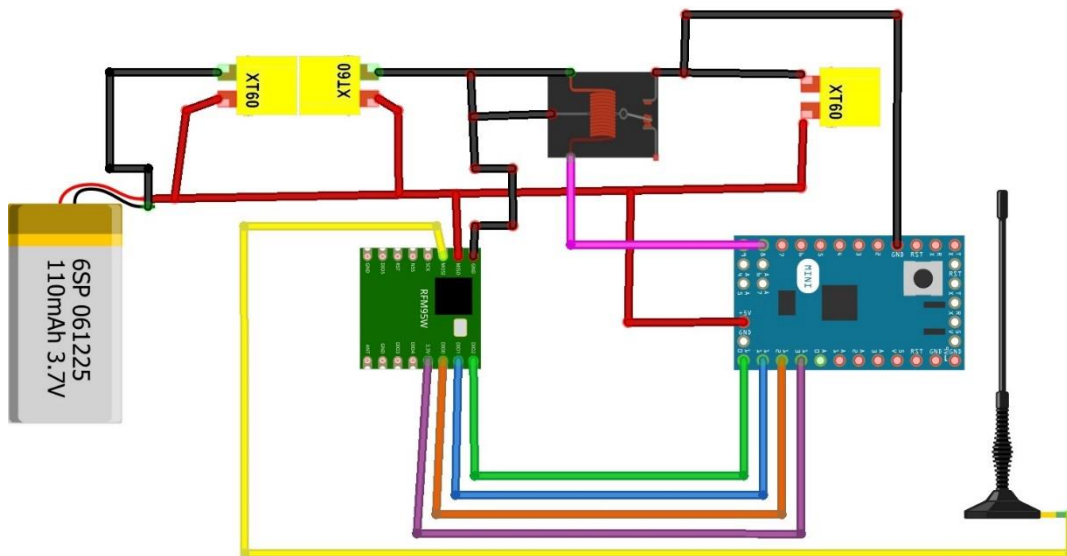


Figura 55: Esquema RF-LoRa-868-SO.

El conexionado sería:

- Pin nSEL conectado a pin digital 10 del Arduino.
- Pin SDI conectado a pin digital 11 del Arduino.
- Pin SDO conectado a pin digital 12 del Arduino.
- Pin SCLK conectado a pin digital 13 del Arduino.
- Pin ANT conectado a un cable, que actúa como antena.
- Pin GND conectado a pin GND.
- Pin VCC del LoRa a VCC del Arduino.
- Pin Vin del Arduino conectado a la batería LiPo.
- Pin positivo del Buzzer a pin digital 3 del Arduino.
- Pin negativo del Buzzer a GND.
- Pin izquierdo Bobina relé a GND.
- Pin interruptor relé a GND.
- Pin derecho Bobina relé a pin digital 9 del Arduino.
- Pin normalmente cerrado del relé ira conectado a un XT60 para cortar la corriente eléctrica hacia el dron.

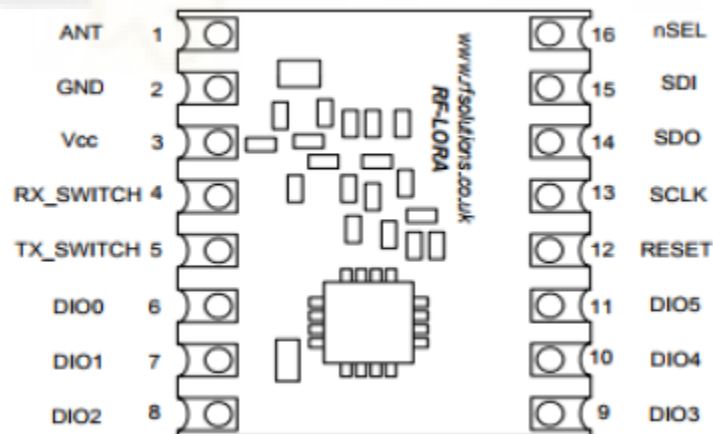


Figura 56: Esquema RF-LoRa-868-SO

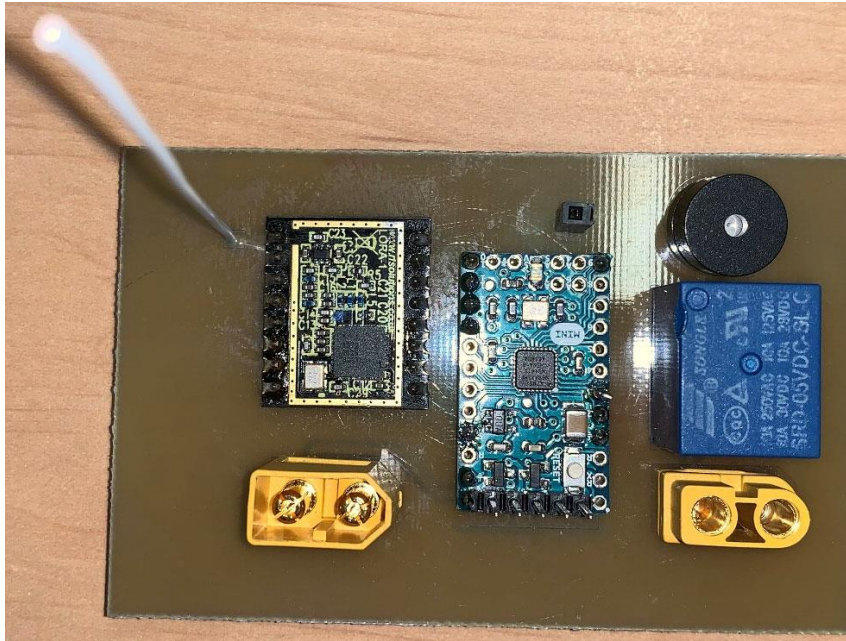


Figura 57: Placa receptora parte delantera.

En la figura 57 tenemos una imagen del circuito lora que se encargara de la recepción de las señales de emergencias emitidas por el sistema lora instalado en el mando, en la imagen podemos ver la fabricación de una placa electrónica con los siguientes componentes:

- Arduino Mini
- RF-LoRa-868-SO
- Relé
- Buzzer
- Conexiones XT 60

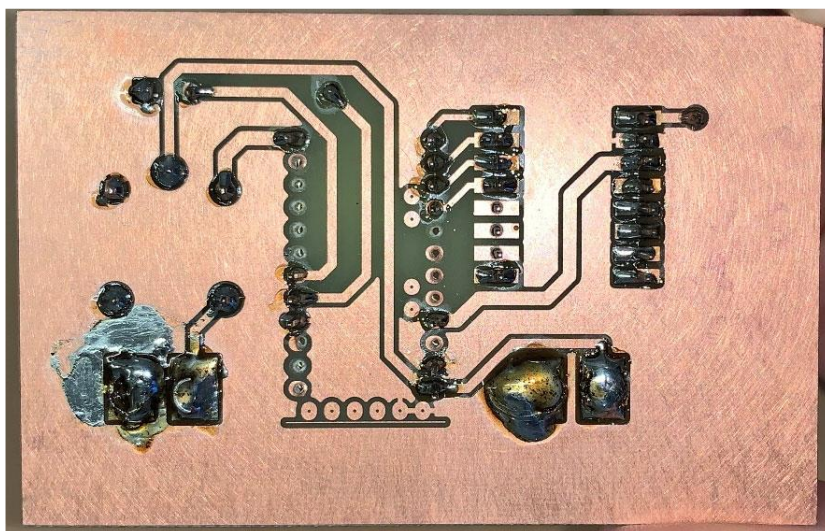


Figura 58: Placa receptora parte trasera.

6. PROGRAMACIÓN

Para la programación del dron vamos a utilizar el entorno de Arduino IDE. El entorno de desarrollo integrado (IDE) de Arduino es una aplicación multiplataforma (para Windows, macOS y Linux) que está escrita en lenguaje C/C++ con la cual podemos programar las distintas placas de Arduino.

Para esta fase del proyecto hemos diferenciado 4 grandes pestañas dentro del entorno en la que volcaremos toda la programación, estas 4 pestañas son:

- Main Principal.
- Calibración MPU6050.
- PIDS.
- Señal PWM.

A continuación, vamos a explicar cada una de estas partes.

6.1 MAIN PRINCIPAL:

```
/*Código para Dron de carreras: *
 * "Mark01" *
 * *
 * Hecho por: *
 * Jose Alberto Juan Segura *
 *****/

#include <EnableInterrupt.h>
// Librerias I2C para controlar el mpu6050
// la libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h
#include "I2Cdev.h"
#include "MPU6050.h"
#include <Servo.h>
#include <Wire.h> // nos permite la comunicacion entre arduino uno y el MPU6050 mediante el protocolo I2
C

#define MPU6050_address 0x68 //I2C address MPU6050
```

```

#define MAX_SIGNAL 2000
#define MIN_SIGNAL 1000

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implícito
MPU6050 sensor;

// Valores RAW (sin procesar) del acelerometro y giroscopio en los ejes x,y,z
int ax, ay, az;
int gx, gy, gz;

long tiempo_prev;
float dt;
float rollAngulo, pitchAngulo, yawAngulo;
float ang_x_prev, ang_y_prev, ang_z_prev;

//////////////////////Variables Mando RC//////////////////////
float wPitchConsigna, wRollConsigna, wYawConsigna = 0.00;

///PIDs
float Roll_referencia=0.00, Roll_error_p, Roll_error_i, Roll_error_d, Roll_error_p_anterior,ITerm_roll_ang,
Pid_Roll_salida=0.00;
float Pitch_referencia=0.00, Pitch_error_p, Pitch_error_i, Pitch_error_d,ITerm_pitch_ang, Pitch_error_p_anterior,
Pid_Pitch_salida=0.00;
float Yaw_referencia=0.00, Yaw_error_p, Yaw_error_i, Yaw_error_d,ITerm_yaw_w, Yaw_error_p_anterior,
Pid_Yaw_salida=0.00;

Servo esc1, esc2, esc3, esc4;
const int ESC1_PIN = 3;
const int ESC2_PIN = 4;
const int ESC3_PIN = 5;
const int ESC4_PIN = 6;

volatile int Pitch;

```

```

volatile int Roll;
volatile int Yaw;

//Tiempos
long tiempo_ejecucion =0.0, loop_timer =0.0, timePrev =0.0;

volatile long contPotenciaInit; // LEER MANDO RC --> POTENCIA
volatile int PulsoPotencia;
void INTpotencia() {
    if (digitalRead(8) == HIGH) contPotenciaInit = micros();
    if (digitalRead(8) == LOW) PulsoPotencia = micros() - contPotenciaInit;
}

volatile long contPitchInit; // LEER MANDO RC --> PITCH
volatile int PulsoPitch;
void INTpitch() {
    if (digitalRead(9) == HIGH) contPitchInit = micros();
    if (digitalRead(9) == LOW) PulsoPitch = micros() - contPitchInit;
}

volatile long contRollInit; // LEER MANDO RC --> ROLL
volatile int PulsoRoll;
void INTroll() {
    if (digitalRead(7) == HIGH) contRollInit = micros();
    if (digitalRead(7) == LOW) PulsoRoll = micros() - contRollInit;
}

volatile long contYawInit; // LEER MANDO RC --> YAW
volatile int PulsoYaw;
void INTyaw() {
    if (digitalRead(10) == HIGH) contYawInit = micros();
    if (digitalRead(10) == LOW) PulsoYaw = micros() - contYawInit;
}

```

```

volatile long contModoInit; // LEER MANDO RC --> Modo de vuelo
volatile int PulsoModo;
void INTmodo() {
  if (digitalRead(11) == HIGH) contModoInit = micros();
  if (digitalRead(11) == LOW) PulsoModo = micros() - contModoInit;
}

void command_throttle(int throttle)
{
  esc1.writeMicroseconds(throttle);
  esc2.writeMicroseconds(throttle);
  esc3.writeMicroseconds(throttle);
  esc4.writeMicroseconds(throttle);
}

void setup() {
  Serial.begin(9600);
  Wire.begin(); //Iniciando I2C
  sensor.initialize(); //Iniciando el sensor
  //digitalWrite(10, HIGH); //Encendemos el led durante la calibracion
  /*if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");
  else Serial.println("Error al iniciar el sensor");*/
  //digitalWrite(10, LOW);

  pinMode(8, INPUT_PULLUP); // POTENCIA - Throttle, CH2
  enableInterrupt(8, INTpotencia, CHANGE);
  pinMode(9, INPUT_PULLUP); // PITCH, CH4
  enableInterrupt(9, INTpitch, CHANGE);
  pinMode(7, INPUT_PULLUP); // ROLL, CH1
  enableInterrupt(7, INTroll, CHANGE);
  pinMode(10, INPUT_PULLUP); // YAW, CH3
  enableInterrupt(10, INTyaw, CHANGE);
  pinMode(11, INPUT_PULLUP); // modo vuelo, CH
  enableInterrupt(11, INTmodo, CHANGE);

```



```

delay(1500);

Serial.println("Attaching motors!");
esc1.attach(ESC1_PIN);
esc2.attach(ESC2_PIN);
esc3.attach(ESC3_PIN);
esc4.attach(ESC4_PIN);
Serial.println("Turn on the power!");

// init servos with zero value
Serial.println("Calibrating high. Wait for 2 seconds and press any key!");
Serial.available();

// Wait for input
command_throttle(MAX_SIGNAL);

delay(5000);

Serial.println("Calibrating Low!");
command_throttle(MIN_SIGNAL);
}

void loop() {

LeerMPU6050();

loop_timer= micros();
tiempo_ejecucion = (timePrev - loop_timer);// dividimos entre 1000 para obtener segundos para calcular ang
ulos

timePrev = micros();

```



```

// Mirar si hay comando
if (command_throttle) {

    Pitch = map(PulsoPitch,1046,2000,-300,300);
    Roll = map(PulsoRoll,1046,2000,-300,300);
    Yaw = map(PulsoYaw,1046,2000,-300,300);
    Modulador();
    PID_ang();

    //command_throttle(PulsoPotencia);

}

/*Serial.print("Pulsothrottle = ");Serial.print(PulsoPotencia); Serial.print(" us.");
Serial.print("| PulsoPitch = ");Serial.print(PulsoPitch); Serial.print(" us.");
Serial.print("| PulsoRoll = ");Serial.print(PulsoRoll); Serial.print(" us.");
Serial.print("| PulsoYaw = ");Serial.print(PulsoYaw); Serial.println(" us.");*/

//Serial.print("| PID Roll = ");Serial.print(Pid_Roll_salida); Serial.print(" us.");
Serial.print("| Motor 1 = ");Serial.print(esc1.readMicroseconds()); Serial.print(" us.");
Serial.print("| Motor 2 = ");Serial.print(esc2.readMicroseconds()); Serial.print(" us.")
Serial.print("| Motor 3 = ");Serial.print(esc3.readMicroseconds()); Serial.print(" us.");
Serial.print("| Motor 4 = ");Serial.print(esc4.readMicroseconds()); Serial.print(" us.");
Serial.print("| Modo vuelo = ");Serial.print(PulsoModo); Serial.println(" us.");

}

```

En esta pestaña principal llamada “Main_Proyecto_DronAlberto” es donde va ocurrir la gran parte del programa aquí es donde declaramos las librerías donde nos vamos apoyar, en nuestro caso son las siguientes:

- Servo.h
- Wire.h
- MPU6050.h
- I2Cdev.h
- EnableInterrupt.h

La librería *servo.h* nos ayudará a la hora de controlar los motores gracias a esta potente librería y todo su motor que lleva detrás hace posible que declarando los motores con su clase *Servo* seamos capaces de mandar y leer un pulso en microsegundos hacia los motores simplificando nuestra tarea a la hora de calcular un pulso PWM.

La librería *Wire.h* nos brinda la comunicación entre Arduino NANO y el sensor MPU6050.

La librería *MPU6050.h* gracias a esta librería podemos simplificar enormemente nuestro proyecto gracias a su autor [Jeff Rowberg](#) podemos obtener datos de inclinación, aceleración etc... simplemente llamando funciones de esta librería para luego poder ser procesados mediante fórmulas.

La librería *I2Cdev.h* es un protocolo síncrono. I2C usa solo 2 cables, uno para el reloj (SCL) y otro para el dato (SDA) es un bus muy usado en la industria, principalmente para comunicar microcontroladores. La principal característica es que utiliza dos líneas para transmitir la información: una para los datos(SDA) y otra para la señal de reloj(SCL).

La librería *EnableInterrupt.h* gracias a esta librería vamos a poder controlar los pulsos que nos manda el mando (emisor) dentro de esta fase del proyecto a la llamada de los pulsos generado por nuestra función cabe destacar que vamos a utilizar un tipo de variable no tan común como puede ser una variable global, en este caso la variable será del tipo *volatile* no es un tipo de variable, Eso significa que la variable en cuestión, debe ser almacenado de un cierto modo que evite algunos problemas raros que pueden surgir cuando una variable puede ser cambiada por una interrupción o por el programa (algo que no ocurre en nuestro ejemplo).

6.2 CALIBRACIÓN MPU6050

```
void iniciar_MPU6050(){

    Wire.beginTransmission(MPU6050_address);

    Wire.write(0x6B); //PWR_MGMT_1 registro 6B hex
    Wire.write(0x00); //00000000 para activar
    Wire.endTransmission(true);

    Wire.beginTransmission(MPU6050_address);

    Wire.write(0x1B); //Configuracion_giroscopio registro 1B hex
    Wire.write(0x08); //00001000: 500dps
    Wire.endTransmission(true);

    Wire.beginTransmission(MPU6050_address);

    Wire.write(0x1C); //Configuracion_acelerometro registro 1C hex
    Wire.write(0x10); //00010000: +/- 8g
    Wire.endTransmission(true);

    Wire.beginTransmission(MPU6050_address);

    Wire.write(0x1A); // LPF registro 1A hex
    Wire.write(0x04); //256Hz(0ms):0x00 - 188Hz(2ms):0x01 - 98Hz(3ms):0x02 - 42Hz(4.9ms):0x03 - 20
    Hz(8.5ms):0x04 - 10Hz(13.8ms):0x05
    Wire.endTransmission();

}

void CalibrarGiroscopio(){

    if(gyro_error == 0){
        //Calibrar giroscopio

        //MPU_6050();

        Wire.beginTransmission(MPU6050_address); //Empezamos la comunicacion
        Wire.write(0x43);
        Wire.endTransmission(false);

        Wire.requestFrom(MPU6050_address, 6, true); //Pedimos 6 bytes a partir de la direccion hex 0x43(mirar
        dataSheet)
    }
}
```

```

gx = Wire.read() << 8 | Wire.read(); //LSB sensitivity Datasheet MPU 6050 para escala de +/- 8g
gy = Wire.read() << 8 | Wire.read();
gz = Wire.read() << 8 | Wire.read();
for(cal_int =0; cal_int < 3000; cal_int++){

    gyro_x_cal = gyro_x_cal + gx;
    gyro_y_cal = gyro_y_cal + gy;
    gyro_z_cal = gyro_z_cal + gz;
    delay(1000);
}

    gyro_x_cal = gyro_x_cal / 3000; //valor medio de las 3000 muestras
    gyro_y_cal = gyro_y_cal / 3000;
    gyro_z_cal = gyro_z_cal /3000;

}
}
pitchGyro = (gx - gyro_X_cal) / 65.5; // 65.5: si leo 65.5 en raw, significa que gira a 1°/s
rollGyro = (gy - gyro_Y_cal) / 65.5;
yawGyro = (gz - gyro_Z_cal) / 65.5;

gyro_roll_input = rollGyro;
gyro_pitch_input = pitchGyro;
gyro_yaw_input = yawGyro;
}

void CalibrarAcelerometro(){

if(acc_error == 0){
    //Calibrar acelerometro
    for(cal_int =0; cal_int < 3000; cal_int++){

        Wire.beginTransmission(0x68);
        Wire.write(0x3B); //Ask for the 0x3B register- correspond to AcX
        Wire.endTransmission(false);

```

```

Wire.requestFrom(MPU6050_address,6,true);

ax=(Wire.read()<<8|Wire.read())/4096.0;
ay=(Wire.read()<<8|Wire.read())/4096.0;
az=(Wire.read()<<8|Wire.read())/4096.0;

angle_pitch_acc_cal = angle_pitch_acc_cal + ((atan(-1*(ax)/sqrt(pow((ay),2) + pow((az),2)))*rad_to_deg))
;
angle_roll_acc_cal = angle_roll_acc_cal + ((atan((ay)/sqrt(pow((ax),2) + pow((az),2)))*rad_to_deg));

if(cal_int == 2999){
  angle_pitch_acc_cal = angle_pitch_acc_cal / 3000; //valor medio de las 3000 muestras
  angle_roll_acc_cal = angle_roll_acc_cal / 3000;
  acc_error=1;
}
}
}
}

```

En este apartado vamos a hacer todos los cálculos en cuanto a calibración de nuestro sensor MPU6050 de tal modo que su programación sea fiable en cuanto a su entorno físico con esto quiero decir que cualquier error que tengamos en cuanto a colocación e inclinación de la placa lo podremos corregir, esta pestaña nos enlazara con los pids.

7. MODOS DE VUELO

Para nuestro proyecto hemos desarrollado dos métodos de vuelo uno pensado mas para un dron de carreras llamado modo 'ACROBÁTICO' y otro modo pensado en grabaciones y fotografías profesionales modo 'HORIZON' en estos dos modos el sensor MPU6050 actuara sobre el dron, pero no con todas sus variables.

7.1 MODO ACROBÁTICO

En este modo de control solo utilizaremos las lecturas de velocidad angular calculadas con el sensor MPU6050. La velocidad la vamos a medir en ‘°/s’, es decir, cuantos grados rota cada eje por segundo. Si por ejemplo uno de nuestros ejes da una vuelta completa en un segundo, la velocidad será de 360°/s.

Vamos a ponernos en situación y hagamos un ejemplo. Si el eje roll de nuestro dron rotara por cualquier razón, porque uno de los motores tiene más potencia, porque hay viento... el sensor MPU60 tendrá que contrarrestar esta desviación actuando sobre los motores correspondientes (¡OJO! no hablamos de estabilización sobre el eje 0°). habría que actuar sobre los motores de la izquierda acelerándolos, y sobre los motores de la derecha decelerándolos. De esta forma conseguiríamos corregir el efecto de la perturbación que ha hecho que nuestro dron rotara en su eje roll. Pero cuidado, en modo acrobático el dron no volverá a su posición inicial, simplemente compensaremos la rotación hasta detener el dron en la posición que le hemos mandado. Esto es debido a que únicamente estamos utilizando como referencia la velocidad de rotación de los ejes.

Ahora bien, ¿cuánta potencia y durante cuánto tiempo hay que acelerar cada motor para contrarrestar estas perturbaciones? En otras palabras, ¿Cómo hacemos que nuestro sensor MPU6050 sea capaz de mantener el dron estático en el aire ? Para esto tenemos los PID.

Si hiciéramos una interlocución entre el dron y nosotros tendríamos el siguiente flujo (Ver figura 52)

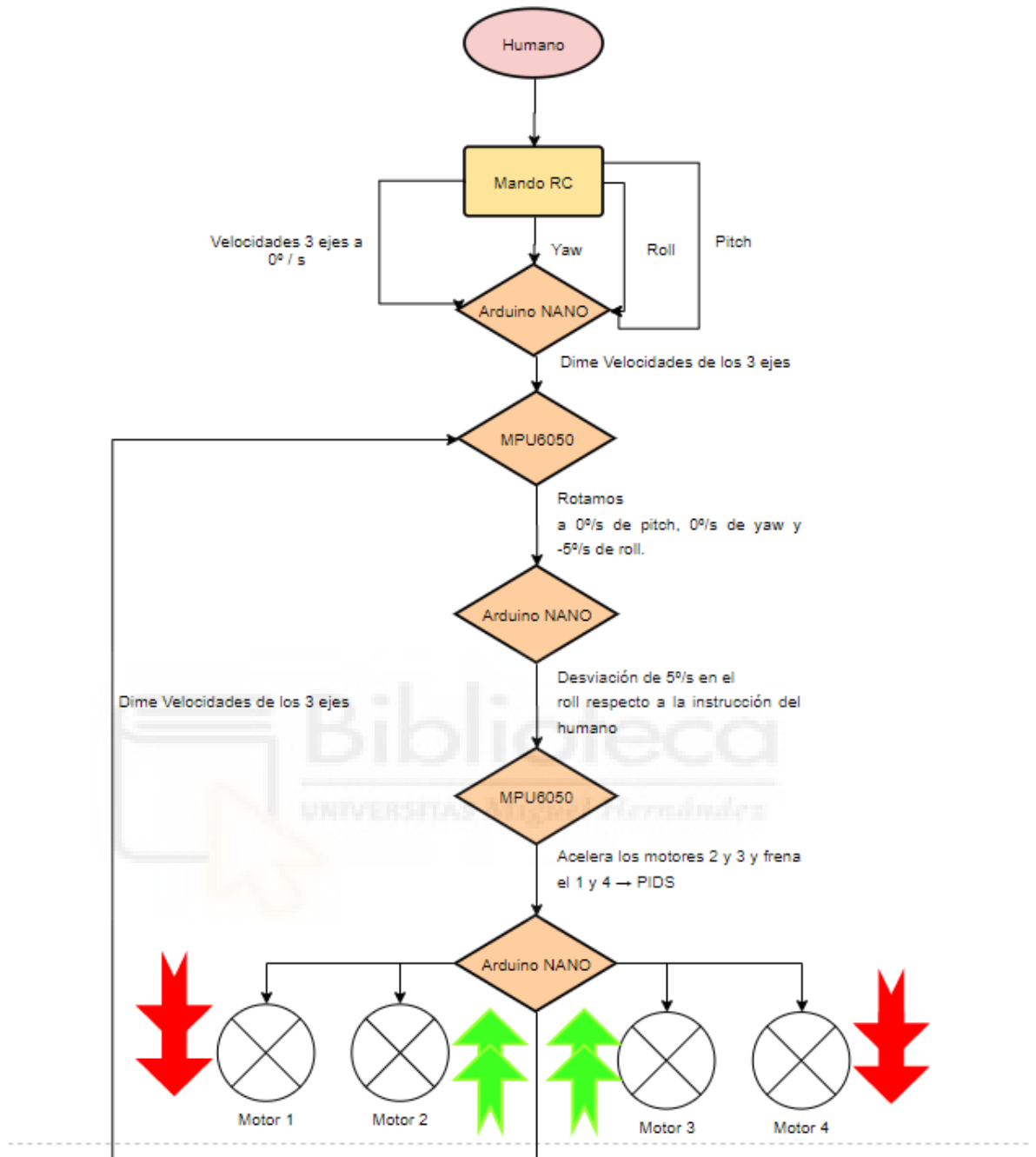


Figura 59: Diagrama de flujo modo acrobático.

Como vemos, los encargados de calcular el error de cada ángulo y actuar en consecuencia son los PID. Los PID son una parte fundamental de nuestro dron por lo que es necesario saber algo de teoría sobre ellos. Hay infinidad de información en todos los idiomas sobre sistemas de control basados en PIDs y sus aplicaciones en dron, Arduino cuenta con una librería para PID, pero recomiendo no utilizarla e intentar entender el funcionamiento de estos controladores programándolos nosotros mismos (son muy sencillos). El objetivo de un PID es conseguir un error entre la consigna de velocidad y la velocidad real de 0°/s (metros,

grados... según la aplicación), es decir, que la velocidad de rotación real sea igual a la consigna que llega desde el mando en todo momento. Pongamos como ejemplo uno de los ejes de nuestro dron, por ejemplo el eje Pitch:

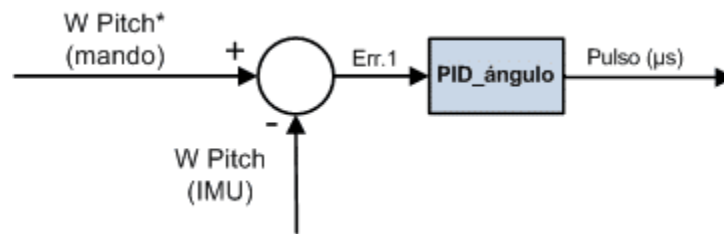


Figura 60: Generar PID.

Lo primero que hace esta estructura de control es comparar la referencia de velocidad angular que nos llega desde el mando, en la imagen ‘W Pitch* (mando)’, con la lectura que recibimos del sensor MPU6050, en la imagen ‘W Pitch (IMU)’. Haciendo la resta de estas dos señales conseguimos un valor de desviación o error en nuestro eje pitch, en la imagen, señal Err.1

Por ejemplo, si desde el mando nos llega la consigna de 0°/s de pitch y el sensor MPU6050 está leyendo que la velocidad real es de +10°/s, la variable ‘Err.1’ tomará el valor -10°/s, es decir, tenemos un error en la velocidad del eje pitch de -10°/s. Este podría ser debido al viento o una mala distribución de pesos en el dron, nuestro dron se está inclinando en una dirección. **El objetivo de nuestro PID será hacer que este error sea siempre de 0.**

$$\text{Err.1} = w \text{ Pitch* (mando)} - w \text{ Pitch (IMU)} = 0 \text{ rad/s} - 10\text{rad/s} = -10\text{rad/s}$$

Si por el contrario desde el mando nos llega una referencia de 10°/s, es decir, queremos que nuestro dron se incline y desplace en su eje pitch, el funcionamiento sería similar. El error pasaría a ser de 10°/s y el PID aceleraría los correspondientes motores hasta aumentar la velocidad a 10°/s, es decir, hasta hacer el error cero.

$$\text{Err.1} = w \text{ Pitch* (mando)} - w \text{ Pitch (IMU)} = 10 \text{ rad/s} - 0 \text{ rad/s} = + 10\text{rad/s}$$

Tras acelerar los motores, el dron empezará a rotar en el sentido que hayamos indicado hasta alcanzar la velocidad deseada, momento en el que el error bajará a 0°/s y habrá finalizado la operación:

$$\text{Err.1} = w \text{ Pitch}^* (\text{mando}) - w \text{ Pitch (IMU)} = 10 \text{ rad/s} - 10 \text{ rad/s} = 0 \text{ rad/s}$$

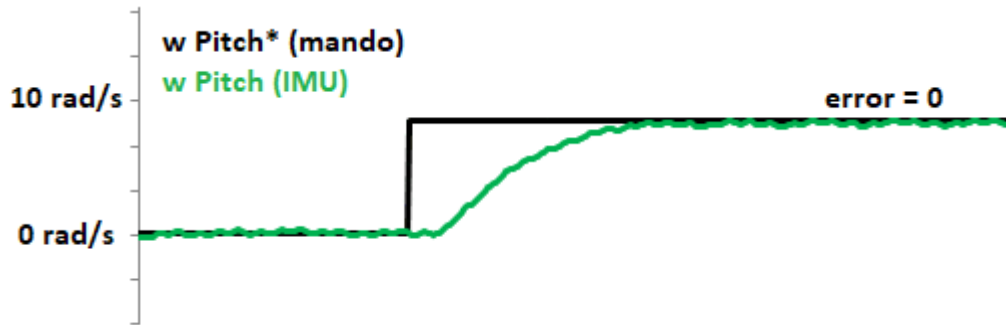


Figura 61: Corrección de error velocidad.

Este error 'Err.1' lo recibe el PID y genera una salida 'Pulso (us)' en función de los parámetros K_p , K_i y K_d que hayamos establecido. Simplemente cogemos el error y lo multiplicamos por estos valores, haciendo más o menos agresivo el control de estabilidad de nuestro dron.

La parte K_p (PID) es proporcional al error, simplemente multiplicamos ambos términos. Si por ejemplo tenemos un valor de K_p de 10 y tenemos un error de 10° :

$$10 * 10^\circ = 100\text{us}$$

Si el motor estaba girando con un PWM de 1.5ms, aceleraría hasta $1.5\text{ms} + 0.1\text{ms} = 1.6\text{ms}$.

La parte K_i (PID) es proporcional al error que vamos acumulando en cada ciclo. Cogemos el error actual y lo multiplicamos por el término K_i , pero en cada nuevo ciclo de control sumamos el valor obtenido en el ciclo anterior. De esta forma conseguimos que el error en régimen permanente sea de 0.

La parte Kd (PID) es proporcional a la diferencia de error entre ciclos. Sirve para suavizar la respuesta del control.

A continuación, vamos a esquematizar todo lo que hemos visto anteriormente pero ahora de forma gráfica (Ver figura 55).

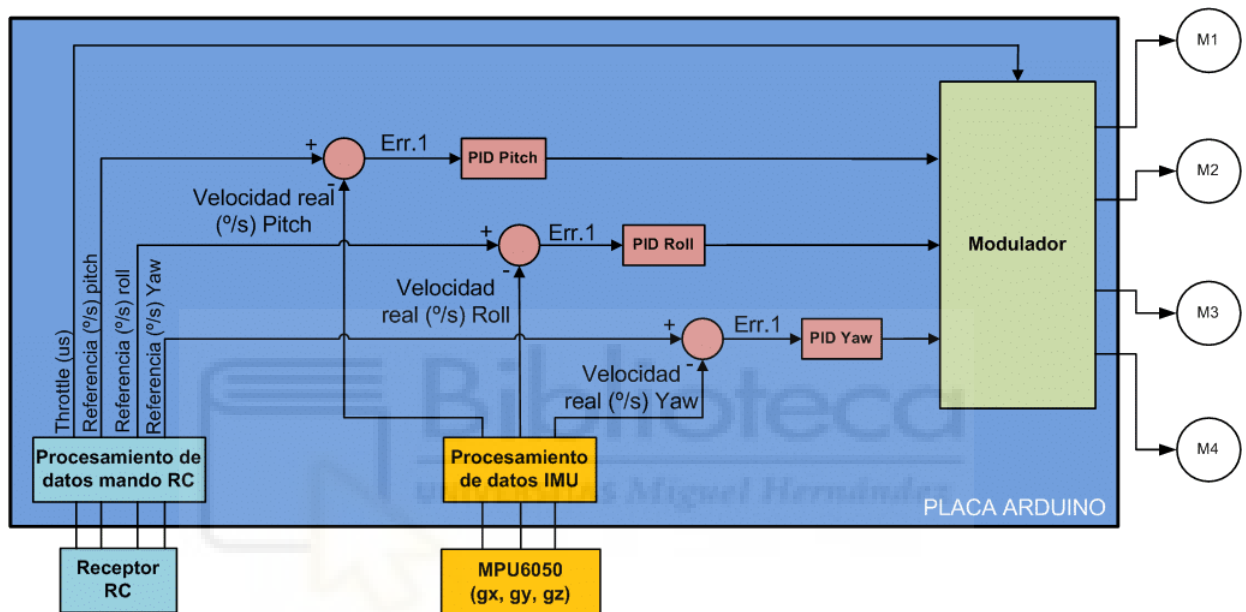


Figura 62: Ordenes modo acrobático.

7.2 MODO HORIZON

Este modo necesita de dos *PID* simultáneos por cada eje, además de lecturas de velocidad de rotación y aceleración proporcionados por el sensor *MPU6050* a partir de los cuales calcular el ángulo de inclinación de cada eje en grados ($^{\circ}$). La ventaja que tiene este modo de vuelo es que el dron es completamente estable y por lo tanto mucho más fácil de controlar. Ocurre todo lo contrario que en el modo acrobático, cuando soltemos alguna de las palancas del mando *RC* el dron volverá automáticamente a su posición de 0° de inclinación. La instrucción que mandamos desde el mando es de grados de inclinación ($^{\circ}$), no de velocidad ($^{\circ}/s$) como en el caso acrobático.

El funcionamiento de control es muy simple. Si detectamos una inclinación, ordenamos al dron que gire en dirección contraria a una velocidad determinada hasta contrarrestar esta inclinación:

Secuencia: Esta secuencia se ejecuta cada 5ms (200Hz)

- Arduino solicita información de velocidad angular y aceleración de los tres ejes al sensor MPU6050.
- El sensor manda sus lecturas y Arduino realiza los cálculos de velocidad angular (°/s) e inclinación (°).
- Con las lecturas de inclinación y las consignas que recibimos desde el mando, el primer PID calcula el error de inclinación y genera la consigna de velocidad para contrarrestarla.
- El segundo PID toma esta consigna del lazo exterior y con la lectura de velocidad de la del sensor MPU6050, genera la salida en microsegundos para enviar a los motores. Los motores aceleran y contrarrestan la desviación de velocidad, que a la vez está contrarrestando la desviación en la inclinación.
- Vuelta al paso 1.

Lo primero que hace esta estructura de control es comparar la consigna de inclinación (°) que nos llega desde el mando, en la imagen '*Pitch* (mando)*', con la lectura que recibimos del sensor *MPU6050*, en la imagen '*Pitch (IMU)*'. Fijaos en que en este modo de vuelo el mando fija la consigna de inclinación (°) y no de velocidad de rotación (°/s) como en el modo acrobático. Haciendo la resta de estas dos señales conseguimos en valor de desviación o error en nuestro eje *pitch*, en la imagen, señal *Err.1*. Esta variable se da en grados de inclinación (°), 5°, 10°... la que sea:

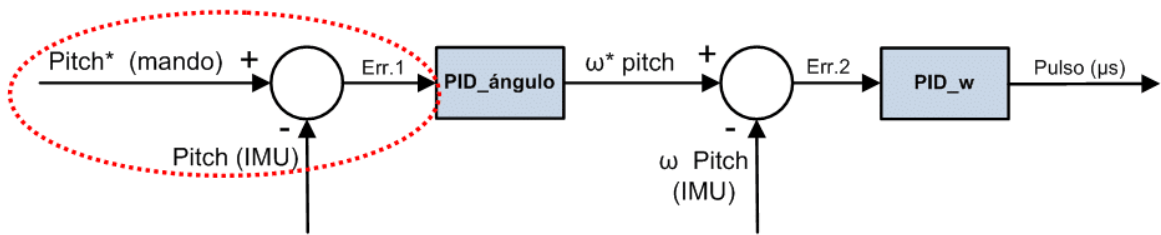


Figura 63: Calculo Err.1.

Esta variable Err.1 pasa por el primer PID (PID_ángulo o PID estable) y genera una salida que se utilizará como referencia para el siguiente lazo o lazo de velocidad (el utilizado en el modo acrobático). Lo que conseguimos con esto es indicar al dron que si está inclinado, tiene que aumentar/reducir la velocidad en los motores y rotar a una determinada velocidad y en dirección contraria a la inclinación para contrarrestarla y volver a la posición inicial de 0°:

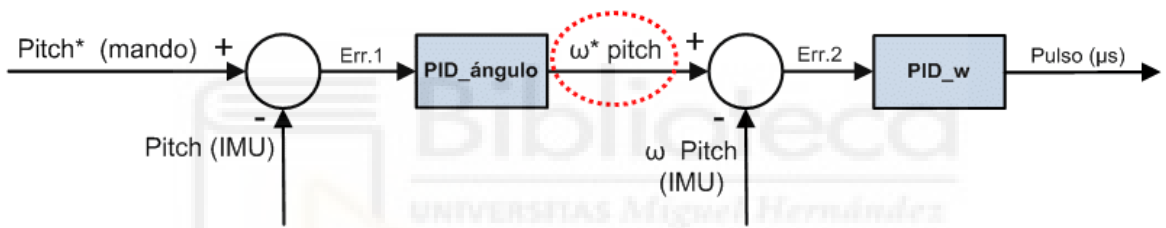


Figura 64: Calculo $\omega * pitch$.

Como hemos dicho, se compara la consigna que nos llega desde el lazo estable, en la imagen 'ω* pitch', con la lectura de velocidad de rotación que recibimos del sensor MPU6050, en la imagen 'ω Pitch (IMU)', para generar el error 'Err.2'. Este error representa la desviación entre la velocidad que necesitamos para contrarrestar la inclinación y la velocidad real de rotación del dron. Finalmente, la variable Err.2 pasa por el PID de velocidad (PID_w) y generamos la salida para el modulador que actuará sobre los motores.

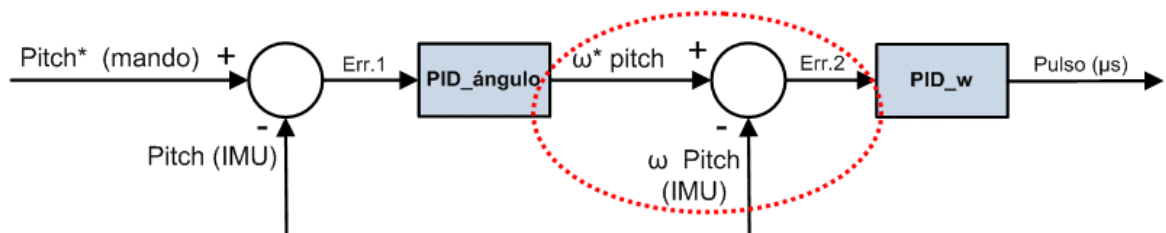


Figura 65: Calculo Err.2.

Cuando el dron comience a girar y se vaya corrigiendo la desviación, tanto las variables Err.1 y Err.2 irán disminuyendo hasta convertirse en 0, momento en el que dron habrá vuelto a su posición inicial y no haya desviación alguna entre la consigna que enviamos desde el mando y la inclinación real.

La siguiente figura muestra la estrategia de control total utilizada en el modo estable representada con bloques. Es parecida a la figura mostrada para el modo acrobático, solo que utilizando un PID más en los ejes pitch y roll (el eje yaw no requiere de otro PID). El primer PID toma la lectura de inclinación ($^{\circ}$) calculada a partir de las lecturas del sensor MPU6050 y la compara con la consigna del mando. Si hay una desviación de inclinación, este primer PID genera una referencia de velocidad para el siguiente lazo, acelerando los correspondientes motores y contrarrestando la inclinación. El segundo PID controla la velocidad a la que rota el dron mientras contrarresta la inclinación. Es una estrategia bastante intuitiva:

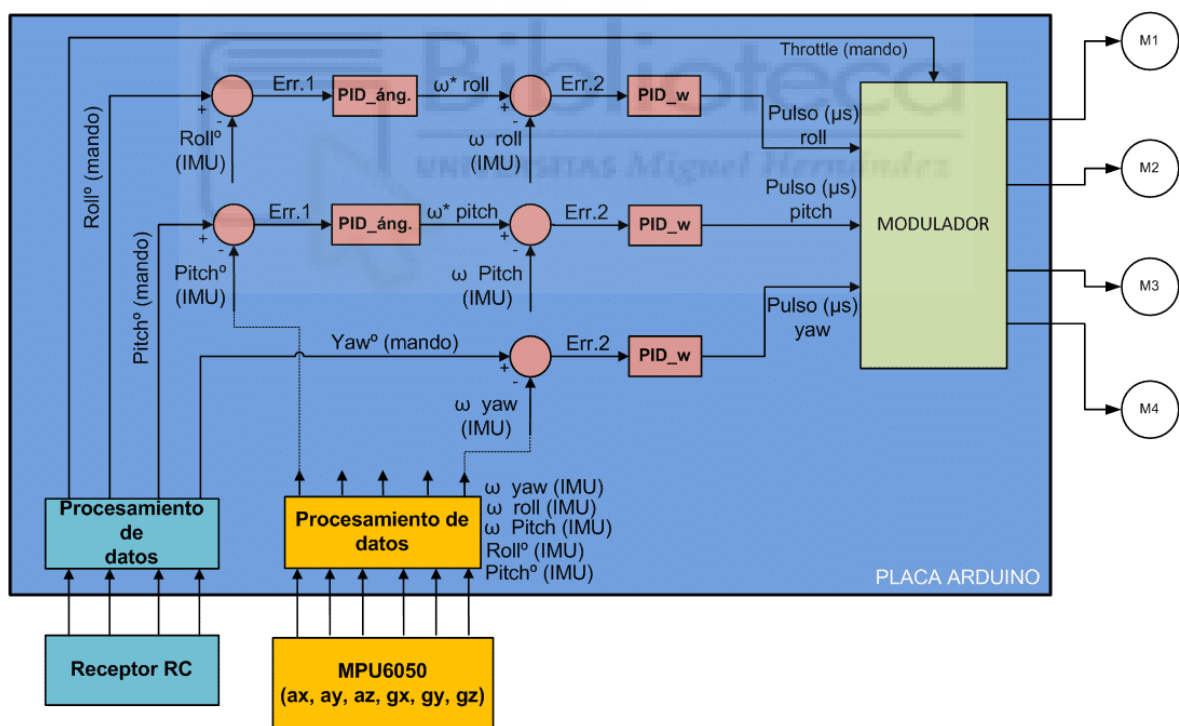


Figura 66: Ordenes modo estable.

Utilizando este método conseguiremos que, al soltar la palanca del mando, el dron vuelva automáticamente a su posición inicial de 0° sin tener que mandar una consigna de velocidad negativa para contrarrestar la inclinación. El funcionamiento de los dos métodos es evidente, el acrobático simplemente compensa la rotación ($^\circ/s$), mientras que el estable compensa la inclinación ($^\circ$).



8. CONCLUSIÓN

En este proyecto se ha determinado el montaje, programación y pruebas de vuelo de un dron instalándose un sistema de emergencia Lora el cual nos brinda esta gran ventaja de control para el sistema de emergencias, en este proyecto nos hemos centrado en el montaje tanto electrónico como físico del dron, además de las múltiples pruebas para conseguir la estabilización del dron tanto en modo acrobático como el modo horizon. Para ello, se ha empleado un circuito similar en recepción y emisión, compuesto de un transmisor LoRa y un Arduino UNO

De los resultados de cada fase sacamos la conclusión:

- En este tipo de proyectos es de vital importancia replantear bien los elementos a distribuir tanto sus pesos como su tamaño ya que nos será bastante más sencillo calibrar el MPU6050 con un reparto de pesos lo más equilibrado posible, la desventaja en este proyecto es la multitud de placas de diferentes tamaños, esto nos obliga a tener que distribuir las placas por la superficie del dron siendo muy difícil el poder apilarlas una encima de otra ganando en espacio y conexionado. Hoy en día existen placas específicas para drones donde viene integrado todo lo necesario para montar el dron lo que supone una ventaja inmensa.
- Hemos calibrado el MPU6050 correctamente dándonos unas medidas de ángulo correctas en todos los movimientos, gracias al filtro implementado hemos reducido todas las vibraciones del dron que pudiesen afectar a la respuesta del sensor, obteniendo una respuesta limpia. después de muchas pruebas en nuestro sistema de calibración “Casera” nos hemos encontrado con algunos inconvenientes. El banco de pruebas no es lo suficientemente bueno como para hacer demostraciones reales para ajustar los pids esto es debido que el máximo rango de inclinación es 36°. Otro problema es el aire generado debajo del dron, este es impulsado hacia abajo y luego parte de él sube hacia arriba haciendo que el MPU6050 tenga lecturas erróneas, por ese motivo el dron se descontrola, para lidiar este paso se necesitaría una jaula de pruebas profesional en la cual el dron queda suspendido y libre en sus dos ejes.

- Ajustando los parámetros de LoRa, estamos ante una tecnología inalámbrica con una cobertura de hasta 10 km en línea recta. Esto pensando lo pequeño que es el dispositivo y los consumos que tiene podemos llegar a la conclusión que es una tecnología con muchas prestaciones.
- El sistema de emergencia ha funcionado correctamente el apagado inmediato es seguro y con una latencia muy baja.

9. PRESUPUESTO

En este apartado vamos a detallar los elementos necesarios que hemos utilizado en este proyecto:

| Ítem | Unidades | Precio Unidad | Total |
|---------------------------------|----------|---------------|----------------|
| Arduino Nano | 1 | 20€ | 20€ |
| Arduino Pro Mini | 1 | 12€ | 12€ |
| Arduino Mini | 1 | 12€ | 12€ |
| RF-LoRa-868-SO | 2 | 7,75€ | 15,5€ |
| 4 ESC Flycolor X-Cross BL32 36A | 1 | 12,90€ | 12,90€ |
| PDB | 1 | 8,59€ | 8,59€ |
| 4 Motores DX2205 | 1 | 35€ | 35€ |
| Chasis | 1 | 20€ | 20€ |
| MPU6050 | 1 | 9,21€ | 9,21€ |
| FTDI | 1 | 5,99€ | 5,99€ |
| Pulsador | 2 | 0,65€ | 1,30€ |
| Buzzer | 1 | 0,85€ | 0,85€ |
| Relé | 1 | 7,30€ | 7,30€ |
| Batería | 1 | 24,99€ | 24,99€ |
| Hélices | 1 | 5€ | 5€ |
| Mando RC | 1 | 47€ | 47€ |
| Receptor video FPV | 1 | 89€ | 89€ |
| Cámara on board | 1 | 15€ | 15€ |
| Emisor video | 1 | 7,75€ | 7,75€ |
| | | Total | 304,38€ |

Tabla 6: Presupuesto del Proyecto.

10. BIBLIOGRAFIA

[1]. "RF-LoRa-868-SO Datasheet". Disponible en la web:

<https://www.digikey.es/product-detail/es/rf-solutions/RF-LORA-868-SO/RF-LORA-868-SO-ND/5291460> Accesible a fecha: 02/05/2021

[2]. "Arduino NANO". Disponible en la web:

<https://store.arduino.cc/arduino-nano> Accesible a fecha: 02/05/2021

[3]. "MPU6050 Datasheet". Disponible en la web:

<https://datasheet4u.com/datasheet-parts/MPU6050-datasheet.php?id=735134>
Accesible a fecha: 02/05/2020

[4]. "PIDS Arduino dron". Disponible en la web:

<https://github.com/lobodol/drone-flight-controller>
Accesible a fecha: 12/05/2020

[5]. Improving LoRa Signal Coverage in Urban and Sub-Urban Environments with UAVs. Disponible en la web:

<https://ieeexplore.ieee.org/document/8730598> Accesible a fecha: 05/05/2021

[6]. LoRa QoS Performance Analysis on Various Spreading Factor. Disponible en la siguiente web:

<https://ieeexplore.ieee.org/document/8605471> Accesible a fecha: 05/05/2021

[7]. "Low Cost LoRa Gateway". Disponible en la web:

<https://github.com/CongducPham/LowCostLoRaGw> Accesible a fecha: 13/05/2021

[8]. "Antenna length for 868 and 433 MHz". Disponible en la web:

<https://www.thethingsnetwork.org/forum/t/antenna-length-for-868-and-433-mhz/5378> Accesible a fecha: 13/05/2021

[9]. Material. Disponible en la web: <https://iha-race.com/> Accesible a fecha: 13/05/2021

[10]. "Low Cost LoRa Gateway". Disponible en la web: <https://github.com/jrowberg/i2cdevlib/blob/master/Arduino/MPU6050/MPU6050.h>
Accesible a fecha: 02/03/2021

[11] . “ Tipos de drones Aéreos”. Disponible en la web:
<http://dronespain.pro/tipos-de-drones-aereos/>
Accesible a fecha: 30/06/2021

