UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje e implementación del control de altura de un cuadricóptero

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Fontalba Roca, Marc

Tutor/a: González Sorribes, Antonio

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje e implementación del control de altura de

un cuadricóptero

Documento Nº1: MEMORIA

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Fontalba Roca, Marc

Tutor/a: González Sorribes, Antonio

CURSO ACADÉMICO: 2022 / 2023

Agradecimientos

Quisiera dar las gracias a mis compañeros por estar conmigo estos cuatro años de carrera y en especial a Kerry, Eme, Kai, Abel y Carlos por haber hecho tan agradable mi paso por la universidad.

Quiero agradecer a mi tutor Antonio González Sorribes por alentarme y ayudarme a realizar este proyecto. Y por su excelente desempeño como profesor en la asignatura de robótica aérea que me ha permitido adquirir los conocimientos para impulsar este trabajo.

Por último, quisiera agradecer a la ETSID y la UPV por permitirme acceder al estudio del grado de ingeniería electrónica industrial y automática.

Resumen

El presente proyecto consta de una primera etapa de diseño, fabricación y calibración de un vehículo aéreo no tripulado de tipo cuadricóptero, y una segunda etapa de implementación de un algoritmo de control de altura y orientación en el vehículo aéreo diseñado. La programación de los controles se llevará a cabo sobre una plataforma Arduino, el cual procesará los datos del sensor de altimetría e inclinación para finalmente generar las acciones de control necesarias sobre los rotores. En el desarrollo del proyecto se abordarán temas como la justificación de los componentes empleados, el montaje del dispositivo, el diseño y simulación de los algoritmos de control mediante Matlab, los movimientos básicos de cuadricópteros, el control de motores brushless mediante variadores eléctricos, la lectura y procesamiento de datos de la unidad de medición inercial.

Índice de figuras:

Índice de tablas:

Índice de la memoria:

Lista de acrónimos

BEC: Battery Eliminator Circuit.

CW: Clockwise.

CCW: Counterclockwise.

ESC: Electronic Speed Control.

IMU: Unidad de Medida Inercial.

MTOW: Maximum Take-Off Weight.

PWM: Pulse-Width Modulation.

RPA: Remotely Piloted Aircraft.

UAV: Unmanned Aerial Vehicle.

VANT: Vehículo Aéreo No Tripulado.

1. Objeto

El proyecto realizado tiene por objetivo el diseño y desarrollo de un cuadricóptero, desde una primera etapa correspondiente a la elección de sus componentes, calibración y fabricación, hasta el diseño e implementación de su control de altura y estabilidad. Así mismo, el montaje y programación del prototipo quedan incluidos dentro del alcance de este.

Será necesario definir una serie de objetivos para el correcto cumplimiento del proyecto:

- La realización de un análisis y elección de los componentes adecuados en base criterios específicos como tamaño, peso, tensiones nominales entre otros.
- Llevar a cabo el montaje distribuyendo los componentes de forma que el centro de masas del prototipo quede situado y orientado en el centro de la aeronave.
- La elaboración del diseño de los bloques de control del vehículo aéreo y la comprobación de su correcto funcionamiento. Así como su implementación en el microcontrolador.

2. Antecedentes

2.1 Introducción

Los vehículos aéreos no tripulados (VANT) son todos aquellos vehículos capaces de realizar vuelos sin necesidad de intervención humana a bordo. Dentro de este grupo, destacan las aeronaves tripuladas por control remoto (RPA), cuya característica fundamental es ser controladas por un piloto o estación de forma remota (Rosa, 2020).

Por otro lado, podemos encontrar los robots aéreos que son aquellos VANT capaces de funcionar de manera autónoma para el cumplimiento de una misión (Miranda Colorado, 2020). Es en este último grupo es donde podemos situar el dispositivo desarrollado en el presente proyecto.

Los VANT, iniciaron su desarrollo en el ámbito bélico durante la primera guerra mundial, posteriormente hasta la actualidad han ido evolucionando y abriéndose paso en otros ámbitos de aplicación. Actualmente, esta tecnología se encuentra en auge debido a sus numerosos usos y utilidad siendo un sector del que se espera un crecimiento importante en los próximos años.

2.2 Motivación del proyecto

El proyecto nace como propuesta a aunar, profundizar y poner en práctica los conocimientos adquiridos en la asignatura de robótica aérea, impartida por primera vez el presente año en la mención de robótica del grado de ingeniería electrónica industrial y automática. Puesto que esta aborda temas como la estructura, electrónica y control de las VANTs. Por lo tanto, se planteó la posibilidad de realizar la fabricación y programación de un cuadricóptero desde cero.

2.3 Historia

Los primeros registros históricos que se encuentran sobre vehículos aéreos no tripulados datan a mediados del siglo XIX. Sin embargo, donde se consigue un gran desarrollo inicial de los VANT, concretamente en el campo de las RPAs, es en el siglo XX a causa de los conflictos bélicos de este periodo y las inversiones en armamento derivadas de estos. Es en esta primera etapa, donde los vehículos aéreos no tripulados se emplearon principalmente para reconocimiento y enfrentamiento.

Realizando un breve recorrido histórico, el primer avión moderno no tripulado, fue el Hewitt-Sperry, diseñado durante la primera guerra mundial, concretamente en 1916 (Miranda Colorado,2020). Posteriormente, en la década de los 40, el uso de armas por control remoto se empieza a asentar en los enfrentamientos bélicos, como es el caso del GB-4, siendo esta la primera arma guiada a control remoto trasmitida por televisión. Más adelante, en la década de los 60 y 70 se hacen avances en los sistemas de guiado, la incorporación de sistemas electrónicos, así como el uso de cámaras, incluso se consigue aeronaves con una autonomía de más de 24 horas. Con la llegada de la década de los 80 y 90, el desarrollo de los sistemas microelectrónico permite una reducción del tamaño y peso de los VANT acercándolos más a los que nos encontramos en la actualidad (Delgado, 2016).

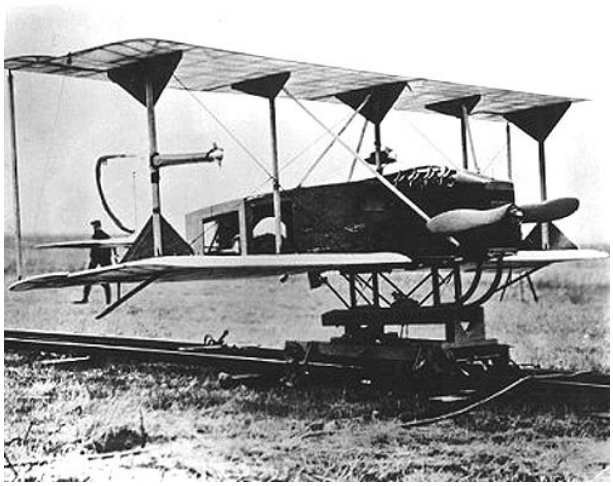


Figura 1: Avión Hewitt-Sperry

Nota. De «Historia de los drones», V. Delgado, 2016, El Drone. De dominio público.

Actualmente, si bien es cierto que el ámbito bélico continúa siendo uno de los campos donde más se utilizan, han aparecido campos donde las VANT resultan una herramienta útil, como en la logístico para aplicaciones de reparto y gestión de inventario, como en salvamento para misiones de búsqueda y rescate, para control y prevención de incendios forestales, o para capturar imágenes de eventos entre muchas otras aplicaciones.

3. Estudio de necesidades

En este apartado, se analizarán las necesidades que requerirá el dispositivo para su correcto funcionamiento, por ello, se recopilaran los diferentes aspectos técnicos, de gestión y económicos necesarios.

3.1 Aspectos técnicos

Respecto a las necesidades técnicas, cuadricóptero requerirá de los siguientes componentes electrónicos:

- Una batería capaz de alimentar tanto a los motores como al microcontrolador.
- Cuatro motores iguales que cumplan el requisito de entre todos ejercer al menos una fuerza de propulsión equivalente a dos veces el peso de la aeronave.
- Cuatro controladores de velocidad (ESC) para controlar los actuadores, además deberán poder soportar la corriente que necesitarán los motores.
- Una unidad de mediada inercial (IMU) para conocer los valores de los ángulos Roll, Pitch, Yaw.
- Un altímetro para conocer la altura de la aeronave y cerrar el bucle del control de altura.

- Un interruptor capaz de asegurar encender y apagar el dispositivo con seguridad.

Continuando con los requisitos técnicos, pasamos a los relativos a la estructura:

- Un fuselaje de un material ligero, resistente y en forma de X.
- Cuatro hélices de las mismas características dos para girar en sentido horario (CW) y otras dos para sentido antihorario (CCW). Deberán poder junto con los motores ejercer un empuje equivalente al doble del peso de la aeronave.
- Un peso total del dispositivo no superior a un kilogramo.

Por lo que respecta a las prestaciones y modos de funcionamiento:

- Ha de tener implementado un control de altura capaz de seguir las consignas establecidas con un error de posición 0.
- Ha de tener implementado un control de estabilidad capaz de asegurar que el dispositivo quede en condiciones donde sus ángulos de Roll y Pitch similares a 0, modo de vuelo hovering.

3.2 Aspectos de Gestión

En cuanto a los aspectos relacionados con la legislación de UAV, el prototipo deberá cumplir con los requisitos de los cuadricópteros de clase C1, definidos en el BOE 2019/945:

- No deberá superar los 900 gramos de masa máxima al despegar (MTOW).
- La altura máxima de vuelo deberá estar limitada a no más de los 120 metros de altura.
- La alimentación del dispositivo ha de ser eléctrica.

Otros aspectos que considerar para cumplir con los requerimientos de un vehículo aéreo tipo C1 pero que escapan al alcance del proyecto son los siguientes:

- Deberá tener un número de serie único que permita su reconocimiento.
- Deberá contar con un sistema de geoconsciencia.
- Deberá contar con un aviso de batería baja y una estación de control.

Por otro lado, deberá ceñirse a la siguiente normativa europea 2014/30/UE sobre la compatibilidad electromagnética, la directiva 2014/53/UE sobre la comercialización de equipos radioeléctricos y la directiva 2011/65/UE sobre restricciones de uso de sustancias peligrosas en aparatos electrónicos.

3.3 Aspectos Económicos

Respecto a las limitaciones económicas, el prototipo deberá cumplir con los siguientes requisitos:

- El cómputo total del coste material de los componentes no deberá superar los 350 €.

3.4 Normativa Vigente

La normativa de obligado cumplimiento seguido en el proyecto:

- Marcado CE (BOE nº. 246, de 11 de octubre de 2008).
- Reglamentos Europeos RE-2019/947 y RD 2019/945.
- Real Decreto 1036/2017.
- Directiva 2014/30/UE.
- Directiva 2011/65/UE.
- Directiva 2014/53/UE.

4. Soluciones alternativas y justificación adoptada

En este apartado, se muestran las diferentes propuestas de solución para el proyecto y una justificación de la alternativa adoptada. La justificación, se llevará a cabo mediante tablas que valorarán los diferentes aspectos relevantes de cada componente, los cuales se puntuaran del 1 al 3, el que mayor puntuación tenga será el seleccionado como solución.

Por lo que respecta al fuselaje, se ha valorado siguiendo cuatro criterios: el material, el tamaño, el peso y el precio. En cuanto al material, se han tenido en cuenta características como resistencia, rigidez y ligereza, de forma que a mejores características mejor valoración. Sobre el tamaño, el peso y el precio, se ha considerado que le mejor caso es cuando más bajo son sus valores.

Tabla 1: Alternativas de fuselaje.

Fuselaje	Material	Tamaño	Peso	Precio	Total
F450	1	2	1	3	7
QAV250	3	3	3	2	11
QL5 V2	3	1	2	1	7

Para la selección de los motores, se ha valorado la eficiencia, peso y precio. En cuanto al peso y el precio, la valoración se ha realizado igual que en el caso anterior. Con respecto a la eficiencia, se ha valorado como mejor el que presente una mayor eficiencia, es decir, relación potencia/empuje.

Tabla 2: Alternativas de motor.

Motor	Eficiencia	Peso	Precio	Total
DX2306	1	1	1	3
BETAFPV 1103 11000KV	2	2	2	6
DX2205	3	3	3	9

Para la batería, se han considerado la capacidad y la velocidad de descarga para la elección del componente, de forma que a mayor valor de estas mejor será su ponderación. Además, se ha considerado el peso y el precio siguiendo con los criterios anteriores.

Tabla 3: Alternativas de batería.

Batería Lipo	Capacidad	Velocidad de descarga	Peso	Precio	Total
HBS 3S 11.1V 50C 6000mAh	3	3	1	1	8
KingDuo 3S 11.1V 30C 2200mAh	2	1	3	2	8
RoaringTop 3S 11.1V 35C 2200mAh	2	2	2	3	9

En cuanto a los ESC, a parte del peso y el precio, se ha valorado si disponen de un BEC integrado y si el amperaje se acerca a los que requerirán los motores.

Tabla 4: Alternativa de ESC.

ESC	BEC	Amperaje	Peso	Precio	Total
EMAX BLHELI 12ª	3	3	3	3	12
Drefeify 20ª	-	1	1	1	3
Soapow 12ª	3	3	2	2	10

En la selección de la IMU y el giroscopio, se han tenido en cuenta criterios anteriores. Además, se ha considerado la precisión del sensor, a mayor precisión mejor se valora.

Por otro lado, también se ha valorado positivamente que el sensor tenga implementado un filtro Kalman a bajo nivel y sea capaz de medir posición angular.

Tabla 5: Alternativas de IMU.

IMU	Precisión	Filtro Interno	Peso	Precio	Total
WitMotion WT31N	2	3	1	2	8
Adafruit BNO055	3	3	2	1	9
MPU-6050	1	-	3	3	7

Tabla 6: Alternativa de giroscopio.

Giroscopio	Precisión	Peso	Precio	Total
WT61	2	2	2	6
L3G4200D	1	3	3	7
WT901B	3	1	1	5

5. Descripción detallada de la solución adoptada

A continuación, se detallarán las soluciones adoptadas para el presente proyecto, junto a una explicación de la función que tiene dentro de la aeronave. La elección, se ha llevado a cabo conforme a los criterios del apartado anterior.

Por otro lado, se ha realizado el siguiente organigrama para representar la estructura del dron, así como las relaciones entre sus componentes:

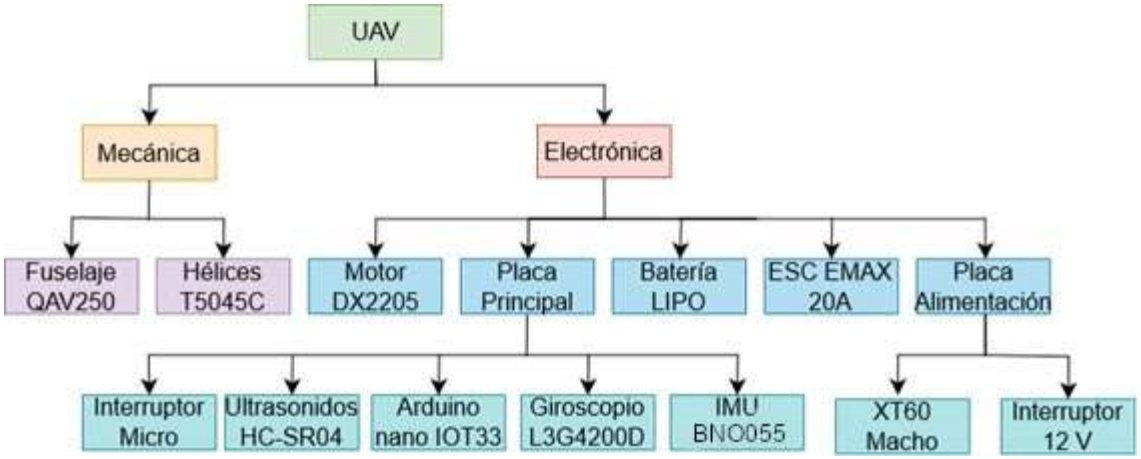


Figura 2: Organigrama del proyecto.

5.1 Conjunto Mecánica

Respecto a la mecánica del dispositivo, está compuesta por dos componentes, el fuselaje y las hélices.

5.1.1 Fuselaje QAV250

El fuselaje es el "cuerpo" de la aeronave, es decir, su principal elemento estructural. En él estarán ubicados todos los demás componentes del UAV y tendrán que volar solidarios a él. Como elemento estructural básico hay que tener en cuenta dos características importantes, el material y la forma.

Respecto al material, debe ser resistente y rígido ya que el fuselaje será el que reciba los impactos en caso de chocar o de un mal aterrizaje. Por otro lado, debe ser ligero para facilitar que el dispositivo vuele. Asimismo, otro factor importante es la resistencia a la fatiga estructural provocada por cargas cíclicas.

En cuanto a su forma, debe tener un tamaño y peso adecuados para poder volar, también deberá cumplir con las restricciones del marco legal. Además, el tipo de aeronave viene determinado por el propio fuselaje, en nuestro caso será el de un cuadricóptero.

Por estas razones se ha seleccionado el modelo QAV250, un fuselaje de cuadricóptero en "X" de dimensiones 250 mm en diagonal, 175 mm de ancho, 130 mm de largo y una masa de 140 g. El material del que está fabricado es de fibra de carbono que encaja con los parámetros mecánicos que requiere el fuselaje.

Figura 3: Fuselaje QAV250.

Nota. De «Catalogo de Amazon», Dilwe QAV250 , (s.f), Amazon. CC-BY-NC.

5.1.2 Hélices T5045C

Las hélices son un elemento esencial en la propulsión del UAV junto con los motores, se encargan de transformar la rotación en movimiento de una masa de aire. Además, tienen un perfil aerodinámico que provoca que exista una diferencia de presiones y velocidades de aire entre las dos caras de las hélices. De esta forma, se consigue la sustentación aérea.

Por otro lado, a la hora de seleccionar las hélices se ha de tener en cuenta su diámetro y sección de paso, puesto que, estos valores afectan a la propulsión de la aeronave. Por ejemplo, el diámetro afecta al consumo y estabilidad, a mayor diámetro mayor será su consumo y estabilidad mientras que en diámetros pequeños el resultado es el opuesto. Ahora bien, por lo que respecta a la sección de paso, esta influye en la eficiencia de las hélices, un menor paso es más eficiente en maniobras de despegue y aterrizaje. Sin embargo, las de paso alto son más indicadas para desplazamientos a altas velocidades.

Respecto a conocer el diámetro y la sección de las hélices, estas dimensiones vienen descritas en la propia nomenclatura del propulsor. De forma tal, que cuando se nombran se acompañan de al menos dos números, el primero corresponde con el diámetro y el segundo con la sección de paso. En nuestro caso, se ha buscado que coincidan con los valores probados por el fabricante en el datasheet del motor. Por lo tanto, el tipo de hélices seleccionado es de 50x45. De manera que, tendrán un diámetro de 5.0 pulgadas y sección de paso de 4.5 pulgadas. Además, están fabricadas de policarbonato por lo que sus características mecánicas como ligereza y durabilidad son buenas para esta aplicación.

Figura 4: Hélices T5045C.

Nota. De «Catalogo de Amazon», Dalprop T5045C, (s.f), Amazon. CC-BY-NC.

5.2 Conjunto Electrónica

Respecto a la electrónica del dispositivo, está compuesta por cuatro componentes, los motores, los ESC, la batería, la placa principal y la placa de alimentación. A su vez, la placa principal y de alimentación son dos subconjuntos formados por varios elementos.

5.2.1 Motores DX2205

Los motores son los encargados de propulsar la aeronave y por lo tanto de hacerla volar, de forma que es necesario que puedan proporcionar un empuje superior al peso de la aeronave. Puesto que, el UAV ha de poder maniobrar y volar adecuadamente, los motores han de cumplir con el criterio de ser capaces de generar entre todos, una fuerza superior al doble del peso del dron. Por lo tanto, siendo T el empuje y W el peso el criterio se podría expresar de la siguiente forma:

$$T \geq 2\,W \tag{1}$$

Además, en los drones de tipo multirrotor se emplean los motores brushless debido a su eficiencia, duración y su capacidad de poder ser controlados mediante un ESC, entre otras ventajas.

Según estos dos criterios, se han seleccionado los motores DX2205 2300KV cuyas características se detallan a continuación:

Factor KV=2300 KV (RPM por voltio aplicado)

Dimensiones: 3.1 x 3.1 x 2.5 cm; 28 gramos

Corriente máxima=19.2 A

Potencia máxima= 213 W

Figura 5: Motor DX2205 2300KV.

Nota. De «Catalogo de Amazon», Dilwe DX2205, (s.f), Amazon. CC-BY-NC.

5.2.2 Batería LIPO

Las baterías son un elemento fundamental puesto que se encargan de suministrar energía eléctrica a todos los componentes electrónicos de la aeronave. Dentro de la selección de baterías hay que tener en cuenta diversos factores, entre ellos los más importantes son: la tensión, ha de ser suficiente para poder alimentar los componentes, pero sin exceder la máxima que aceptan; la capacidad, que determina la cantidad de corriente por unidad de tiempo que puede proporcionar; y la velocidad de descarga, corresponde con el tiempo que tarda en descargarse. Además, otro factor importante es el tipo de batería, las que más ventajas presentan son las de Ion-Litio y las Li-Po, entre ellas no ser tan contaminantes como el resto, ligeras con relación a su capacidad y no presentan efecto memoria.

Por lo que respecta a la batería seleccionada, se ha optado por emplear una batería LiPo de 3 celdas, cuya tensión es de 11.1 V para coincidir con los parámetros de testeo de los motores. En cuanto a su capacidad y velocidad de descarga, la primera es de 2200 mAh, mientras que la segunda es de 35C.

Figura 6: Batería LIPO 2200mAh 35C RoaringTop.

Nota. De «Catalogo de Amazon», Roaring Top, (s.f), Amazon. CC-BY-NC.

5.2.3 ESC

Los ESC son dispositivos capaces de controlar la velocidad de los motores eléctricos, de escobillas y brushless, variando la tensión que les proporcionan. Por esa razón, se

emplean en los UAV. En el caso del control de los brushless, como los que usan los cuadricópteros, los ESC cuentan con tres salidas una para cada fase. Mediante estas salidas es capaz de controlar el motor por secuencias de señales de corriente alterna trifásica. En lo relativo al control, estos dispositivos emplean señales de modulación de por ancho de pulsos (PWM) de 1 a 2 ms de duración. Según el tiempo del pulso que recibe el ESC, hará que el motor vaya a mayor o menor velocidad. Además, algunos cuentan con un circuito de eliminación de batería (BEC) que permite prescindir de una batería extra para alimentar con una tensión constante el resto de los componentes de la aeronave. Dentro de los BEC, podemos distinguir dos tipos los lineales y los conmutados. Respecto a los primeros son más baratos, pero son menos eficientes desde el punto de vista energético. En cuanto a los segundos, su principal ventaja es la eficiencia energética, pero son más caros (HobbyKing, 2021).

A la hora de seleccionar un ESC, se ha de tener en cuenta la corriente máxima que pueden aceptar. Puesto que, si la corriente demanda por los motores es superior a este valor podrían dañarse. En el caso de nuestra aeronave, con un ESC de 12 A bastará para controlarlos de forma segura.

Figura 7: ESC EMAX 12A.

Nota. De «Catalogo de Amazon», Dilwe EMAX 12A, (s.f), Amazon. CC-BY-NC.

5.2.4 Subconjunto Placa Principal

En este subconjunto, quedan incluidos tanto el microcontrolador, como los sensores y la propia placa.

5.2.4.1 Arduino Nano IOT33

El microcontrolador es el componente que actúa como el “cerebro” de la aeronave, se encarga de leer los sensores, calcular y enviar la acción de control a los ESC y procesar la información de los transmisores. Por lo tanto, es otro de los elementos fundamentales en un UAV.

En nuestro caso hemos optado por utilizar un Arduino Nano IoT33. Esto se debe a que ya se disponía de uno que se empleó en otra aplicación previamente. Además, al ser un Arduino el lenguaje de programación es sencillo y ya se conoce, también cuenta con mucha documentación. Asimismo, cuenta con doce pines capaces de enviar señales PWM de los que necesitaremos cuatro para controlar ESC y ocho más para leer los

sensores. Encima, el IoT33 viene con un módulo Wi-Fi con el que se podrían recibir las consignas de vuelo, dentro de sus limitaciones. Por último, como el control que ha de realizar el microcontrolador es de un proceso dinámicamente lento respecto a la velocidad con la que es capaz de trabajar es adecuado para controlar la aeronave.

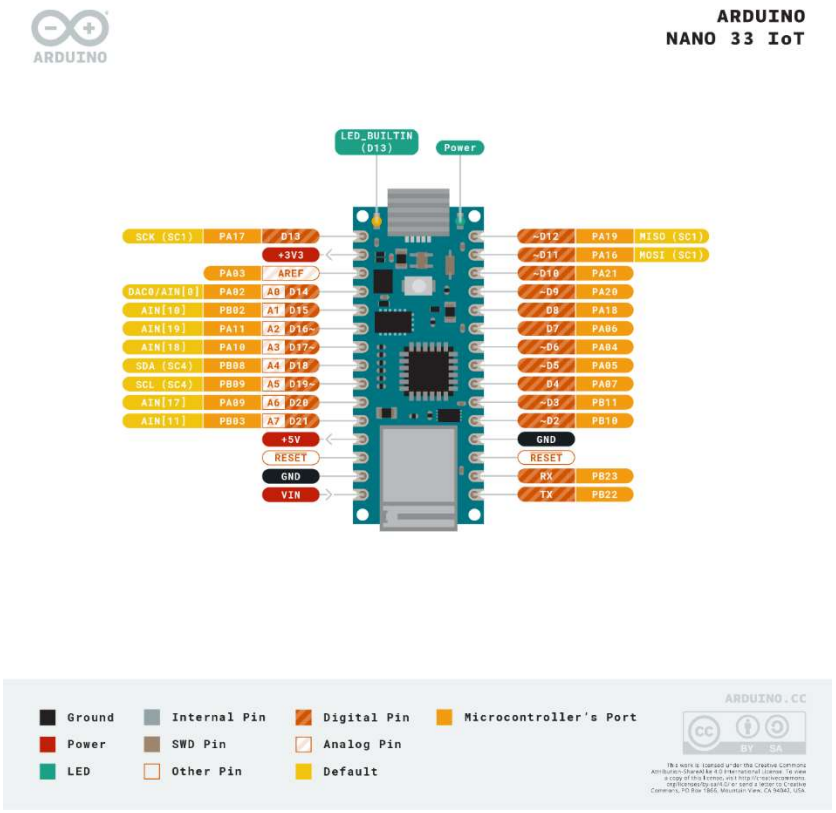


Figura 8: Arduino Nano IoT33 pinout.

Nota. De «Nano 33 IoT| Arduino Documentation», Arduino, (s.f), Arduino.cc. CC-BY-NC.

5.2.4.2 Ultrasonidos HC-SR04

Los sensores de ultrasonidos permiten realizar una medida de la distancia a un cierto objeto mediante la emisión/recepción de una onda ultrasónica que rebota en el objeto a detectar. Dicha onda, es emitida por el sensor mediante la excitación de una lámina magnetostrictiva que produce el ultrasonido, posteriormente esta onda rebota en el objeto y es captada por el sensor. Conociendo la velocidad de transmisión del sonido en el medio y el tiempo de retorno de la onda se puede estimar la distancia al objeto. Siendo “v” la velocidad de la onda, “tr” el tiempo de retorno y “d” la distancia, la manera de determinar la posición es la siguiente:

$$d = \frac{v \cdot tr}{2} \quad (2)$$

Debido a esta capacidad de medir la distancia, si se orienta de manera que apunte al suelo se puede emplear como una herramienta para medir la altura y contrastar la información obtenida por el altímetro. La principal problemática de estos sensores es su distancia de detección que por lo general ronda los dos metros. Sin embargo, siempre que no se exceda esta distancia pueden resultar útiles y más económicos que otros sensores como los LIDAR.

Ahora bien, el sensor utilizado es el HC-SR04 debido a que ya se disponía de uno y es compatible con Arduino. Emplear este sensor, nos limita a poder trabajar en alturas menores a los dos metros, pero para el diseño de un control de altura es suficiente. En el caso de que se quisiera conseguir mayor altura sería necesario recurrir a otros sensores con más distancia de detección.

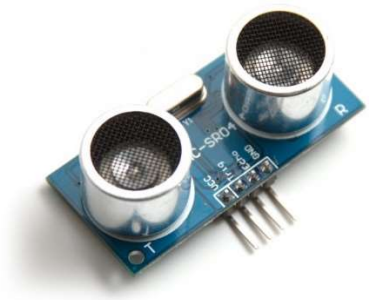


Figura 9: Ultrasonidos HC-SR04.

Nota. De «Catalogo de Amazon»,AZDelivery, (s.f), Amazon. CC-BY-NC.

5.2.4.3 IMU BNO055

La IMU es el sensor más importante en un UAV, puesto que aporta información sobre la orientación y desplazamiento de la aeronave. Pese a su utilidad, presentan la desventaja de que son muy sensible al ruido producido por las vibraciones, para solucionar esta problemática algunos fabricantes recomiendan envolver la IMU en un material espumoso capaz de absorber dichas vibraciones.

En este caso, esta IMU es capaz de medir posición angular y filtrar ruido debido a que realiza un procesado a bajo nivel de los valores del acelerómetro, giróscopo y magnetómetro que incluye. De esta forma, no se requerirá un procesado por nuestra parte lo que facilitará la implementación de los algoritmos de control y reducirá el coste computacional al no tener que procesar los datos en crudo.

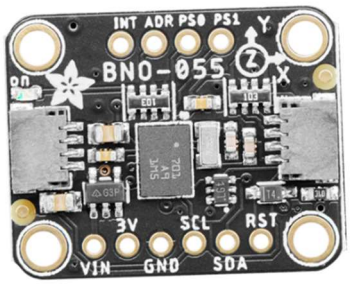


Figura 10: IMU BNO055.

Nota. De «Catalogo de Amazon», Adafruit, (s.f), Amazon. CC-BY-NC.

5.2.4.4 Giroscopio L3G4200D

El giroscopio es el sensor capaz de medir la velocidad angular de la aeronave, es decir, cambios en las derivadas de los ángulos Roll Pitch y Yaw. Mediante este dispositivo, se puede cerrar el bucle de control para la posición X e Y.

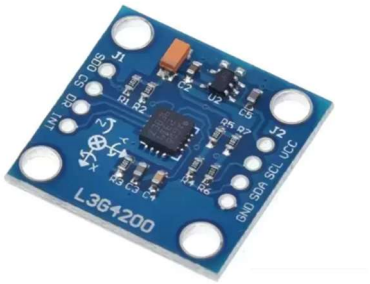


Figura 11: Giroscopio L3G4200D.

Nota. De «Catalogo de Amazon», ARCELI, (s.f), Amazon. CC-BY-NC.

5.2.5 Subconjunto Placa Alimentación

En este subconjunto, corresponde con la placa que permitirá alimentar al resto de elementos del UAV mediante la batería. En esta placa, destacan dos elementos, el conector XT60 macho y el interruptor de 12V.

5.2.5.1 Conector XT60

El XT60 es un conector formado por bornas de oro envuelto en una carcasa de nylon resistente a altas temperaturas. Estos terminales, permiten conexiones sólidas para amperajes de hasta 60 A constantes con tensiones de 12 a 24 VDC.

La batería seleccionada, emplea uno tipo hembra por lo que para alimentar la placa necesitaremos otro XT60, pero deberá ser macho.

Figura 12: Conector XT60 macho.

Nota. De «Catalogo de Amazon», Makerfire, (s.f), Amazon. CC-BY-NC.

5.2.5.2 Interruptor 12V

Para encender o apagar el UAV con seguridad será necesario emplear un interruptor capaz de soportar las tensiones y corrientes del dispositivo.

5.3 Proceso de fabricación y montaje

En este apartado se describen los pasos para la ejecución del montaje de la aeronave.

5.3.1 Implementación de la electrónica

Paso 1:

En primer lugar, se soldarán los headers a los pines de los sensores en los que fuese necesario como la IMU y el giroscopio. Este paso no hace falta en el caso de que todos los componentes electrónicos del subconjunto placa principal ya estén soldados previamente. Además, se deberán adherir con silicona la IMU y el giroscopio a sus respectivos encapsulados de espuma.

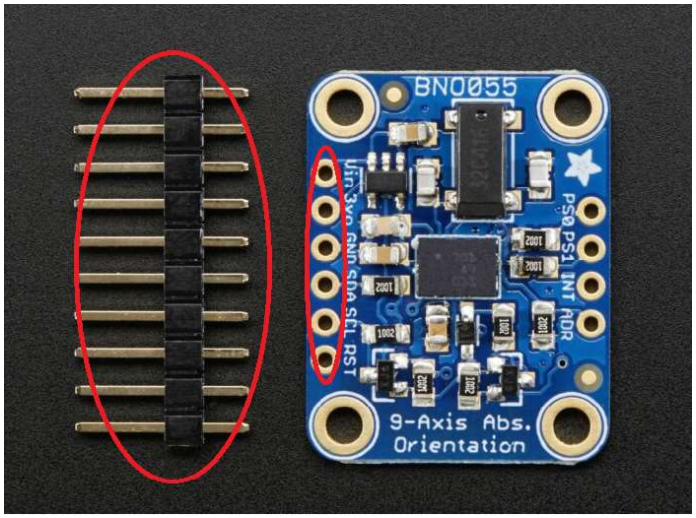


Figura 13: Paso 1, headers y donde soldarlos.

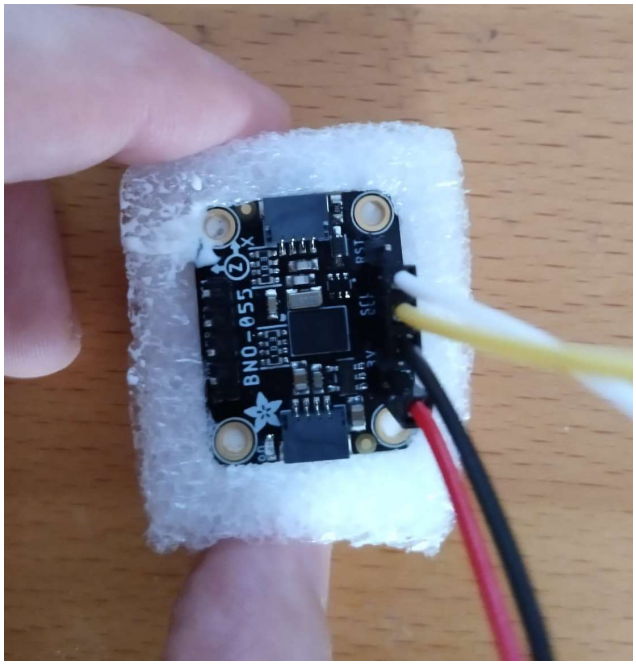


Figura 14: Paso 1, IMU BNO055 envuelta en espuma.

Paso 2:

En segundo lugar, se soldarán el interruptor micro, cablecillos junto con los headers macho y hembra a la placa principal siguiendo con el plano 08. También, se soldarán las conexiones de la placa de alimentación, será necesario un conector XT60 macho, cinco pares de cables de sección 1.5 mm y un interruptor, plano 03.

Paso 3:

En tercer lugar, se realizará el crimpado de los terminales tipo bala de las entradas y salidas tanto de los motores, y ESCs como el de los cables de la placa de alimentación. Siendo los terminales de los motores tipo hembra, las salidas y entrada del ESC tipo macho y los de los cables de la placa de alimentación tipo hembra.

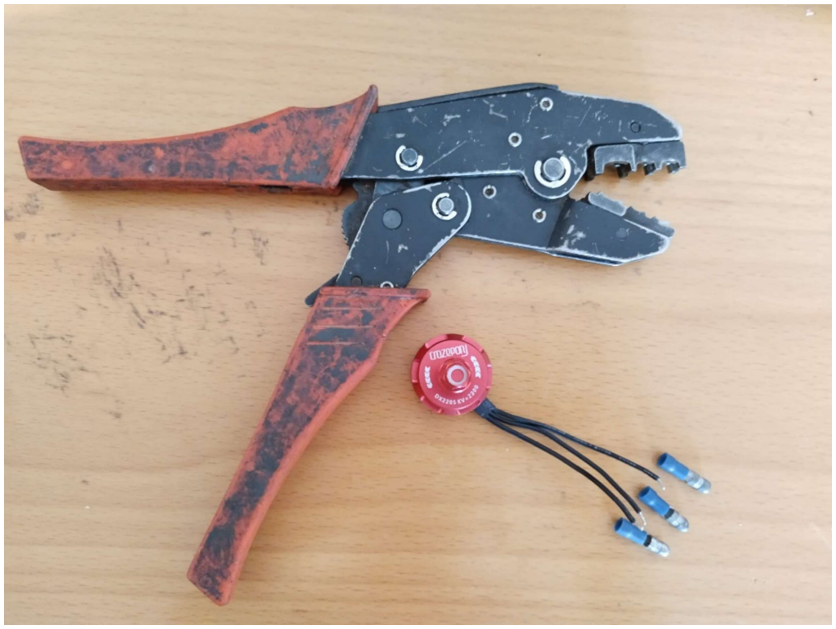


Figura 15: Paso 3, motor previo al crimpado.

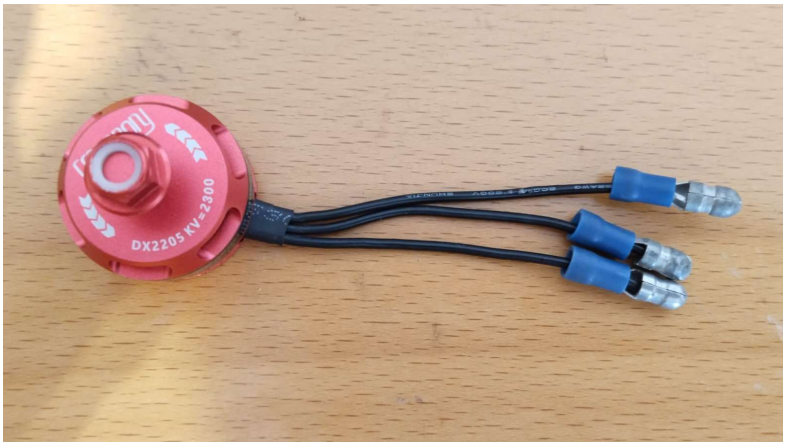


Figura 16: Paso 3, motor resultante.

5.3.2 Montaje de la base del fuselaje

Por lo que respecta el fuselaje el propio fabricante aporta un manual de instrucciones para llevar a cabo este procedimiento. Aun así, explicaremos los pasos seguidos.

Paso 4:

El fuselaje consta de los siguientes elementos principales, la base inferior, los brazos, la base intermedia que es exactamente igual a la inferior, la base superior, el tren de aterrizaje, varillas de aluminio y tornillería. En este paso, vamos a atornillar las varillas de aluminio a la base intermedia usando tornillos de métrica 3 de longitud 6 mm, el resultado debería ser el siguiente:

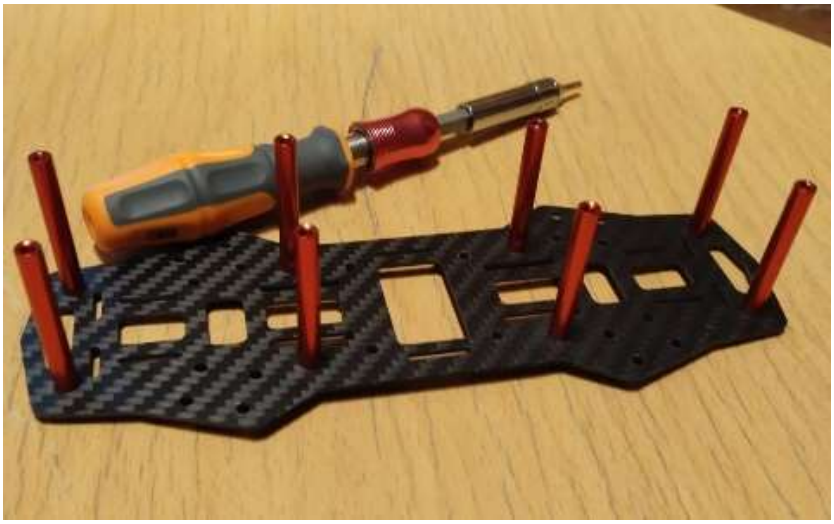


Figura 17: Paso 4, varillas y base intermedia.

Paso 5:

A continuación, se colocarán los cuatro brazos entre la base inferior y la base intermedia, colocados como queda reflejado en el manual. Posteriormente, atornillaremos los brazos a ambas bases con tornillos de métrica 3 longitud 10 mm de manera que el resultado sea como el mostrado a continuación.

Figura 18: Paso 5, base inferior, brazos y base intermedia.

5.3.3 Inclusión de la electrónica a la aeronave y montaje de la base superior

Paso 6:

En este paso, se deberán adherirá mediante silicona la IMU y el giroscopio envuelto en espuma en los extremos de la base intermedia a la altura de los brazos. Cabe destacar, que ambos sensores llevan impresos un símbolo con la orientación de los ejes X,Y,Z que medirán, por lo que ambos sensores se deberán orientar igual, con el eje Y en dirección paralela a la línea que forma las dos hileras de varillas y en sentido que el eje Y del giroscopio vaya hacia el centro de la aeronave. Es decir, se deberán colocar conforme al plano 06. De forma que quedaría como muestra la siguiente figura:

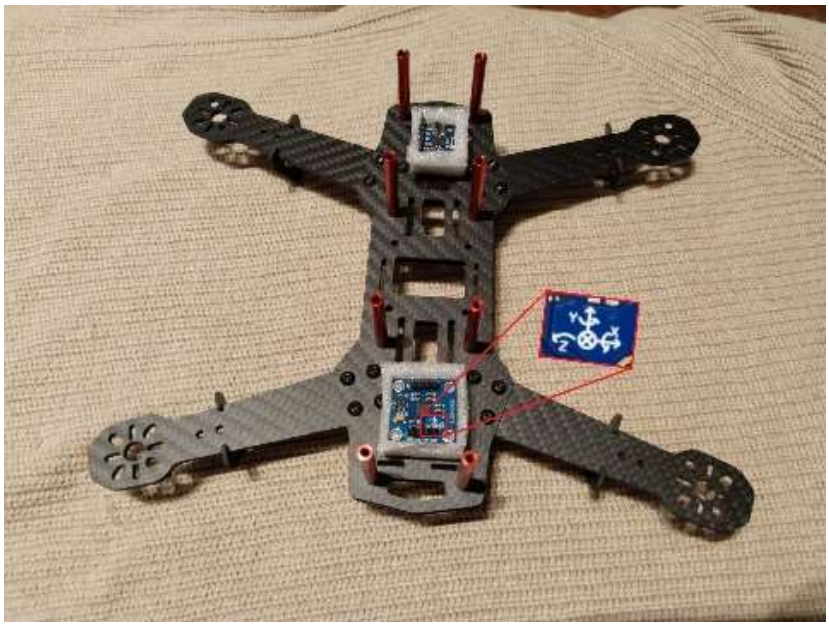


Figura 19: Paso 6, inclusión de IMU y giroscopio.

Paso 7:

A continuación, se deberán adherir tanto la placa principal que quedará ubicada en el centro simétrico del cuerpo del dron, de forma que el largo de la placa quede perpendicular al eje Y anterior. De manera similar, se colocará el ultrasonido en el centro simétrico solo que este deberá estar adherido al base inferior encarado hacia debajo de forma que el emisor y receptor del sensor queden mirando hacia en suelo. Es decir, se deberán colocar conforme al plano 06. Además, el largo del sensor deberá quedar perpendicular al eje Y como en la placa principal.

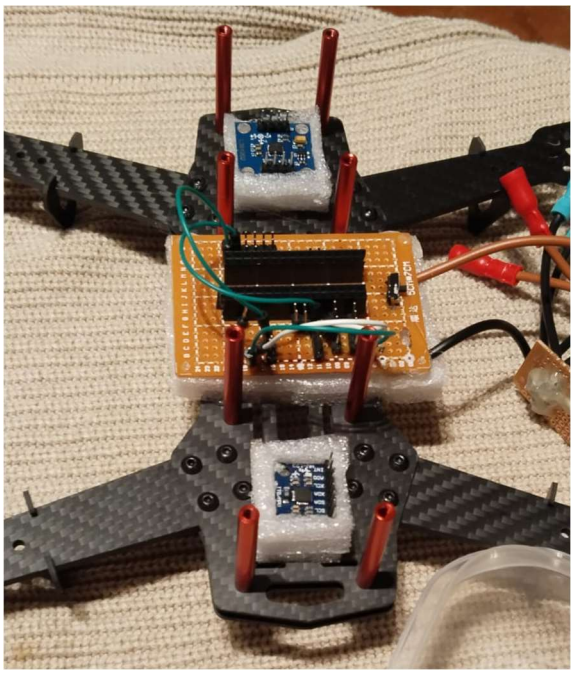


Figura 20: Paso 7, ultrasonidos y placa principal.

Paso 8:

Se colocarán los motores en los extremos de cada brazo y posteriormente se atornillarán al fuselaje mediante los tornillos proporcionados por el fabricante del motor. Además, se deberán ubicar de forma que los motores del mismo tipo CW o CCW queden en el vértice opuesto. La configuración de los motores deberá seguir el siguiente esquema:

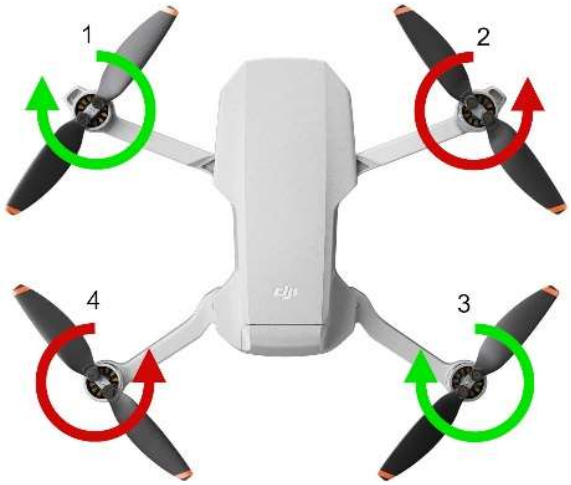


Figura 21: Paso 8, sentido de giro de los motores.

Nota. Adaptado de «Catalogo de Amazon», DJI, (s.f), Amazon. CC-BY-NC.

Paso 9:

En el paso 9, colocaremos los ESCs en cada uno de los brazos entre el motor y la base. Posteriormente, los uniremos a los brazos mediante bridas de manera que queden bien sujetos. Por último, uniremos los terminales bala a los del motor de forma que los permita girar conforme a la configuración anterior.

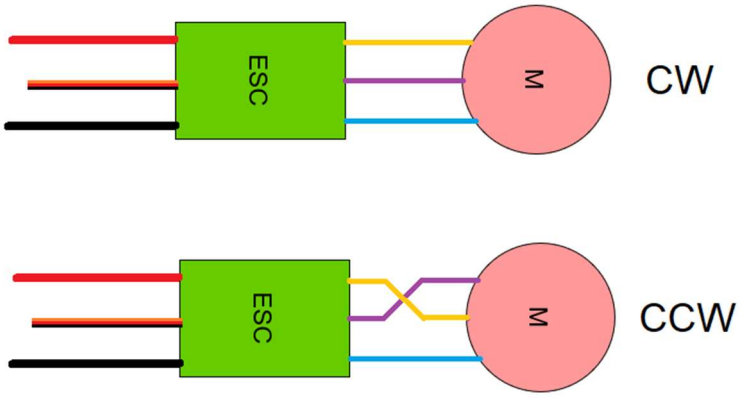


Figura 22: Paso 9, conexión de ESC para configuración de giro CW y CCW.

Figura 23: Paso 9, ESCs en la aeronave.

Paso 10:

En este paso, se realiza el cableado de los sensores y actuadores a la placa perforado mediante cablecillos, estás conexiones se realizan conforme al plano 07.

Paso 11:

A continuación, se atornillará la base superior a la base intermedia mediante tornillos de métrica 3 que se deberán pasar por las varillas que sobresalen de la base intermedia.

Paso 12:

Se colocarán las hélices en los motores y se unirán a estos mediante una tuerca proporcionada por el fabricante del motor. Será necesario que las hélices CW vayan con los motores CW y lo mismo con los CCW.

Figura 24: Paso 12, inclusión de las hélices.

Paso 13:

Se colocará sobre la base superior la batería y la placa de alimentación sobre la batería de forma que queden ubicadas como en el plano 04. Posteriormente, se asegurará la correcta sujeción de ambas atándolas mediante correas a la base superior.

Paso 14:

Finalmente, conectamos los terminales de alimentación de la placa principal y los ESCs a la placa de alimentación y se asegurará que ningún cable pueda quedar expuesto a entrar en contacto con las hélices.

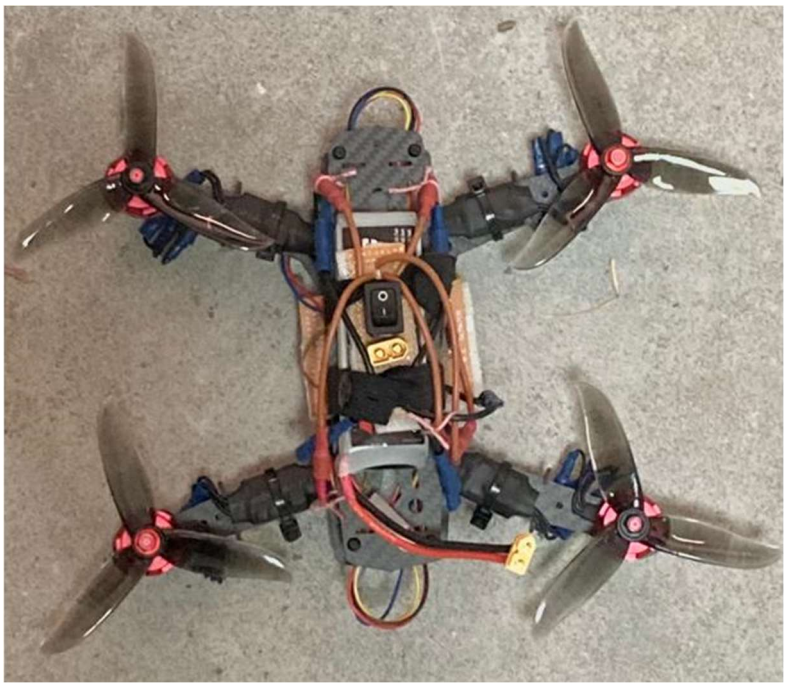


Figura 25: Paso 14, resultado final del montaje.

Paso 15:

Como último paso del montaje, quedará calibrar el centro de gravedad de la aeronave. Para ello, se accionarán todos los motores a una velocidad suficiente como para elevar el dispositivo. S en el despegue presenta derivas hacia un lado, se deberá añadir algo de peso en el lado opuesto ya que estas derivas son causa principalmente de un centro de gravedad alejado del centro simétrico de la aeronave.

6. Justificación detallada de la solución

En este apartado, se desarrollan los cálculos relacionados con los componentes que lo requerían, así como el diseño de los controladores requeridos en la aplicación.

6.1 Selección de componentes

En este aparatado se realizarán los cálculos de los componentes que se han seleccionado.

Los datos empleados en la realización de los cálculos se han extraído de las hojas de características de los componentes. Para el diseño, se ha tenido en cuenta primero una estimación del peso de la aeronave, posteriormente se han seleccionado los motores capaces de lograr un empuje que cumpla el criterio de levantar dos veces el peso del

dispositivo. Por último, se han seleccionado tanto la batería y el ESC en función de las necesidades de los motores.

6.1.1 Motores

Como se ha explicado anteriormente, se tiene que cumplir el criterio de que el motor puede ejercer un empuje igual o mayor al peso de la aeronave.

Para el cálculo del empuje, se ha realizado siendo N el número de motores, Wmax la potencia máxima y ε la eficiencia, teniendo en cuenta que la masa estimada será de unos 600 g, el valor que tendrán que ejercer será más de 1200 g doble de la masa de la aeronave.

$$Masa\ UAV \approx 600\ g \tag{3}$$

$$Tmax = N \cdot Wmax \cdot \varepsilon = 4 \cdot 213 \cdot 3.1 = 2640\ g \tag{4}$$

$$2640\ g > 1200\ g \tag{5}$$

Por lo que podemos concluir, que el empuje máximo que podrán generar los motores será superior al doble al peso de la aeronave, De forma que se cumple el criterio anteriormente establecido.

MOTOR PERFORMANCE DATA :

MODEL	KV (rpm/V)	Voltage (V)	Prop	Load Current (A)	Pull (g)	Power (W)	Efficiency (g/W)	Lipo Cell	Weight (g) Approx
DX2205	2300	11.1	5045	19.2	660	213	3.1	2-4S	28
		14.8		27.6	950	408	2.3		
	2600	11.1	4045	18.5	530	205	2.6		
		14.8		23.2	710	343	2.1		

Figura 26: Datasheet motor DX2205 2300KV.

Nota. Adaptado de «Datasheet Motor DX2205», CrazyPonny, (s.f), CrazyPonny. CC-BY-NC

6.1.2 Batería

Respecto a la batería, será necesario conocer el consumo del UAV y su autonomía, es decir, el tiempo aproximado que podrá estar en funcionamiento antes de que esta se descargue. Para ello, se ha realizado una serie de cálculos que se muestran a continuación. Además, cabe destacar que se ha partido del supuesto de que la aeronave únicamente está ejerciendo la fuerza suficiente como para vencer su propio peso.

En primer lugar, se ha calculado el consumo de corriente necesario para los propulsores. Siendo “m” la masa de la aeronave, ε la eficiencia de los motores, V la tensión de la batería e I_m la corriente total demandada.

$$W = \frac{m}{\varepsilon} = \frac{600\ g}{3.1\ g/W} = 193.54\ W \tag{6}$$

$$W = I_m * V \rightarrow I_m = \frac{W}{V} = \frac{193.548\ W}{11.1\ V} = 17.43\ A \tag{7}$$

Esta será la corriente total que demanden entre los cuatro motores para levantar el peso de la aeronave.

A continuación, se analizará el tiempo que tardará en descargarse la batería suministrando la corriente anterior de manera constante. Siendo, C la capacidad, I_m la corriente total demanda y T_a el tiempo de autonomía.

$$T_a = \frac{C \cdot \frac{60\ min}{1\ h}}{I_m} = \frac{2.2\ A \cdot \frac{60\ min}{1\ h}}{17.43\ A} = 7.57\ min \tag{8}$$

Por lo tanto, podemos concluir que la autonomía estimada del dispositivo en condiciones de vuelo donde únicamente ejerza la fuerza suficiente para mantenerse en el aire será de unos 7 minutos y medio.

También se valoró emplear dos baterías conectadas en paralelo en lugar de una única batería. De ser así, el peso del UAV aumentaría 180 g así como su capacidad otros 2200mAh por lo que la autonomía final sería mayor. Sin embargo, se optó por emplear una única batería por tener un dispositivo más ligero y fácil de controlar.

6.1.3 ESC

En cuanto a los ESC, necesitaremos conocer el consumo de corriente que requerirán los motores de forma que el controlador pueda soportar el amperaje. Para ello calculamos el consumo de corriente en condiciones donde la aeronave ejerza la fuerza suficiente para mantenerse en el aire. El cálculo es el siguiente donde I_m es la corriente que se suministra, para conseguir que todos los motores generen una fuerza suficiente para levantar la masa de la aeronave y N es el número de motores que tiene el UAV.

$$I = \frac{I_m}{N} = \frac{17.43\ A}{4} = 4.35\ A \tag{9}$$

De forma que el amperaje del ESC deberá ser superior a 4.35A, por esa razón con un ESC de 12 A será suficiente para controlar los motores.

B.Product specification

Item	Continuous Current	Burst current (10S)	Li-xx Battery (cell)	Dimension L*W*H(mm)	Weight (g) wires Included	BEC Mode	BEC Output	Programmable
EMAX BLHeli-6A	6A	8A	1-2	22×13×5.5	6	Linear	0.8A/5V	YES
EMAX BLHeli -12A	12A	15A	2-4	42×20×8	11	Linear	1A/5V	YES
EMAX BLHeli -20A	20A	25A	2-4	52×26×7	28	Linear	2A/5V	YES
EMAX BLHeli -25A	25A	30A	2-4	52×26×7	28	Linear	2A/5V	YES
EMAX BLHeli -30A	30A	40A	2-4	52×26×7	28	Linear	2A/5V	YES
EMAX BLHeli -30A-OPTO	30A	40A	2-6	67×26×10	25	----	----	YES
EMAX BLHeli -40A-UBEC	40A	50A	2-6	73×28×12	41	Switch	3A/5V	YES
EMAX BLHeli -50A-UBEC	50A	60A	2-6	73×28×12	41	Switch	3A/5V	YES
EMAX BLHeli -60A-UBEC	60A	80A	2-6	73×36×12	63	Switch	5A/5V	YES
EMAX BLHeli -80A-UBEC	80A	100A	2-6	86×38×12	81	Switch	5A/5V	YES

Figura 27: Datasheet ESC EMAX BLHELI 12A.

Nota. Adaptado de «Datasheet ESC EMAX BLHELI 12A», EMAX, (s.f), EMAX. CC-BY-NC.

6.2 Peso

Antes de la construcción del dispositivo se elaboró la siguiente estimación del peso que tendría la aeronave basada en el de los componentes. El resultado fue de un peso estimado de 593 gramos mientras que el real ronda los 605 gramos. Cabe destacar, que los elementos que más influencian en el peso son la batería y el fuselaje que en conjunto superan la mitad de la masa total. Por otro lado, los motores y los ESCs contribuyen con un 32% del peso total. Por lo que respecta al resto de componentes no suponen una diferencia superior al 5% de manera individual.

Tabla 7: Computo del peso en la aeronave.

Componentes	Peso Unidad (g)	Cantidad	Peso Total (g)	Peso Relativo(%)
Arduino	9	1	9	2%
IMU	6	1	6	1%
Giroscopio	6	1	6	1%
Ultrasonidos	10	1	10	2%
Fuselaje	140	1	140	24%
Motor	28,4	4	113,6	19%
ESC	20	4	80	13%
Batería	185	1	185	31%
Perfboard	10	2	20	3%
Interruptor 12V	15	1	15	3%
Interruptor Micro	5	1	5	1%
XT60 hembra	4	1	4	1%
		Total	593,6	100%

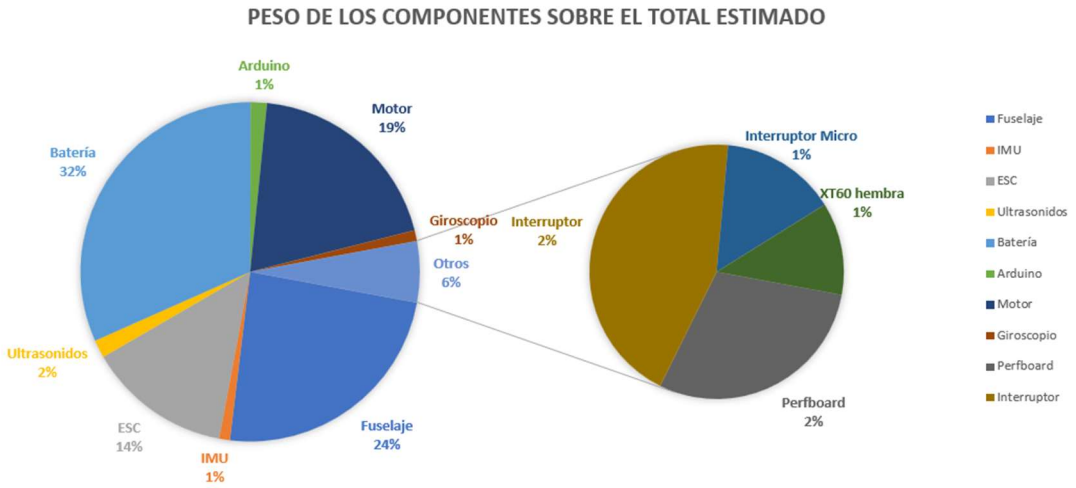


Figura 28: Reparto del peso en la aeronave.

6.3 Diseño del control

En este apartado, desarrollaremos los cálculos y procedimiento del diseño de los reguladores para el control de altura y estabilidad.

6.3.1 Modelado

El modelado del cuadricóptero se puede obtener mediante un estudio de la cinemática y dinámica del UAV. El análisis del modelo dinámico parte de las ecuaciones de Newton-Euler empleando un balance de fuerzas y pares en referencia al cuerpo de la aeronave. Por otro lado, el análisis cinemático parte del estudio de las velocidades lineales y angulares en referencia fija frente a las velocidades en el cuerpo del UAV (Miranda Colorado, 2020).

Para el análisis, partiremos de los siguientes supuestos, en primer lugar, que el UAV se puede entender de forma ideal como un cuerpo rígido con cuatro motores iguales ubicados al final de los brazos del vehículo. Por otra parte, el modelado se realiza estudiando el movimiento de traslación y rotación del cuadricóptero.

6.3.1.1 Análisis cinemático

El análisis cinemático se lleva a cabo estudiando la velocidad de traslación y rotación referenciadas a un sistema de referencia fijo frente a las del cuerpo de la aeronave, estas se expresan empleando la siguiente simbología:

Tabla 8: Simbología del análisis cinemático.

$\dot{x}$	Velocidad de traslación en X en referencia al sistema de referencia fijo
$\dot{y}$	Velocidad de traslación en Y en referencia al sistema de referencia fijo
$\dot{z}$	Velocidad de traslación en Z en referencia al sistema de referencia fijo
U	Velocidad de traslación en X en referencia al cuerpo del UAV
V	Velocidad de traslación en Y en referencia al cuerpo del UAV
W	Velocidad de traslación en Z en referencia al cuerpo del UAV
$\dot{\varphi}$	Velocidad de rotación en Roll en referencia al sistema de referencia fijo
$\dot{\theta}$	Velocidad de rotación en Pitch en referencia al sistema de referencia fijo

Ψ	Velocidad de rotación en Yaw en referencia al sistema de referencia fijo
P	Velocidad de rotación en Roll en referencia al cuerpo del UAV
Q	Velocidad de rotación en Pitch en referencia al cuerpo del UAV
R	Velocidad de rotación en Yaw en referencia al cuerpo del UAV
I_v	Vector de velocidad lineal del sistema de referencia fijo (I)
B_v	Vector de velocidad lineal local al cuerpo del UAV (B)
W	Vector de velocidades angulares del sistema de referencia fijo (I)
W_B	Vector de velocidades angulares al cuerpo del UAV (B)

Las ecuaciones que definen las velocidades lineales del cuadricóptero vienen de la relacionar el sistema de referencia (I) con el cuerpo de la aeronave (B), donde los vectores I_v y B_v son:

$$I_v = (\dot{x}, \dot{y}, \dot{z})' \quad (10)$$

$$B_v = (u, v, w)' \quad (11)$$

Por lo tanto, la relación de velocidades lineales viene determinada por la rotación en Roll,Pitch,Yaw para alinear los ejes y los vectores de velocidades en ambos sistemas:

$$I_v = {}^{I}R_B \cdot {}^{B}v \quad (12)$$

Si realizamos el cálculo anterior y descomponemos el vector I_v en cada uno de sus componentes obtenemos la expresión de las velocidades en x, y, z:

$$\dot{x} = u[\cos\theta \cos\psi] + v[\sin\varphi \sin\theta \cos\psi - \cos\varphi \sin\psi] + w[\cos\varphi \sin\theta \cos\psi + \sin\varphi \sin\psi] \quad (13)$$

$$\dot{y} = u[\cos\theta \sin\psi] + v[\sin\varphi \sin\theta \sin\psi - \cos\varphi \cos\psi] + w[\cos\varphi \sin\theta \sin\psi - \sin\varphi \cos\psi] \quad (14)$$

$$\dot{z} = u[-\sin\theta] + v[\sin\varphi\cos\theta] + w[\cos\varphi\cos\theta] \quad (15)$$

Estas expresiones, se pueden simplificar cuando la aeronave está en vuelo estacionario o en vuelo hovering donde los ángulos de Roll y Pitch son aproximadamente 0. Finalmente, las ecuaciones quedarían de la siguiente forma:

$$Si\ \varphi \approx 0\ \ y\ \ \theta \approx 0 \quad (16)$$

$$\dot{x} = u\cos\psi - v\sin\psi \quad (17)$$

$$\dot{y} = u\sin\psi + v\cos\psi \quad (18)$$

$$\dot{z} = w \quad (19)$$

Por otro lado, necesitaremos otras tres ecuaciones correspondientes a las velocidades angulares, estas se consiguen relacionando las velocidades rotacionales del cuerpo y el sistema inercial. De forma que, sus vectores de velocidades son los siguientes:

$$W = (\dot{\varphi}, \dot{\theta}, \dot{\Psi})' \quad (20)$$

$$W_B = (p, q, r)' \quad (21)$$

Por lo tanto, se establece la relación de velocidades angulares como:

$$W = J^{-1}W_B \quad (22)$$

Finalmente, si descomponemos el vector de velocidades angulares del sistema de referencia fijo en sus componentes Roll, Pitch, Yaw se quedan las siguientes expresiones:

$$\dot{\varphi} = p + q\ [\sin\varphi\ \tan\theta] + r[\cos\varphi\tan\theta] \quad (23)$$

$$\dot{\theta} = q\ [\cos\varphi] + r[-\sin\varphi] \quad (24)$$

$$\dot{\psi} = q\left[\frac{\sin\varphi}{\cos\theta}\right] + r\left[\frac{\cos\varphi}{\cos\theta}\right] \quad (25)$$

Al igual que en el caso de las velocidades traslacionales, en aplicaciones donde el vuelo se realiza de forma que los ángulos de Roll y Pitch sean cercanos a 0, las expresiones se simplifican coincidiendo las velocidades angulares de ambos sistemas:

$$Si\ \varphi \approx 0\ \ y\ \ \theta \approx 0 \tag{26}$$

$$\dot{\varphi} = p \tag{27}$$

$$\dot{\theta} = q \tag{28}$$

$$\dot{\psi} = r \tag{29}$$

6.3.1.2 Análisis dinámico

El análisis dinámico se realiza estudiando las aceleraciones lineales y rotacionales de la aeronave, relacionando la fuerzas y momentos de los propulsores junto con las fuerzas y momentos externos, todo referenciado al cuerpo del UAV.

Tabla 9: Simbología del análisis dinámico.

$\dot{u}$	Aceleración lineal en el eje X en referencia al centro del UAV
$\dot{v}$	Aceleración lineal en el eje Y en referencia al centro del UAV
$\dot{w}$	Aceleración lineal en el eje Z en referencia al centro del UAV
$\dot{p}$	Aceleración angular en Roll
$\dot{q}$	Aceleración angular en Pitch
$\dot{r}$	Aceleración angular en Yaw
M	Masa de la aeronave
$\vec{F}^B$	Total de las fuerzas externas en el cuerpo de la aeronave
$\vec{F}_x^B$	Fuerza externa en X en el cuerpo de la aeronave
$\vec{F}_y^B$	Fuerza externa en Y en el cuerpo de la aeronave

$\vec{F}_z^B$	Fuerza externa en Z en el cuerpo de la aeronave
I_{xx}	Inercia en el eje X sobre el centro de la aeronave
I_{yy}	Inercia en el eje Y sobre el centro de la aeronave
I_{zz}	Inercia en el eje Z sobre el centro de la aeronave
$\vec{\Gamma}^B$	Componente de fuerza giroscópica
$\vec{F}_G^B$	Fuerzas gravitatorias en el centro del UAV
$\vec{F}_T^B$	Fuerzas de empuje de los actuadores
$\vec{F}_A^B$	Fuerzas aerodinámicas
$\vec{F}_{GR}^B$	Fuerzas de tracción del suelo
$\vec{M}^B$	Total de pares
$\vec{M}_T^B$	Pares generados por los actuadores
$\vec{M}_A^B$	Pares generados por fuerzas aerodinámicas
$\vec{M}_{GR}^B$	Pares generados por la tracción del suelo

La relación de las fuerzas externas con las aceleraciones lineales del cuerpo de la aeronave viene determinada por la siguiente ecuación:

$$\vec{F}^B = m\dot{\vec{v}}^B + \vec{w}^B \times (m\vec{v}^B) \tag{30}$$

Por lo que las aceleraciones lineales del cuerpo de la aeronave quedan definidas de la siguiente forma:

$$\dot{u} = rv - qw + \frac{\vec{F}_x^B}{m} \tag{31}$$

$$\dot{v} = pw - ru + \frac{\vec{F}_x^B}{m} \tag{32}$$

$$\dot{w} = qu - pv + \frac{\vec{F}_z^B}{m} \tag{33}$$

Respecto a las aceleraciones angulares provocado por pares externos vienen determinado por la siguiente ecuación:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{34}$$

$$\vec{M}^B = I\dot{\vec{w}}^B + \vec{w}^B \times (I\vec{w}^B) + \vec{\Gamma}^B \tag{35}$$

Finalmente, las aceleraciones rotacionales provocadas por momentos externos quedarían como se presentan a continuación:

$$\dot{p} = \frac{(I_{yy} - I_{zz})}{I_{xx}} qr + \frac{M_x^B - \Gamma_x^B}{I_{xx}} \tag{36}$$

$$\dot{q} = \frac{(I_{zz} - I_{xx})}{I_{yy}} pr + \frac{M_y^B - \Gamma_y^B}{I_{yy}} \tag{37}$$

$$\dot{r} = \frac{(I_{xx} - I_{yy})}{I_{zz}} pq + \frac{M_z^B - \Gamma_z^B}{I_{zz}} \tag{38}$$

Con lo visto hasta ahora, queda resulto todas las aceleraciones provocadas por factores externos al cuerpo de la aeronave. Únicamente resta analizar las aceleraciones resultantes de fuerzas y pares que actúan sobre el cuerpo del UAV.

Respecto a las fuerzas que actuarán en el UAV, estas serán, la fuerza gravitatoria, la provocada por la tracción de los motores, las fuerzas aerodinámicas y las fuerzas de tracción del suelo.

$$\vec{F}^B = \vec{F}_G^B + \vec{F}_T^B + \vec{F}_A^B + \vec{F}_{GR}^B \tag{39}$$

A su vez, los pares provendrán de las mismas fuentes que las fuerzas. Sin embargo, se considera que la fuerza de la gravedad no provocará par alguno ya que esta se aplica sobre el centro de gravedad del UAV que a su vez corresponderá con el centro simétrico.

$$\vec{M}^B = \vec{M}_T^B + \vec{M}_A^B + \vec{M}_{GR}^B \tag{40}$$

Las fuerzas gravitatorias de la aeronave vienen determinadas de la siguiente forma:

$$^BR_E = RPY(\varphi, \theta, \psi) \tag{41}$$

$$\vec{F}_G^B = \begin{pmatrix} \vec{F}_{Gx}^B \\ \vec{F}_{Gy}^B \\ \vec{F}_{Gz}^B \end{pmatrix} = {}^BR_E\vec{F}_G^G = {}^BR_E\begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} = \begin{pmatrix} mg\sin\theta \\ -mg\cos\theta\sin\varphi \\ -mg\cos\theta\cos\varphi \end{pmatrix} \tag{42}$$

Cuando Roll y Pitch son aproximadamente 0, la expresión anterior se simplifica:

$$\vec{F}_G^B = \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} \tag{43}$$

La fuerza de tracción proveniente de los motores se puede entender como un empuje total en el eje Z, dicho empuje no es más que la suma de la tracción ejercida por cada actuador:

$$\vec{F}_T^B = \vec{T}^B = \begin{pmatrix} 0 \\ 0 \\ T \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \sum_1^n T_i \end{pmatrix} \tag{44}$$

Las fuerzas aerodinámicas provienen de la rotación del motor y el giro de las hélices y se definen de la siguiente forma:

$$\vec{F}_A^B = \vec{D}^B = \begin{pmatrix} D_x^B \\ D_y^B \\ D_z^B \end{pmatrix} = -T\begin{bmatrix} \bar{c} & 0 & 0 \\ 0 & \bar{c} & 0 \\ 0 & 0 & 0 \end{bmatrix}\left(\vec{v}^B - \vec{W}^B\right) = -T\begin{pmatrix} \bar{c}(u-u_w) \\ \bar{c}(v-v_w) \\ 0 \end{pmatrix} \tag{45}$$

Las fuerzas aerodinámicas se anulan cuando la variación angular u y v se anulan respecto al sistema de referencia local de la aeronave. También ocurre cuando el viento anule la velocidad angular aparente en los ejes X e Y.

La fuerza de tracción del suelo, provienen del denominado efecto suelo que provoca que la sustentación aérea se incremente en vuelos cercanos al suelo ya que con su presencia las corrientes de aire debajo de la aeronave se modifican resultando en una mayor sustentación.

$$\vec{F}_{GR}^{G} = \begin{pmatrix} F_{GRx}^{G} \\ F_{GRy}^{G} \\ F_{GRz}^{G} \end{pmatrix} \quad (46)$$

6.3.1.3 Modelo lineal

Las ecuaciones obtenidas en los apartados anteriores cubren las doce variables de estado seis correspondientes a velocidades lineales y rotacionales. En cambio, las otras seis corresponden a las aceleraciones traslacionales y rotacionales del cuerpo del UAV. A continuación, se recogen las ecuaciones obtenidas:

$$\dot{x} = u[\cos\theta\cos\psi] + v[\sin\varphi\sin\theta\cos\psi - \cos\varphi\sin\psi] + w[\cos\varphi\sin\theta\cos\psi + \sin\varphi\sin\psi] \quad (47)$$

$$\dot{y} = u[\cos\theta\sin\psi] + v[\sin\varphi\sin\theta\sin\psi - \cos\varphi\cos\psi] + w[\cos\varphi\sin\theta\sin\psi - \sin\varphi\cos\psi] \quad (48)$$

$$\dot{z} = u[-\sin\theta] + v[\sin\varphi\cos\theta] + w[\cos\varphi\cos\theta] \quad (49)$$

$$\dot{\varphi} = p + q\,[\sin\varphi\ \tan\theta] + r[\cos\varphi\tan\theta] \quad (50)$$

$$\dot{\theta} = q\,[\cos\varphi] + r[-\sin\varphi] \quad (51)$$

$$\dot{\psi} = q\left[\frac{\sin\varphi}{\cos\theta}\right] + r\left[\frac{\cos\varphi}{\cos\theta}\right] \quad (52)$$

$$\dot{u} = rv - qw + g\sin\theta + \frac{A_x^B + D_x^B + F_{GRx}^B}{m} \quad (53)$$

$$\dot{v} = pw - ru - g\cos\theta\sin\varphi + \frac{A_y^B + D_y^B + F_{GRy}^B}{m} \quad (54)$$

$$\dot{w} = qu - pv - g\cos\theta\cos\varphi + \frac{T + A_z^B + D_z^B + F_{GRz}^B}{m} \quad (55)$$

$$\dot{p} = \frac{(I_{yy} - I_{zz})}{I_{xx}} qr + \frac{M_{Tx}^{B} + M_{wx}^{B} + M_{GRx}^{B} - \Gamma_{x}^{B}}{I_{xx}} \quad (56)$$

$$\dot{q} = \frac{(I_{zz} - I_{xx})}{I_{yy}} pr + \frac{M_{Ty}^{B} + M_{wy}^{B} + M_{GRy}^{B} - \Gamma_{y}^{B}}{I_{yy}} \quad (57)$$

$$\dot{r} = \frac{(I_{xx} - I_{yy})}{I_{zz}} pq + \frac{M_{Tz}^{B} + M_{wz}^{B} + M_{GRz}^{B} - \Gamma_{z}^{B}}{I_{zz}} \quad (58)$$

Por lo tanto, el sistema a controlar corresponde con uno no lineal, altamente acoplado y subactuado, ya que únicamente tenemos cuatro actuadores y seis grados de libertad. Sin embargo, en condiciones donde el Roll, Pitch, Yaw cercanos a 0 se puede simplificar el sistema de forma que se podrá linealizar y desacoplar. En ese caso, las ecuaciones pasan a ser las siguientes:

$$\dot{x}(t) = u(t) \quad (59)$$

$$\dot{y} = v(t) \quad (60)$$

$$\dot{z} = w(t) \quad (61)$$

$$\dot{\varphi} = p(t) \quad (62)$$

$$\dot{\theta} = q(t) \quad (63)$$

$$\dot{\psi} = r(t) \quad (64)$$

$$\dot{u} = g * \theta(t) \quad (65)$$

$$\dot{v} = -g * \theta(t) \quad (66)$$

$$\dot{w} = \frac{1}{m} T(t) \quad (67)$$

$$\dot{p} = \frac{1}{I_{xx}} M_{Tx}(t) \quad (68)$$

$$\dot{q} = \frac{1}{I_{yy}} M_{Ty}(t) \quad (69)$$

$$\dot{r} = \frac{1}{I_{zz}} M_{Tz}(t) \quad (70)$$

Ahora que ya tenemos el proceso linealizado y desacoplado, podemos deducir la función de transferencia del proceso entre la relación entre altura y torque corresponde con un doble integrador dependiente de la masa del UAV.

$$\dot{z}(t) = \frac{1}{m}T(t) \rightarrow G(s) = \frac{1}{ms^2} \tag{71}$$

6.3.2 Control de altura

El objetivo de este control, es el de asegurar la sustentación de la aeronave respecto a una consigna de altura. Para ello el cuadricóptero, ha de corregir las desviaciones en la variable a controlar o, en otras palabras, el error respecto a la referencia. La manera de actuar sobre la variable de la altura, es generando un mayor o menor empuje en el eje Z de la aeronave, aumentado o reduciendo la velocidad de rotación de los motores. Sin embargo, para lograr que el proceso siga la referencia es necesario el diseño de un regulador que en el caso del proyecto se optó por un regulador PD.

Para el diseño del PD se partió del modelo desacoplado y linealizado del proceso, que coincide con un doble integrador.

$$G(s) = \frac{1}{ms^2} \tag{72}$$

De forma que el proceso controlado con el regulador quedaría como el siguiente diagrama de bloques:

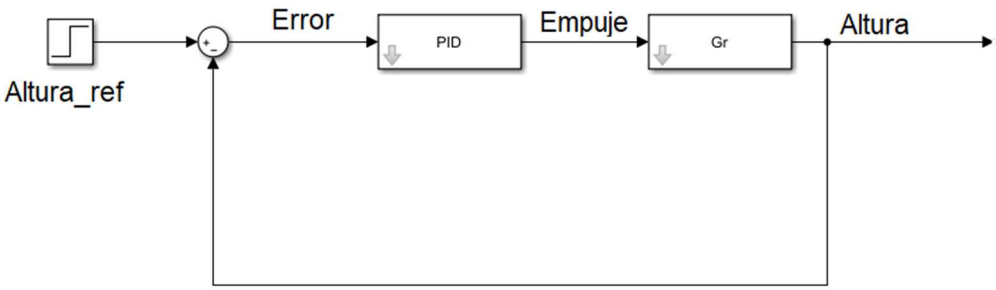


Figura 29: Diagrama de bloques del control de altura.

Para la obtención de los parámetros del regulador se ha empleado Matlab. Se optó por siguiente metodología, en primer lugar, se ha diseñado el regulador PD en tiempo continuo, a continuación, se ha convertido el regulador a tiempo discreto mediante Tustin, finalmente se ha calculado la antitransfomada del regulador. Cabe destacar, que al tratarse de un doble integrador un regulador PD conseguirá un error de posición cero respecto a la referencia. En cuanto a especificaciones del regulador, se ha decidido que este cumpla con un tiempo de establecimiento de 10 segundos.

Respecto al diseño en tiempo continuo, partimos de la expresión del regulador PD en formato ISA standard.

$$Gr(s) = K\left(1 + \frac{Td\ s}{\frac{Td}{N}s + 1}\right) \tag{73}$$

Posteriormente se obtuvieron los puntos de ruptura en función Td mediante la siguiente propiedad:

$$\frac{d}{dt}\{Gr * G\} = 0 \rightarrow \frac{d}{dt}\left\{K\left(1 + \frac{Td\ s}{\frac{Td}{N}s + 1}\right) * \left(\frac{1}{ms^2}\right)\right\} = 0 \quad (74)$$

Cuyo resultado es la posición en el lugar de las raíces donde se corta con el eje real y a su vez será donde ajustaremos el regulador:

$$sol = \frac{-23.8197}{11 * Td} \quad (75)$$

El valor Td se calcula gracias a que se ha de cumplir el criterio del tiempo de establecimiento de forma que el punto de ruptura cumpla con esta especificación.

$$\sigma = \frac{4}{te} \rightarrow sol = \frac{4}{te} \rightarrow td = \frac{num * te}{4} = 5.4136 \quad (76)$$

A continuación, se muestra el lugar de las raíces resultante:

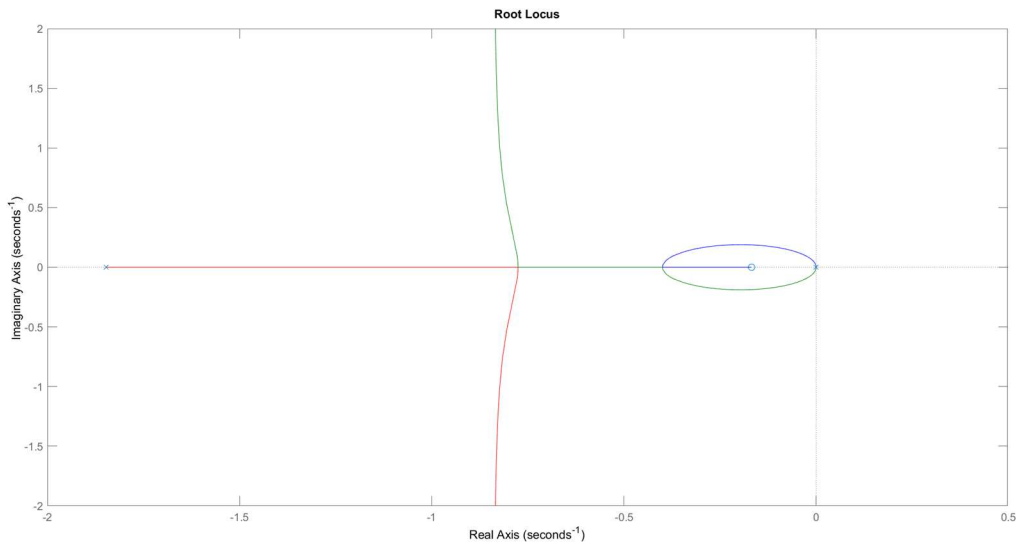


Figura 30: Lugar de las raíces del proceso controlado.

Para terminar con el diseño en tiempo continuo faltará conocer la constante del regulador, esta se obtiene de aplicar el criterio del módulo una vez ya conocemos Td:

$$K = \left|\frac{1}{Gr(s) * G(s)}\right| = 0.056 \tag{77}$$

Una vez ya tenemos el PD en tiempo continuo, se convierte a tiempo discreto mediante Tustin utilizando un periodo de muestreo adecuado cumpliendo con el teorema de Nyquist-Shannon, que dicta que el periodo de muestreo ha de ser al menos dos veces menor que el periodo de la señal muestreada.

$$T \leq \frac{te}{2} \rightarrow T \leq \frac{10\ seg.}{2} \rightarrow T = 100\ ms \tag{78}$$

El PD resultante de discretizar por Tustin con un periodo de muestreo T=100 ms:

$$PD = \frac{0.5687\ z - 0.5592}{z - 0.8309} \tag{79}$$

A continuación, se observa el comportamiento del PD en continuo frente al PD discreto para el periodo de muestreo anterior y consigna de 1 metro de altura. Se puede observar que con el periodo de muestreo de 100 ms la respuesta de ambos controles es muy similares.

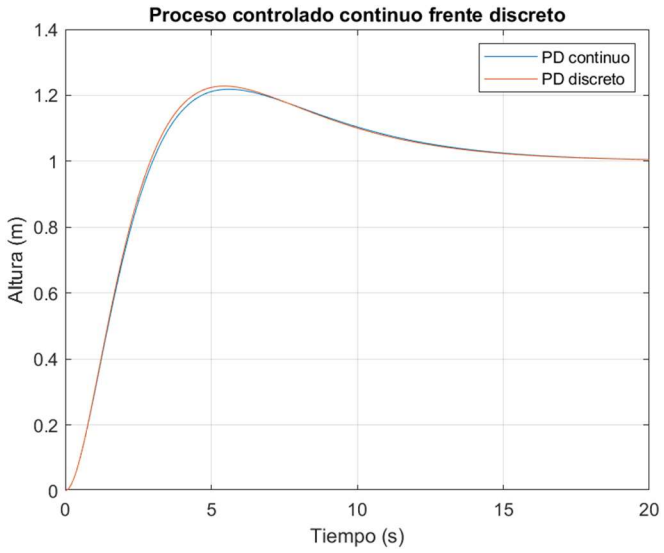


Figura 31: Control continuo vs discreto para un T=100 ms y Zref=1 m.

Por último, aplicamos la antitransfomada y obtenemos la expresión para controlar el proceso que se utilizará en el microcontrolador. Hay que tener en cuenta, que se deberá añadir el termino correspondiente al peso de la aeronave.

$$\frac{U(z)}{E(z)} = \frac{0.5687\,z - 0.5592}{z - 0.8309} \rightarrow U(z) * [z - 0.8309] = E(z) * [0.5687\,z - 0.5592] \quad (80)$$

$$U(z) * [z - 0.8309] * z^{-1} = E(z) * [0.5687\,z - 0.5592] * z^{-1} \quad (81)$$

$$uk - 0.8309\,uk_1 = 0.5687\,ek - 0.5592\,ek_1 \quad (82)$$

$$uk = 5.933 + 0.8309\,uk_1 + 0.5687\,ek - 0.5592\,ek_1 \quad (83)$$

Se ha simulado el comportamiento del sistema con el regulador y los resultados muestran cómo se cumple el error de posición cero. A su vez, el tiempo de establecimiento no es exactamente el especificado, esto es provocado por la dinámica del doble integrador.

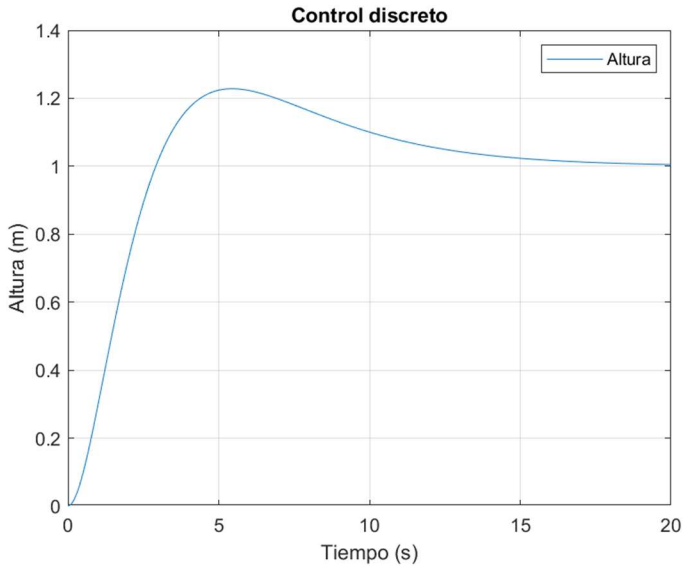


Figura 32: Comportamiento del control de altura, referencia 1 metro.

6.3.3 Control de estabilidad

El objetivo del control de estabilidad, es el de mantener el vehículo aéreo en vuelo siguiendo con las referencias de los ángulos Pitch y Roll que se establecerán a 0º para que la aeronave no vuelque en una dirección. Por lo tanto, este control corregirla las posibles derivas en algún eje para evitar que el cuadricóptero caiga. La forma de actuar sobre los ángulos Pitch y Roll es mediante una diferencia de velocidades entre los motores que generar un movimiento del UAV, este fenómeno queda explicado en apartado correspondiente a los movimientos de la aeronave.

Para el diseño del PID no utilizamos el modelo desacoplado y linealizado ya que en este caso no conocemos datos suficientes como Ixx o Iyy. Por lo que el diseño se realizó por estimación partiendo de la ecuación general del PID discreto:

$$uk = uk_1 + \left(K_p + \frac{K_i * T_s}{2} + \frac{K_d}{T_s}\right) * ek + \left(-K_p + \frac{K_i * T_s}{2} - \frac{2K_d}{T_s}\right) * ek_1 + \frac{K_d}{T_s} ek_2 \quad (84)$$

Para conseguir los valores Kp, Ki y Kd de ambos reguladores, se siguió con la siguiente metodología, basada en parte en la teoría de sintonización de PIDs de Ziegler-Nichols. En primer lugar, con el dron sujeto y en el aire, se fue incrementado en pasos de 0.2 la constante Kp y manteniendo Ki=0 y Kd=0, hasta conseguir una respuesta inestable para roll este valor era 1.4 y para pitch 1.4. Una vez conseguimos la Kp que haga el proceso inestable, se redujo su valor a la mitad.

A continuación, se repitió el mismo procedimiento, pero para Kd, manteniendo Ki=0 y el valor Kp reducido. El valor de Kd se incrementó hasta conseguir una respuesta con muchas oscilaciones y de poca amplitud. Una vez la respuesta cumplía con esta condición se redujo el valor de Kd un 25%, siendo la Kd resultante de 14.5.

Por último, manteniendo Kd y Kp con los valores anteriores se obtuvo Ki, se aumentó poco a poco hasta que el UAV siguiese las referencias correctamente y sin provocar oscilaciones, con una Ki de 0.05 se consigue un comportamiento adecuado del regulador.

Finalmente, con el dispositivo se procedió a un ajuste más fino teniendo los valores aproximados resultantes de la metodología anterior. Para ello se tubo en cuenta que implica un aumento o reducción de los valores Kp, Ki, Kd. Por ejemplo, en el caso de Kp si es muy pequeño la respuesta del PID no será lo suficientemente contundente para corregir las desviaciones. Sin embargo, un valor muy elevado producirá una respuesta inestable. En cuanto a Kd, si el valor es elevado el dron oscilará con pequeñas amplitudes, por otro lado, si es muy pequeño no responderá lo suficientemente rápido. Para Ki, si el valor es muy elevado se producirán oscilaciones como en el caso de una Kp alta.

Cabe destacar, que implica cada ganancia en el control de estabilidad. El valor de Kp corresponderá con como de agresiva será la respuesta del regulador al error entre la referencia y el valor que lee el sensor. Mientras que, el valor de Kd corresponderá con como de fuerte será la respuesta frente a cambios en la variable a controlar. Por último, el termino integral, deberá corregir las desviaciones del ángulo a lo largo del tiempo. Además, el termino integral se encargará de corregir desviaciones en el centro de gravedad o por efectos causados por las fuerzas aerodinámicas para asegurar que el error entre la referencia y la señal sea 0 (*Technik, 2023)*.

Finalmente, los reguladores resultantes tanto para pitch como para roll corresponde con el siguiente:

$$uk = Kp * ek + Ki * \sum_{j=0}^{k} ek(j) + Kd * ek_1 \tag{85}$$

$$uk = 0.75 * ek + 0.05 * \sum_{j=0}^{k} ek(j) + 14.5 * ek_1 \tag{86}$$

7. Estudio económico

Se ha realizado una estimación de los costes materiales del dispositivo diseñado. Sin embargo, se puede observar un estudio más exhaustivo de los costes en el documento de presupuestos. En el análisis de costes materiales se observa que se ha cumplido con la especificación de no superar los 350 €. Por otro lado, se observa que los ESC suponen un coste del 36% del total, estos se podrían reducir comprándolos en pack o con unas prestaciones inferiores. Por el contrario, los motores sí que se adquirieron de esa forma por lo que se redujo el coste de estos. Además, cabe destacar que se han reflejado los costes del microcontrolador y el sensor de ultrasonidos, pero estos fueron reutilizados de otros proyectos así que no se compraron para el desarrollo del prototipo, aun así, se ha reflejado su coste.

Tabla 10: Costes materiales.

Costes Materiales						
Ref	**Ud**	**Descripción**	**Precio Unitario**	**Cantidad**	**Precio Final**	**Porcentaje sobre el total**
Materiales						
m1	ud.	Arduino Nano IoT33	33,14 €	1	33,14 €	11,32%
m2	ud.	IMU BNO055	39,13 €	1	39,13 €	13,36%
m3	ud.	Giroscopo L3G4200d	5,99 €	1	5,99 €	2,05%
m4	ud.	Ultrasonidos HC-SR04	5,49 €	1	5,49 €	1,88%
m5	ud.	Fuselaje QAV250	30,88 €	1	30,88 €	10,55%
m6	ud.	Motor DX2205	12,50 €	4	49,99 €	17,07%
m7	ud.	ESC 12A BEIHELI EMAX	23,44 €	4	93,76 €	32,02%
m8	ud.	Batería LIPO 3S	19,40 €	1	19,40 €	6,63%
m9	ud.	PerfBoard	0,80 €	2	1,60 €	0,55%
m10	ud.	Interruptor	3,50 €	1	3,50 €	1,20%
m11	ud.	Interruptor Micro	0,20 €	1	0,20 €	0,07%
m12	ud.	Conector XT60	1,40 €	1	1,40 €	0,48%
m13	ud.	Terminales tipo bala	0,16 €	21	3,31 €	1,13%
m14	ud.	Hélice T5045	1,25 €	4	5,00 €	1,71%
				Total	292,79 €	100%

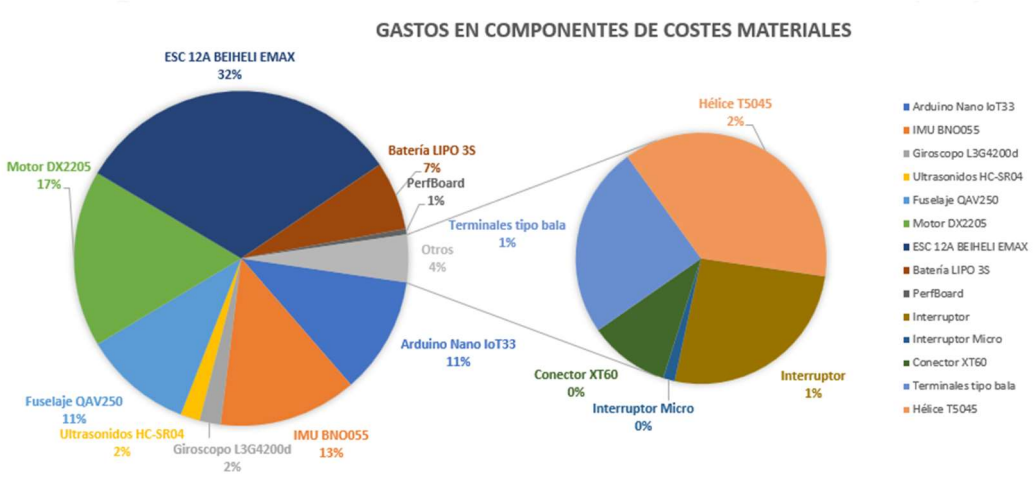


Figura 33: Grafico de costes materiales

8. Conclusiones

El objetivo del proyecto es el diseño y desarrollo de un cuadricóptero, así como el diseño e implementación de su control de altura y estabilidad.

Desde el punto de vista de la fabricación, calibración y montaje, se ha cumplido con el objetivo puesto que el prototipo es funcional. Esto se consiguió seleccionando los componentes siguiendo con unos criterios específicos. Además, fue necesario comprender que utilidad tienen en la aeronave, así como su funcionamiento los diversos componentes que la forman.

Por el lado del control, se ha conseguido realizar el control de estabilidad siendo capaz de corregir errores en los ángulos de pitch y roll, así como el control de la altura. Sin embargo, el control de estabilidad podría mejorarse ajustando sus ganancias.

Gracias a este proyecto queda demostrado que es posible la fabricación y control de un cuadricóptero basado en Arduino.

9. Posibles mejoras

A continuación, se enumeran mejoras que podrían dar lugar a trabajos futuros o bien alternativas que podrían perfeccionar el diseño.

- Implementar un control de orientación en el eje Yaw. De esta forma, permitiría poder controlar la orientación angular en el eje correspondiente.
- Implementara un control de posición en X e Y para poder controlar desplazamientos pudiendo así seguir trayectorias establecidas por el usuario.
- Implementar la conexión con una estación base para recibir y enviar datos entre estación y cuadricóptero. Así como el diseño de una interfaz con las que se podrían establecer las consignas de vuelo del vehículo aéreo.
- Realizar diseños alternativos basados en otros microcontroladores o diferentes com<ponentes.

10. Bibliografía

Adafruit (22 de abril, 2015). «Adafruit BNO055 Absolute Orientation Sensor». *Adafruit Learning System*, https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/arduino-code. Accedido 23 de junio de 2023.

HobbyKing (2 de julio, 2021). «Blog - CÓMO FUNCIONAN LOS REGULADORES ELECTRÓNICOS DE VELOCIDAD CON Y SIN ESCOBILLAS». *Hobbyking*, https://hobbyking.com/es_es/blog/brushed-brushless-electronic-speed-controllers-work/?___store=de_de. Accedido 30 de junio de 2023.

desdecero, drone, (19 de julio, 2018). «Drone Arduino | Conceptos generales sobre drones». *ArduProject.es*, https://arduproject.es/conceptos-generales-sobre-drones/. Accedido 10 de junio de 2023.

FpvMax (21 de diciembre, 2016). «Variador electrónico (ESC): Qué es y cómo funciona». *FpvMax*, https://www.fpvmax.com/uncategorized/variador-electronico-esc-funciona/. Accedido 20 de mayo de 2023.

Delgado, V (29 mayo, 2016). «Historia de los drones». *El Drone*, http://eldrone.es/historia-de-los-drones/. Accedido 11 de mayo de 2023.

Perez, J. (19 de octubre, 2022).. *RPA, RPAS, UAS y UAV: ¿Qué son y en qué se diferencian?* Umilesgroup,, https://umilesgroup.com/rpas-uas-uav-diferencias/. Accedido 10 de mayo de 2023.

Miranda Colorado, Roger. *Drones: modelado y control de cuadricópteros*. Marcombo, 2020.

Rosa. (10 de diciembre, 2020) «Tipos de Drones y Nueva Clasificación Según Peso». *DRONES MÁLAGA*, https://www.dronesmalaga.net/en/normativa/tipos-de-drones-nueva-clasificacion-segun-mtow/. Accedido 11 de mayo de 2023.

Technik (30 de enero, 2023) Setting the PID controller of a drone properly. https://www.technik-consulting.eu/en/optimizing/drone_PID-optimizing.html. Accedido 23 de junio de 2023.

.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje e implementación del control de altura de

un cuadricóptero

Anexo 1: Principios de vuelo en cuadricópteros.

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Fontalba Roca, Marc

Tutor/a: González Sorribes, Antonio

CURSO ACADÉMICO: 2022 / 2023

Índice Anejo 1:

1. Introducción

En este documento se va a realizar un estudio de los posibles movimientos que pueden ejecutar los vehículos aéreos tipo cuadricóptero, y como han de comportarse los actuadores para llevarlos a cabo.

2. Fuerzas Implicadas

Antes de poder analizar el comportamiento de los cuadricópteros, hay que conocer las principales fuerzas implicadas en el movimiento estos, que son, el peso, el empuje y el arrastre o fricción. En cuanto al peso, es la fuerza producida por el efecto de la gravedad sobe el vehículo aéreo. Por otro lado, el empuje es la suma de las fuerzas ejercidas por cada uno de los motores y hélices, ya que los motores ejercerán un movimiento mecánico rotacional y las hélices transformarán esta rotación en una fuerza de sustentación por movimiento de masas de aire, este efecto se basa en el principio de Bernouilli. De forma que esta fuerza, deberá compensar el peso en magnitud, dirección y sentido para que la aeronave sea capaz de volar. Por último, está el arrastre que se opondrá al movimiento que efectúe el dron debido al rozamiento con el aire. También, se deberían considerar las fuerzas y perturbaciones provocadas por el propio efecto del movimiento del aire, pero en condiciones ideales no se tienen en cuenta.

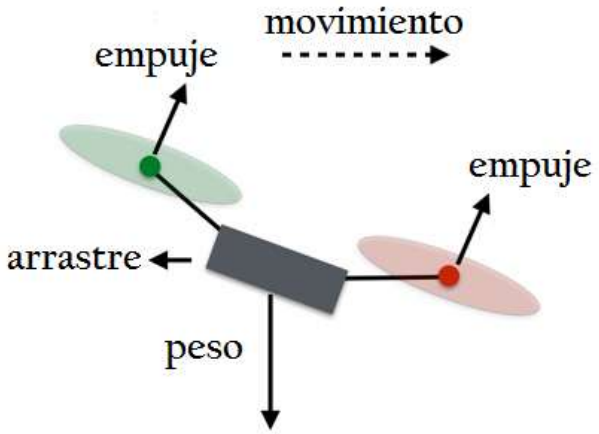


Figura 34: Fuerzas implicadas en el UAV.

Nota. De «Fundamentos»,F. Sanchez, (s.f),HazTuDron . Derechos de autor por HazTuDrON.

3. Movimientos generados por los actuadores

A continuación, se procederá a explicar los movimientos de cuadricópteros como producto de la rotación de los motores. En este apartado se analizarán, los desplazamientos y rotaciones en los ejes Roll, Pitch, Yaw. Por lo que respecta al resto de los movimientos posibles, no serán más que la combinación de los anteriormente mencionados. Sin embargo, antes de comenzar cabe destacar que basaremos la explicación en los vehículos en configuración en X y cuya distribución de motores corresponde con la siguiente figura.

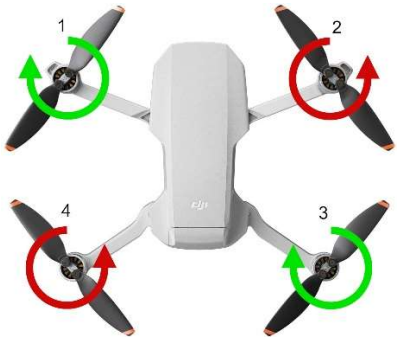


Figura 35: Configuración de motores.

Nota. Adaptado de «Catalogo de Amazon», DJI, (s.f), Amazon. CC-BY-NC.

Como podemos observar, los motores 1 y 3 rotan en sentido horario o CW, mientras que 2 y 4 giran en sentido antihorario o CCW. Para poder realizar maniobras, se deberá actuar sobre la velocidad de giro de cada motor.

3.1 Movimientos en el eje vertical

El desplazamiento en el eje vertical corresponde con la elevación o descenso del dron únicamente en eje perpendicular al suelo, entendiendo este como una superficie plana sin ningún tipo de inclinación. Además, se asume que el centro de gravedad de la aeronave corresponde con el centro simétrico del cuerpo del UAV. Para el desplazamiento de subida, la fuerza de empuje, total de la suma de fuerzas ejercidas por cada motor, deberá ser mayor que el peso de la aeronave como bien se ha explicado con anterioridad. Por el contrario, el movimiento de bajada es producto de que la fuerza ejercida por los motores es menor al peso. Cabe destacar que, en ambos movimientos, los actuadores han de girar a la misma velocidad, si no el desplazamiento no será únicamente en el eje vertical o se podrían producir rotaciones sobre el propio eje vertical.

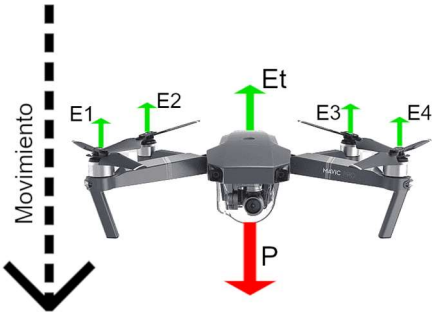


Figura 36: Fuerzas en el desplazamiento de bajada.

Nota. Adaptado de «Mavic Pilots», DJI, (s.f), Mavic Pilots. CC-BY-NC.

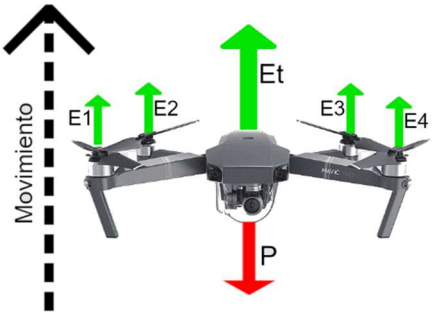


Figura 37:Fuerzas en el desplazamiento de subida.

Nota. Adaptado de «Mavic Pilots», DJI, (s.f), Mavic Pilots. CC-BY-NC.

Es mediante este desplazamiento vertical con el balance del empuje y el peso en el que se basa el control de altura ya que este proceso controlará la velocidad de giro de los motores para conseguir elevar o descender el cuadricóptero de forma que se alcance la referencia de altura deseada.

3.2 Rotación en el eje vertical o Yaw

Las rotaciones en el eje vertical o Yaw, se realizan mediante una diferencia de velocidades en los motores del UAV. Esta diferencia, genera un contrapar que produce el efecto de rotacional en este eje. Para conseguir este efecto, los motores deberán girar de forma que las parejas de motores 1 y 3 o 2 y 4, tengan mayor velocidad que la otra. Es decir, si los motores 1 y 3 rotan a mayor velocidad que 2 y 4 se producirá un giro en Yaw en sentido opuesto al de rotación de 1 y 3, en sentido CCW. Por el contrario, si 2 y 4 son los que van a más velocidad el giro resultante será en sentido CW.

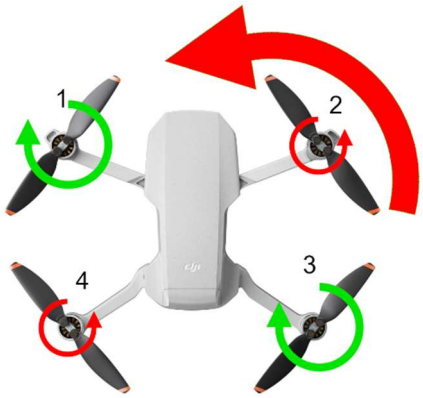


Figura 38: Rotación en Yaw, sentido CCW.

Nota. Adaptado de «Catalogo de Amazon», DJI, (s.f), Amazon. CC-BY-NC.

Figura 39: Rotación en Yaw, sentido CW.

Nota. Adaptado de «Catalogo de Amazon», DJI, (s.f), Amazon. CC-BY-NC.

3.3 Desplazamiento en Roll y rotación en Pitch

La rotación en Pitch, se define como un giro realizado por la aeronave alrededor del eje imaginario paralelo al eje que se forman atravesando los motores 1 y 4 o 2 y 3, y que pasa por el centro del eje simétrico del vehículo aéreo. Por otro lado, el eje Roll es el paralelo a los ejes que unen los motores 1 y 2 o 3 y 4 de forma que este atraviese el centro de la aeronave.

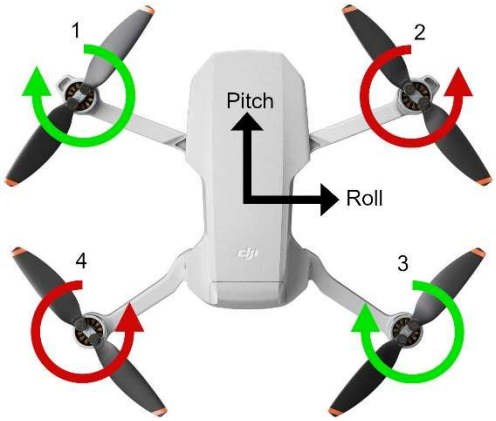


Figura 40: Ejes Roll y Pitch en la aeronave.

Nota. Adaptado de «Catalogo de Amazon», DJI, (s.f), Amazon. CC-BY-NC.

Por lo que tanto, la rotación en Pitch se podrá conseguir con un aumento en la velocidad de giro de las parejas de motores 1 y 4 o 2 y 3, respecto a la otra pareja. Por lo que, la

fuerza resultante de este desequilibrio en los actuadores provocará una rotación alrededor del eje.

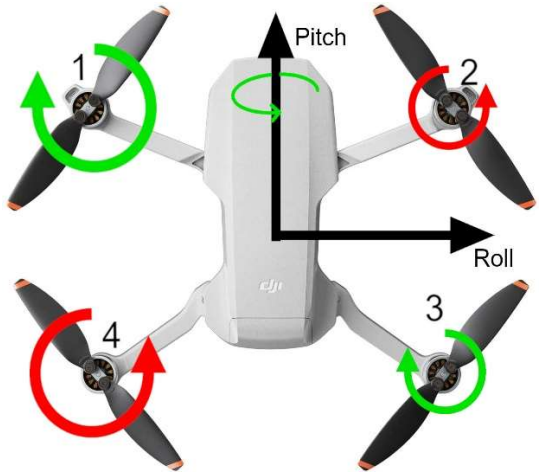


Figura 41: Rotación positiva en Pitch.

Nota. Adaptado de «Catalogo de Amazon», DJI, (s.f), Amazon. CC-BY-NC.

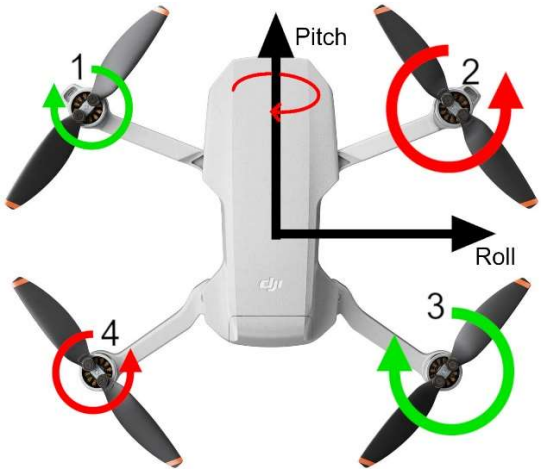


Figura 42: Rotación negativa en Pitch.

Nota. Adaptado de «Catalogo de Amazon», DJI, (s.f), Amazon. CC-BY-NC.

Por lo que respecta al control de estabilidad, el comportamiento de este se basa en parte, en esta rotación. Esto se debe, a que para corregir las desviaciones del ángulo pitch deseado será necesario cambiar la velocidad de giro de los motores que compensen el error angular.

Por otro lado, al producirse el giro en el Pitch, la fuerza del empuje de los motores no será perpendicular al suelo si no que tendrá la misma inclinación que el ángulo girado. Por lo tanto, el empuje podrá descomponerse en dos una perpendicular al peso que mantendrá en aeronave en el aire y otra en el eje Roll que provocará un desplazamiento a lo largo de este.

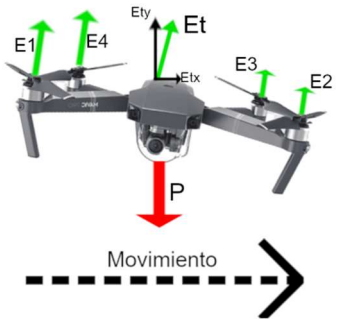


Figura 43: Desplazamiento positivo en Roll.

Nota. Adaptado de «Mavic Pilots», DJI, (s.f), Mavic Pilots. CC-BY-NC.

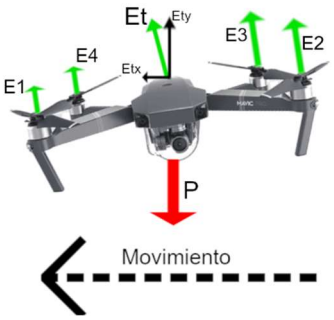


Figura 44.Desplazamiento negativo en Roll

Nota. Adaptado de «Mavic Pilots», DJI, (s.f), Mavic Pilots. CC-BY-NC.

.

3.4 Desplazamiento en Pitch y rotación en Roll

En el caso de la rotación en Roll, se conseguirá mediante un desequilibrio como en la rotación en Pitch, pero en este caso los motores deberán emparejarse de forma que 1 y 4 o 3 y 2, sen los que tengan una velocidad diferente al de la otra pareja. Y el desplazamiento resultante será igual que en el caso anterior, pero en el eje Pitch.

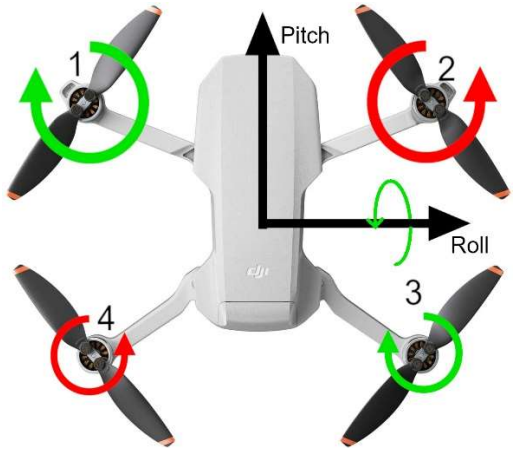


Figura 45: Rotación positivo en Roll.

Nota. Adaptado de «Catalogo de Amazon», DJI, (s.f), Amazon. CC-BY-NC.

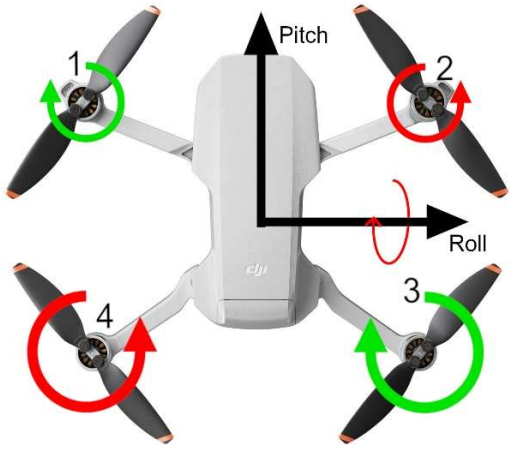


Figura 46: Rotación negativa en Roll.

Nota. Adaptado de «Catalogo de Amazon», DJI, (s.f), Amazon. CC-BY-NC.

Al igual que en el caso de la rotación en Pitch, es mediante esta rotación que el control de estabilidad es capaz de corregir el error angular en Roll.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje e implementación del control de altura de

un cuadricóptero

Anexo 2: Control y calibración de los motores mediante ESCs.

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Fontalba Roca, Marc

Tutor/a: González Sorribes, Antonio

CURSO ACADÉMICO: 2022 / 2023

Índice Anexo 2:

1. ESCs

Los variado de velocidad son componentes electrónicos diseñados para el control de la velocidad de giro de motores eléctricos, algunos para motores brushed y otros para brushless. En este documento, nos centraremos en estos últimos ya que son los que se emplean en el dispositivo diseñado.

En primer lugar, se va ha desarrollar en profundidad la explicación de las entradas, salidas y funcionamiento de los ESCs, ya que estos presentan un papel fundamental en el del dispositivo, por lo que será crucial entender su comportamiento.

1.1 Salidas

Respecto a los ESCs para el control de motores brushless, tendrán tres salidas una para cada fase del motor. Con estas salidas, podrá controlar la velocidad de los motores energizando dos de las tres fases del motor y utilizando la otra para conocer información del giro mediante señales provocadas por la propia rotación. De esta forma, es capaz de saber cómo tiene que energizar las salidas para que el motor siga la consigna dada.

Por otro lado, dependiendo de cómo se conecte el ESC al motor, este girará en un sentido u en otro, es decir en configuración CW o CCW. Para el sentido de giro en CW, habrá que conectar las salidas a las tres fases del motor de manera directa, es decir, que corresponda la posición de la salida con la fase de entrada del motor. En cambio, para el sentido de giro en configuración CCW habrá que conectar dos fases cruzadas entre ellas.

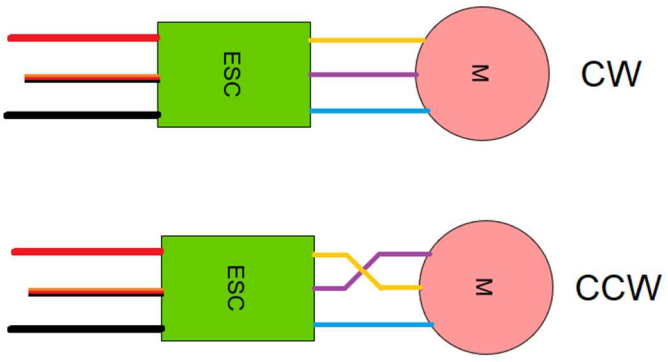


Figura 47: Configuración de giro con ESC.

Aunque algunos ESCs, como los que se han empleado en el prototipo, permiten cambiar el sentido de giro sin necesidad de cruzar las fases. Esto se consigue mediante software, ya que son programables y el sentido de giro es un aspecto de los que el fabricante permite configurar. Entraremos más en detalle en la programación de los ESC más adelante.

1.2 Entradas

Respecto a las entradas, consta de tres, dos serán para la alimentación, es decir para masa y para el positivo, y la entrada restante, corresponde con una de tipo servo, que permitirá tanto el control como la programación del variador. Por otra parte, el cable tipo servo tendrá tres pines, naranja, rojo y marrón, el naranja es por donde enviaremos la señal de control, el rojo es la salida del BEC del ESCs y proporcionará +5V, pero no la utilizaremos y finalmente el marrón corresponde con la masa.

1.3 Señales de control

En cuanto al control de los ESC, este se lleva a cabo mediante el empleo de señales PWM de un ancho concreto. Las señales de modulación de ancho de pulsos se caracterizan por tener dos intervalos dentro de su periodo, un intervalo a nivel alto, llamado Ton, y otro a nivel bajo, Toff. En el caso de los ESC la se señal deberán tener un Ton de 1 a 2 milisegundos siendo 1 ms el equivalente al 0% de la potencia y los 2 ms al 100%.

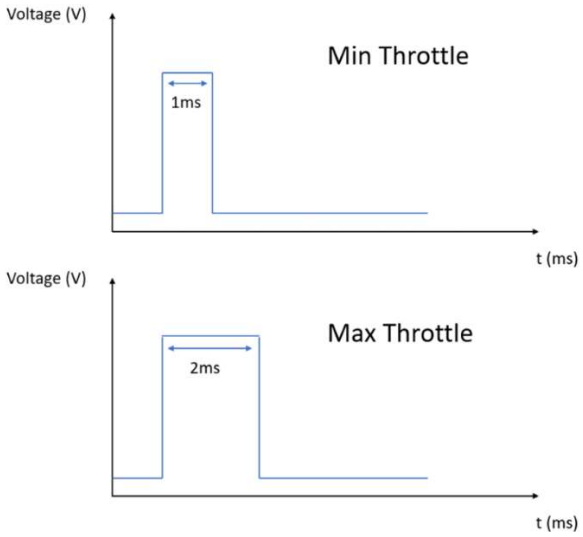


Figura 48: Intervalo de la PWM en un ESC.

Nota. De «What is an ESC and How does ESC work», L. Nagel, 2022, Tyto Robotics, Derechos de Autor de Tyto Robotics.

Para la generación de estas señales mediante el Arduino existen varias opciones:

- Una alternativa, es emplear la librería que viene preinstalada en el compilador para control de servos y manejar el ESC con las PWM que se pueden generar con la librería. Esta opción es bastante sencilla de implementar el único

inconveniente es que no podemos cambiar el periodo de la señal PWM que tiene un ciclo de 20 ms por lo que estaríamos perdiendo como mínimo 18 ms en los que no podríamos hacer cambios en la velocidad.

- Otra opción, es emplear los pines PWM del microcontrolador que trabajan a una frecuencia de 490 Hz, aproximadamente 2 ms de ciclo, y emplear el comando "analogWrite()". Este comando, genera una PWM en función de un valor de entrada entre 0 y 255. El principal problema de esta alternativa es que limita la velocidad en 128 valores ya que la señal más pequeña que acepta el ESC corresponde con la mitad del ciclo, es decir 1 ms. Cabe destacar que este inconveniente se podría solucionar cambiando la frecuencia de reloj de los pines actuando sobre los registros del Arduino. Sin embargo, estos cambios podrían afectar a las funciones de temporización, algo no deseable a la hora de controlar otros procesos.

- Por último, existe la posibilidad de generar nuestras propias señales PWM de manera manual. Esto se puede conseguir poniendo los pines a nivel alto o nivel bajo durante el tiempo correspondiente en cada uno de los ciclos de control haciendo uso de las funciones de temporización. Esta solución permite una solución más optimizada, pero en situaciones donde el tiempo de muestreo sea superior a los 20 ms tampoco supondrá una ventaja respecto a la primera alternativa.

1.4 Calibración

Previo a que el dispositivo comience a poner en marcha los motores en cada vuelo se deberá realizar la calibración de los rangos de tensiones de la señal PWWM que recibirá el ESC. Es decir, antes de que el dispositivo comience a volar, será necesario enviar al variador la señal correspondiente al 100% y al 0% de la potencia. Este proceso ha de realizarse conforme a la siguiente figura:

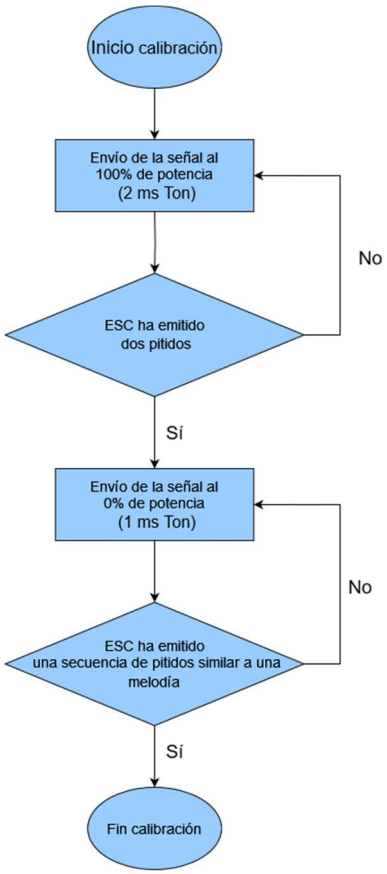


Figura 49: Diagrama de flujo calibración.

Por lo tanto, siguiendo el diagrama anterior la secuencia será enviar la señal correspondiente a la máxima que espera recibir el ESC, esperar a recibir dos pitidos. Posteriormente, se envía la señal mínima y se espera a escuchar la melodía que describe el fabricante en el datasheet del ESC. Una vez se ha cumplido esto ya se habrá terminado la calibración del rango de velocidades del variador y ya estará listo para recibir las señales de velocidad de giro de los motores.

Para la calibración de los motores se ha utilizado la siguiente función:

```
#include <Servo.h>
#define ESC_pin1 12
#define ESC_pin2 9
#define ESC_pin3 7
#define ESC_pin4 5
Servo ESC_1, ESC_2, ESC_3, ESC_4;

void motor_calib(){

  //Calibra el rango de entrada de los ESC
  ESC_1.attach(ESC_pin1,min_pul,max_pul);
  ESC_2.attach(ESC_pin2,min_pul,max_pul);
  ESC_3.attach(ESC_pin3,min_pul,max_pul);
  ESC_4.attach(ESC_pin4,min_pul,max_pul);

  //1ro Envia la señal correspondiente al valor máximo
  //Serial.println("Estableciendo limite superior de la señal PWM");
  ESC_1.writeMicroseconds(max_pul);
  ESC_2.writeMicroseconds(max_pul);
  ESC_3.writeMicroseconds(max_pul);
  ESC_4.writeMicroseconds(max_pul);
  delay(4000);
  //2nd Envia la señal correspondiente al valor mínimo
  //Serial.println("Estableciendo limite inferior de la señal PWM");
  ESC_1.writeMicroseconds(min_pul);
  ESC_2.writeMicroseconds(min_pul);
  ESC_3.writeMicroseconds(min_pul);
  ESC_4.writeMicroseconds(min_pul);
  delay(4000);

}
```

Figura 50: Función de calibración de motores.

1.5 Programación de ESC

Algunos fabricantes permiten cambiar características de los ESCs, estos parámetros son configurables con un Arduino de una manera similar a la calibración. En caso de querer programar el variador habrá que enviar la señal de 2 ms o 1 ms para ir seleccionando la configuración deseada, este proceso esta explicado en el datasheet del ESC. Los pasos, deberán ser los siguientes:

1. Enviar la señal máxima hasta entrar en modo programación. Se sabrá porque el ESC emitirá una melodía después de haber emitido los dos primeros pitidos del primer paso de la calibración.

2. A partir de este momento, el variador irá emitiendo pitidos de una longitud y numero concreto que determina que parámetro se está cambiando. Cuando el pitido corresponda con el parámetro que se quiere configurar se enviará una

señal de 1 ms. Entonces el ESC volverá a pitar ahora harán referencia a los posibles valores del parámetro a configurar, por ejemplo, sentido de giro un pitido sentido normal, dos pitidos sentido opuesto.

3. Una vez se haya emitido el sonido correspondiente al parámetro deseado se enviará una señal de 2 ms indicando que ese valor es el que queremos. Entonces, el ESC hará una melodía que informa que se ha guardado el valor.

4. Por último, nada más emitir la melodía anterior se enviará una señal de 1ms. En este momento, el variador indicará que toda la configuración se ha guardado y que la señal de mínima potencia se ha registrado. Estando así listo para empezar a volar.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje e implementación del control de altura de

un cuadricóptero

Anexo 3: Código empleado.

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Fontalba Roca, Marc

Tutor/a: González Sorribes, Antonio

CURSO ACADÉMICO: 2022 / 2023

Índice Anexo 3:

1. Código Matlab

El código empleado en Matlab se utiliza para la obtención de los parámetros del regulador del control de altura tal como se explica en el apartado de soluciones adoptadas.

´

1.1 Diagrama de flujo

El código sigue una secuencia como la que muestra el siguiente diagrama de flujo.

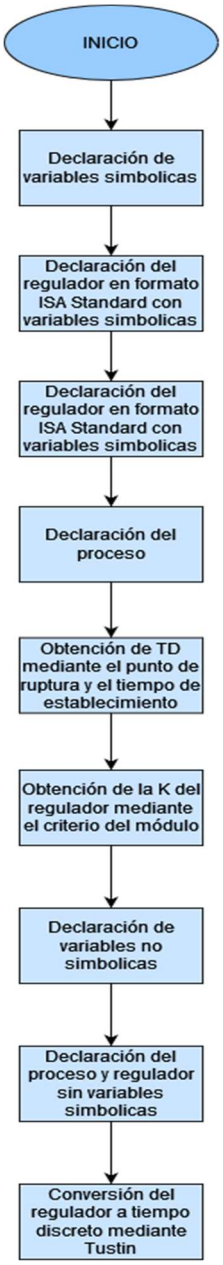


Figura 51: Diagrama de flujo del código Matlab.

1.2 Código

```
%Obtención de la TD y K del regulador PD.

%Definición de variables
syms x td k;
m=0.605; %Masa UAV en kg
PDsys=k*(1+td*x/((td/10)*x+1)); %Regulador PD en formato ISA
Standard
Gsys=1/(m*x^2); %Proceso a controlar

%Obtención del polo que cumpla el criterio del tiempo de
establecimiento
%mediante el punto de ruptura

%Obtención del punto de ruptura en función de td
sol=solve(diff(Gsys*PDsys,x)==0,x);

%Obtención del valor de td que cumpla con el criterio de
tiempo de establecimiento
TD=double(solve(sol(1)==-0.4,td));

%Obtención de la K del regulador mediante el criterio del
módulo
P_R=subs(PDsys,[k,td,x],[1,TD,-1]);
G_R=subs(Gsys,x,-1);
K_R=double(abs(1/(G_R*P_R)));%Obtención de la constante del
regulador


%Conversión del regulador continuo a regulador discreto

%Definición de variables, proceso y regulador
s=tf('s');
Gr=1/(m*s^2);%Proceso
g=9.807;
Peso=m*g;
PID=K_R*(1+TD*s/((TD/10)*s+1));%Regulador continuo


%Tiempo de muestro y conversión a discreto del PD
T=2/20; %Periodo de muestreo
GD=c2d(PID,T,'Tustin'); %Conversión a discreto mediante
Tustin
```

1.3 Simulación

Para el estudio del comportamiento del proceso controlado se empleo el siguiente montaje en simulink. En esta simulación se puede obsrvar tanto la respuesta del proceso como la comparación entre utilizar el regulador continuo y el discreto.

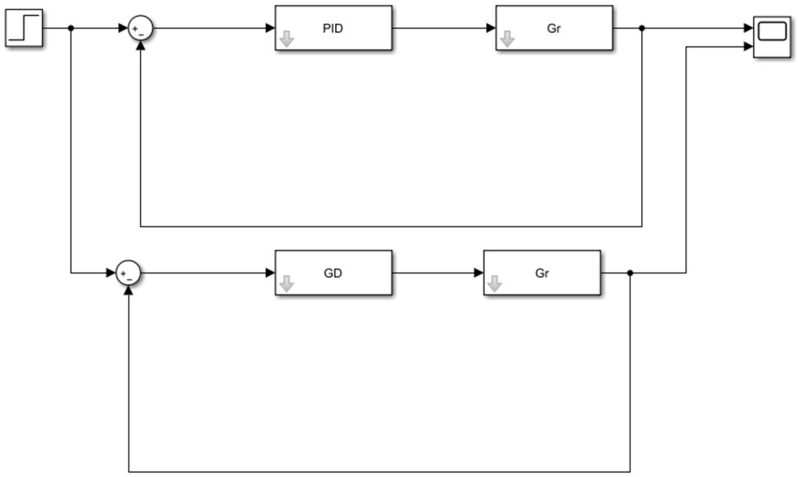


Figura 52: Simulación del control de altura con regulador continuo y discreto.

2. Código Principal

El código principal consta de varias funciones que se explicarán más detalladamente más adelante

2.1 Diagrama de flujo

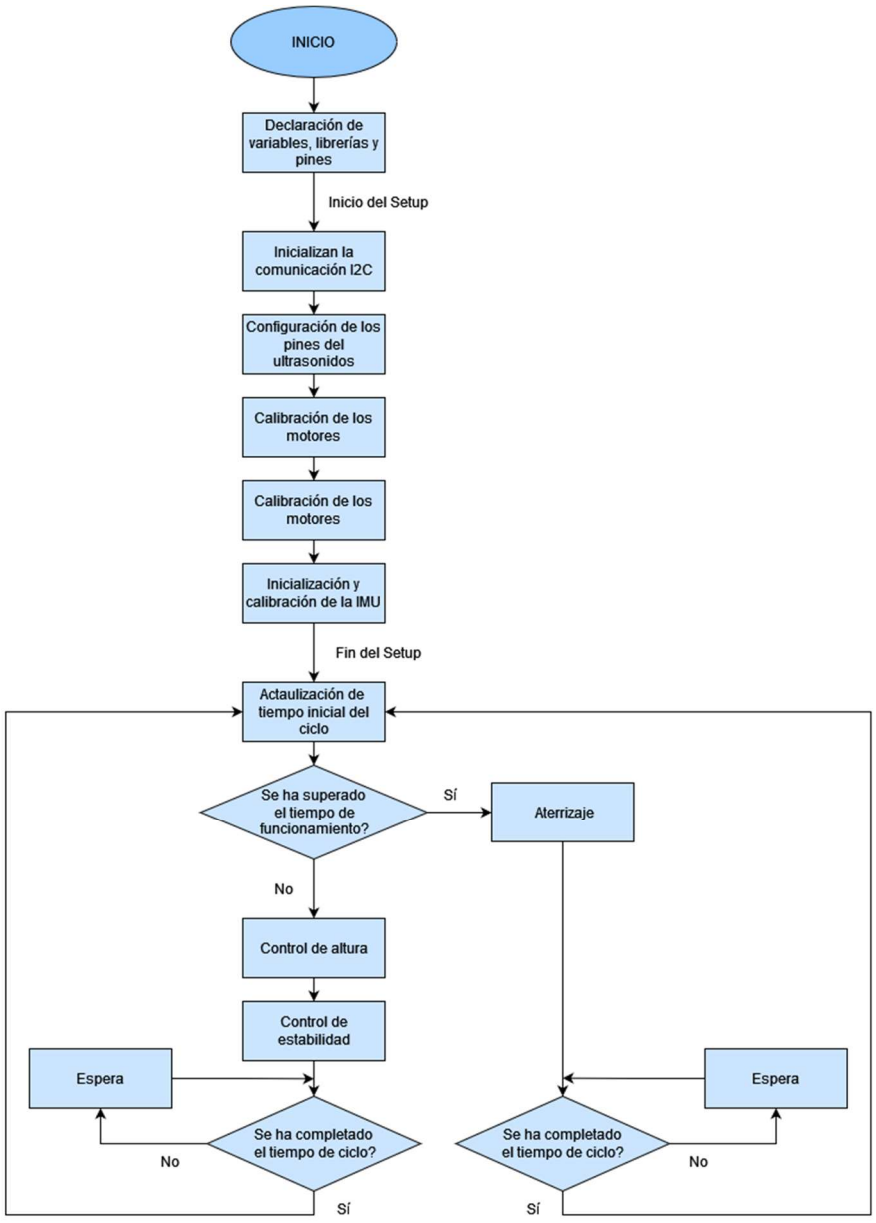


Figura 53: Diagrama de flujo del programa principal.

2.2 Código completo

```
//Librerías empleadas
#include <Servo.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>

//Definición de pines y direcciónes

#define Trig 2
#define Echo 3
#define ESC_pin1 12
#define ESC_pin2 9
#define ESC_pin3 7
#define ESC_pin4 5
Servo ESC_1, ESC_2, ESC_3, ESC_4;



// Declarción de variables
Adafruit_BNO055 bno = Adafruit_BNO055(55);
float Roll, Pitch, Yaw, Roll_cal, Pitch_cal, Yaw_cal;//Variables de
posicion angular y su valor inicial
float WRoll, WPitch, WYaw, WRoll_cal, WPitch_cal, WYaw_cal;
  //Variables para definir la velocidad de giro en los motores
const int min_pul=1000; //Ancho de pulso en us para un 0% de
velocidad de los motores
const int max_pul=2000; //Ancho de pulso en us para un 100% de
velocidad de los motores
int pulso=0; //Ancho de pulso que se le enviará a los motores
int pulsoE1,pulsoE2,pulsoE3,pulsoE4; //Pulso individual en cada
motor

   //Variables para el regulador de altura
float Pmax=4*11.1*12*0.001*4.45; //9.81*4*11.1*12*0.001*4.3 //Empuje
máximo capaz de ejercer la aeronave
float altura=0.04; //Varaible a controlar de la altura
float zref=0.1; //Consigna de altura
float peso_nave=0.605;//0.605*9.81; //Peso de la aeronave
float uk,ek,uk1,ek1; //Variables del regulador discreto

  //Variables de muestreo del tiempo
long t0=0; //Variable del tiempo inicial del periodo de muestreo
long tm=0; //Variable del tiempo final del periodo de muestreo
const int T=25; //Periodo de muestreo
int num_ciclos=0; //Numero de ciclos ejecutados

  //Variable para el regulador de estabilidad
float roll_PID,roll_PID_1,er_roll,er_roll_1,er_roll_2; //Variables
del regulador discreto del angulo roll
float pitch_PID,pitch_PID_1,er_pitch,er_pitch_1,er_pitch_2;
//Variables del regulador discreto del angulo pitch
float yaw_PID,yaw_PID_1,er_yaw,er_yaw_1,er_yaw_2;
float PID_pitch_sat=150; //Valor de saturción del regulador en pitch
```

```
float PID_roll_sat=150; //Valor de saturción del regulador en roll
float PID_yaw_sat=50;
float Roll_ref=0; //Consigna en roll
float Pitch_ref=0.2; //Consigna en pitch
float Yaw_ref=0;




//Inicio del Setup
void setup() {

  //Inicialización de la comunicación por I2C
  Wire.begin();

 //Inicialización de la comunicación serie
  //Serial.begin(115200);


 //Inicialización de sensor de altura
 altimetro_inicializar();
    //Serial.println("Lector de distancia incializado");

  //Calibración de motores
  motor_calib();
    //Serial.println("Motores calibrados");

  //Iniciar IMU
    //Serial.println("IMU ini");
  IMU_Ini();
    //Serial.println("IMU ini fin");



  /*
  Serial.print("Altur");
  Serial.print("\t");
  Serial.print("dt");
  Serial.print("\t");

  Serial.print("Pitch");
  Serial.print("\t");
  Serial.print("Roll");
  Serial.println("\t");

  Serial.print("PID_P");
  Serial.print("\t");
  Serial.print("PID_R");
  Serial.print("\t");
  Serial.print("E1");
  Serial.print("\t");
  Serial.print("E2");
  Serial.print("\t");
  Serial.print("E3");
  Serial.print("\t");
  Serial.println("E4");
```

```
  Serial.print("\t");
  Serial.println("Tm");
  */

}


float pitch_P, pitch_I, pitch_D;
float roll_P, roll_I, roll_D;
float yaw_P, yaw_I, yaw_D;

//Bulce Principal
void loop() {
  //Inicio del ciclo
    t0=millis();
    num_ciclos=num_ciclos+1;

  //Control de altura
    control_altura();

  //Control de estabilidad:
    control_estabilidad();

  //Actualizar salidas

    ESC_1.writeMicroseconds(pulsoE1);
    ESC_2.writeMicroseconds(pulsoE2);
    ESC_3.writeMicroseconds(pulsoE3);
    ESC_4.writeMicroseconds(pulsoE4);

/*
  Serial.print("X: ");
  Serial.print(Yaw, 4);
  Serial.print("\tY: ");
  Serial.print(Pitch, 4);
  Serial.print("\tZ: ");
  Serial.print(Roll, 4);
  Serial.println("");
  */
/*
    Serial.print(pulso);
    Serial.print("\t");
    Serial.print(altura);
    Serial.print("\t");


    Serial.print(Pitch);
    Serial.print("\t");
    Serial.print(pitch_P);
    Serial.print("\t");
    Serial.print(pitch_I);
    Serial.print("\t");
    Serial.print(pitch_D);
    Serial.print("\t");


    Serial.print(Roll);
```

```
    Serial.print("\t");
    Serial.print(roll_P);
    Serial.print("\t");
    Serial.print(roll_I);
    Serial.print("\t");
    Serial.print(roll_D);
    Serial.print("\t");

    Serial.print(pitch_PID);
    Serial.print("\t");
   // Serial.print(roll_PID);
   // Serial.print("\t");
    //Serial.print(yaw_PID);
    //Serial.print("\t");
    Serial.print(pulsoE1);
    Serial.print("\t");
    Serial.print(pulsoE2);
    Serial.print("\t");
    Serial.print(pulsoE3);
    Serial.print("\t");
    Serial.println(pulsoE4);
   */

  //Espera hasta finalizar el bucle
    tm=millis();
    /*
    //Serial.print("Tm:");
    Serial.print("\t");
    Serial.println((tm-t0));
    */
    while(tm-t0<T)tm=millis();

}

//float K1p=0.6153 , K2p=14.2, K3p=13.59;
//float K1r=0.6153 , K2r=14.2, K3r=13.59;
//float K1p=0.6153 , K2p=15.77, K3p=15.1;
//float K1p=0.6153 , K2p=15.25, K3p=14.6; 0.29
//float K1p=0.6153 , K2p=5.784, K3p=5.538;
float K1p=0.6153 , K2p=4.206, K3p=4.028;
float K1r=0.6153 , K2r=4.206, K3r=4.028;
//float pitch_KP=1.4, pitch_KI=0.000001, pitch_KD=100000;
//float pitch_KP=1.4, pitch_KI=0.03, pitch_KD=10;
//float roll_KP=1.5, roll_KI=0.03, roll_KD=10;

//KD=9; 12*0.75
//0.2
float pitch_KP=0.75, pitch_KI=0.04, pitch_KD=14.5; //1.6(0.8) 0.0
22(13.5))
//float pitch_KP=0, pitch_KI=0, pitch_KD=0; //
float roll_KP=0.75, roll_KI=0.04, roll_KD=14.5;//1.4(0.7) 0.02
18(13.5)
//float roll_KP=0, roll_KI=0, roll_KD=0;//
float yaw_KP=0, yaw_KI=0., yaw_KD=0;

float I_pitch_sat=30, I_roll_sat=30, I_yaw_sat=15;
```

```
float q0p=  pitch_KP * (1 + (T*pitch_KI)/(2*1000) +
1000/(pitch_KD*T));
float q1p= -pitch_KP * (1 - (T*pitch_KI)/(2*1000) +
2*1000/(pitch_KD*T));
float q2p=  pitch_KP*1000/(pitch_KD*T);
float q0r=  roll_KP * (1 + (T*roll_KI)/(2*1000) + 1000/(roll_KD*T));
float q1r= -roll_KP * (1 - (T*roll_KI)/(2*1000) +
2*1000/(roll_KD*T));
float q2r=  roll_KP*1000/(roll_KD*T);

void control_estabilidad(){
  //Lectura de la IMU

  IMU_read_abspos();


  //Calculo de los ángulos

  Yaw=Yaw-Yaw_cal;
  Roll=Roll-Roll_cal;
  Pitch=Pitch-Pitch_cal;


  //PID Pitch

   //Caluclo del error en Pitch
    er_pitch=Pitch-Pitch_ref;

    //Calculo de la acción de control
   //pitch_PID=pitch_PID_1 +
(pitch_KP+pitch_KI*T/(2*1000)+pitch_KD*1000/T)*er_pitch+(-
pitch_KP+pitch_KI*T/(2*1000)-
2*pitch_KD*1000/T)*er_pitch_1+(pitch_KD*1000/T)*er_pitch_2;
    //pitch_PID=pitch_PID_1 +
q0p*er_pitch+q1p*er_pitch_1+q2p*er_pitch_2;
    //pitch_PID=pitch_PID_1 +
q0p*er_pitch+q1p*er_pitch_1+q2p*er_pitch_2;
    //pitch_PID= K1p*pitch_PID_1 + K2p*er_pitch -
K3p*er_pitch_1    pitch_P= pitch_KP*(er_pitch);
    pitch_P= pitch_KP*(er_pitch);
    pitch_I+=  pitch_KI*(er_pitch);
    pitch_I=constrain(pitch_I,-I_pitch_sat,I_pitch_sat);
    pitch_D= pitch_KD*(er_pitch-er_pitch_1);
    pitch_PID=pitch_P+pitch_I+pitch_D;
    pitch_PID=constrain(pitch_PID,-PID_pitch_sat,PID_pitch_sat);
//Limitación de la señal de control

    //Actualización de los valores del regulador
    pitch_PID_1=pitch_PID;
    er_pitch_2=er_pitch_1;
    er_pitch_1=er_pitch;


/*
    if(er_pirch>Pitch_ref+0.5){
      PID_pitch=5;
    }
```

```
    else if(er_pitch<Pitch_ref-0.5){
      PID_pitch=-5;
    }
    else{
      PID_pitch=0;
    }
    */
  //PID Roll

    //Calculo del error en Roll
    er_roll=Roll-Roll_ref;

    //Calculo de la acción de control
    //roll_PID=roll_PID_1 +
(roll_KP+roll_KI*T/(2*1000)+roll_KD*1000/T)*er_roll+(-
roll_KP+roll_KI*T/(2*1000)-
2*roll_KD*1000/T)*er_roll_1+(roll_KD*1000/T)*er_roll_2;
    //roll_PID=roll_PID_1 + q0r*er_roll + q1r*er_roll_1 +
q2r*er_roll_2;
    //roll_PID= K1r*roll_PID_1 + K2r*er_roll - K3r*er_roll_1;
    roll_P= roll_KP*(er_roll);
    roll_I+= roll_KI*(er_roll);
    roll_I=constrain(roll_I,-I_roll_sat,I_roll_sat);
    roll_D= roll_KD*(er_roll-er_roll_1);
    roll_PID=roll_P+roll_I+roll_D;
    roll_PID=constrain(roll_PID,-PID_roll_sat,PID_roll_sat);
//Limitación de la señal de control

    //Actualización de los valores del regulador
    roll_PID_1=roll_PID;
    er_roll_2=er_roll_1;
    er_roll_1=er_roll;

    //PID Yaw

    //Calculo del error en Roll
    er_yaw=Yaw_ref-Yaw;

    //Calculo de la acción de control
    yaw_P= yaw_KP*(er_yaw);
    yaw_I+= yaw_KI*(er_yaw);
    yaw_I=constrain(yaw_I,-I_yaw_sat,I_yaw_sat);
    yaw_D= yaw_KD*(er_yaw-er_yaw_1);
    yaw_PID=yaw_P+yaw_I+yaw_D;
    yaw_PID=constrain(yaw_PID,-PID_yaw_sat,PID_yaw_sat);
//Limitación de la señal de control

    //Actualización de los valores del regulador
    yaw_PID_1=yaw_PID;
    er_yaw_2=er_yaw_1;
    er_yaw_1=er_yaw;



    /*if(er_roll>Roll_ref+0.5){
      PID_roll=5;
    }
```

```
    else if(er_roll<Roll_ref-0.5){
      PID_roll=-5;
    }
    else{
      PID_roll=0;
    }
    */




  //Ajuste de la velocidad de giro de los motores

    //Si nos encontramos en los primeros 15 segundos se utiliza el
control de estabilidad
    if(num_ciclos<15*10*2.5*2){
      //Calculo de la velocidad de giro de cada motor
individualmente
      pulsoE1=pulso + pitch_PID + roll_PID + yaw_PID;
      pulsoE2=pulso - pitch_PID + roll_PID - yaw_PID;
      pulsoE3=pulso + pitch_PID - roll_PID - yaw_PID;
      pulsoE4=pulso - pitch_PID - roll_PID + yaw_PID;
     }

    //Si han pasado 15 segundos se aterrizará
    else{

      //Todos los motores girarán a la misma velocidad
      pulsoE1=pulso;
      pulsoE2=pulso;
      pulsoE3=pulso;
      pulsoE4=pulso;
    }

    //Limitación del <ancho de pulso de la señal PWM de los ESC
    pulsoE1= constrain(pulsoE1,min_pul,max_pul);
    pulsoE2= constrain(pulsoE2,min_pul,max_pul);
    pulsoE3= constrain(pulsoE3,min_pul,max_pul);
    pulsoE4= constrain(pulsoE4,min_pul,max_pul);
/*
    Serial.print("ESC1: ");
    Serial.println(pulsoE1);
    Serial.print("ESC2: ");
    Serial.println(pulsoE2);
    Serial.print("ESC3: ");
    Serial.println(pulsoE3);
    Serial.print("ESC4: ");
    Serial.println(pulsoE4);*/

}

void control_altura(){

  //Lectura de la variable a controlar
    //Serial.print("Altura: ");
    //altura=altimetro();
```

```
 //Si no han pasado 15 segundos se activará el control de altura
   if(num_ciclos<15*10*2.5*2){

      //Actualizacón del error de altura
      //Serial.print("Error: ");
      ek=zref-altura;
      //Serial.println(ek);

      //Calculo de la acción de control
      //Serial.print("Uk: ");
      uk=0.8309*uk1+0.5687*ek-0.5592*ek1;//0.6248*uk1+3.129*ek-
3.0001*ek1;
      //Serial.println(uk);

      //Actualización de los valores del regulador
      ek1=ek;
      uk1=uk;
      uk=uk/9.81;

      //Suma del peso de la aeronave a la acción de control
      //Serial.print("Uk 1: ");
      uk=uk+peso_nave;

      //Conversion a Empuje a ancho de pulso de la señal PWM
      pulso=(uk/Pmax)*1000+1000;


      //Limitación del ancho de pulso de la señal PWM
      //Serial.print("Pulso Co<nstraint: ");
      pulso= constrain(pulso,min_pul,max_pul);
      //Serial.println(pulso);

    }

    //Si han pasado 15 segundos se aterrizará
    else {

      //Si el ancho de pulso es superior a 1000 ms
      if(pulso>1000){

        //Reduce la velocidad progrsivamente con cada ciclo
        pulso=pulso-5;
        pulso= constrain(pulso,min_pul,max_pul);
        //Serial.print("Pulso: ");
        //Serial.println(pulso);
      }
    }
}

void altimetro_inicializar(){

  //Inicializa los pines Trigger y Echo del sensor HC-Sr04
  pinMode(Trig,OUTPUT);
  digitalWrite(Trig,LOW);
  pinMode(Echo,INPUT);
```

```
}

void motor_calib(){

  //Se habilitán los pines de los ESC para el envio de señales PWM
  ESC_1.attach(ESC_pin1,min_pul,max_pul);
  ESC_2.attach(ESC_pin2,min_pul,max_pul);
  ESC_3.attach(ESC_pin3,min_pul,max_pul);
  ESC_4.attach(ESC_pin4,min_pul,max_pul);

  //1ro Envia la señal correspondiente al valor máximo
  //Serial.println("Estableciendo limite superior de la señal PWM");
  ESC_1.writeMicroseconds(max_pul);
  ESC_2.writeMicroseconds(max_pul);
  ESC_3.writeMicroseconds(max_pul);
  ESC_4.writeMicroseconds(max_pul);
  delay(3000);

  //2nd Envia la señal correspondiente al valor mínimo
  //Serial.println("Estableciendo limite inferior de la señal PWM");
  ESC_1.writeMicroseconds(min_pul);
  ESC_2.writeMicroseconds(min_pul);
  ESC_3.writeMicroseconds(min_pul);
  ESC_4.writeMicroseconds(min_pul);
  delay(7000);
}


float altimetro(){
  //Mide la altura mediante el sensor HC-SR04

  //Inicialización de varibles:
  long t; //timepo de vuelo
  float d; //distancia

  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);          //Enviamos un pulso de 10us

  digitalWrite(Trig, LOW);
  t = pulseIn(Echo, HIGH,23500); //Obtenemos el ancho del pulso

  d = t*0.00034/2;              //Distancia en metros
  //Serial.print("Altura: ");
  //Serial.println(d);
  return d;
}

void IMU_Ini(void){
  //Serial.println("Orientation Sensor Test"); Serial.println("");

  /* Initialise the sensor */
  if(!bno.begin())
  {
    /* There was a problem detecting the BNO055 ... check your
connections */
    //Serial.print("Ooops, no BNO055 detected ... Check your wiring
or I2C ADDR!");
```

```
    while(1);
  }

  delay(1000);

  bno.setExtCrystalUse(true);

  for (int cal_int = 0; cal_int < 10 ; cal_int ++) {
    IMU_read_abspos();
    /*
    Serial.print("X: ");
    Serial.print(Yaw, 4);
    Serial.print("\tY: ");
    Serial.print(Pitch, 4);
    Serial.print("\tZ: ");
    Serial.print(Roll, 4);
    Serial.println("");
    */
    delay(100);
  }

  Roll_cal=Roll;
  Pitch_cal=Pitch;
  Yaw_cal=Yaw;
}

void IMU_read_abspos(){
  /* Get a new sensor event */
  sensors_event_t event;
  bno.getEvent(&event);

  /* Display the floating point data */
 // Serial.print("X: ");
  Yaw=event.orientation.x;
  if (Yaw>=180) Yaw= Yaw-360;
  //Serial.println(Yaw, 4);
  //Serial.print("\tY: ");
  Pitch=event.orientation.y;
  //Serial.print(Pitch, 4);
  //Serial.print("\tZ: ");
  Roll=event.orientation.z;
  //Serial.print(Roll, 4);
  //Serial.println("");
}
```

2.4 Funciones

2.4.1 Altimetro_inicializar()

Esta función configura los pines de entrada y salida del microcontrolador correspondientes a los pines Echo y Trigger del sensor de ultrasonidos HC-Sr04.

2.4.1.1 Código

```
void altimetro_inicializar(){

  //Inicializa los pines Trigger y Echo del sensor HC-Sr04
  pinMode(Trig,OUTPUT);
  digitalWrite(Trig,LOW);
  pinMode(Echo,INPUT);
}
```

2.4.2 Altimetro()

Esta función, calcula la distancia respecto al suelo. Para ello, envía una señal de Trigger al sensor de ultrasonidos, posteriormente se espera a recibir la señal Echo. Finalmente, con el tiempo de vuelo de la señal del Hc-Sr04 y la velocidad del sonido se obtiene la distancia respecto al suelo.

2.4.2.1Código

```
float altimetro(){
  //Mide la altura mediante el sensor HC-SR04

  //Inicialización de varibles:
  long t; //timepo de vuelo
  float d; //distancia

  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);          //Enviamos un pulso de 10us

  digitalWrite(Trig, LOW);
  t = pulseIn(Echo, HIGH,23500); //Obtenemos el ancho del pulso

  d = t*0.00034/2;              //Distancia en metros
  //Serial.print("Altura: ");
  //Serial.println(d);
  return d;
}
```

2.4.3 Motor_calib()

Esta función calibra los motores estableciendo los anchos de pulso de entrada de los ESCs correspondientes al 0% y al 100% de velocidad de giro. Este procedimiento ya se ha explicado en el apartado donde se profundiza sobre los variadores eléctricos. Sin embargo y resumiendo este procedimiento, la manera de configurarlos es la siguiente primero se envía a los ESCs la señal correspondiente a la velocidad máxima y tras unos segundos la señal de la velocidad mínima.

2.4.3.1 Diagrama de flujo

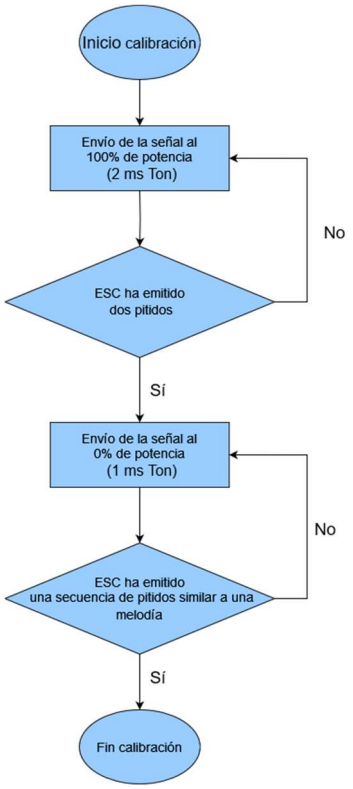


Figura 54: Diagrama de flujo de la calibración de los motores.

2.4.3.2 Código

```
void motor_calib(){

  //Se habilitán los pines de los ESC para el envio de señales PWM
  ESC_1.attach(ESC_pin1,min_pul,max_pul);
  ESC_2.attach(ESC_pin2,min_pul,max_pul);
  ESC_3.attach(ESC_pin3,min_pul,max_pul);
  ESC_4.attach(ESC_pin4,min_pul,max_pul);

  //1ro Envia la señal correspondiente al valor máximo
  //Serial.println("Estableciendo limite superior de la señal PWM");
```

```
  ESC_1.writeMicroseconds(max_pul);
  ESC_2.writeMicroseconds(max_pul);
  ESC_3.writeMicroseconds(max_pul);
  ESC_4.writeMicroseconds(max_pul);
  delay(3000);

  //2nd Envia la señal correspondiente al valor mínimo
  //Serial.println("Estableciendo limite inferior de la señal PWM");
  ESC_1.writeMicroseconds(min_pul);
  ESC_2.writeMicroseconds(min_pul);
  ESC_3.writeMicroseconds(min_pul);
  ESC_4.writeMicroseconds(min_pul);
  delay(7000);
}
```

2.4.4 IMU_Ini()

Esta función inicia la comunicación por I2C con la IMU BNO055. Posteriormente, toman diez medidas de los ángulos abosultos, suficientes como para obtener los valores de offset de los ángulos roll, pitch, yaw. Esta función parte de un ejemplo que viene incluido en las librerías del propio sensor.

2.4.4.1 Diagrama de flujo

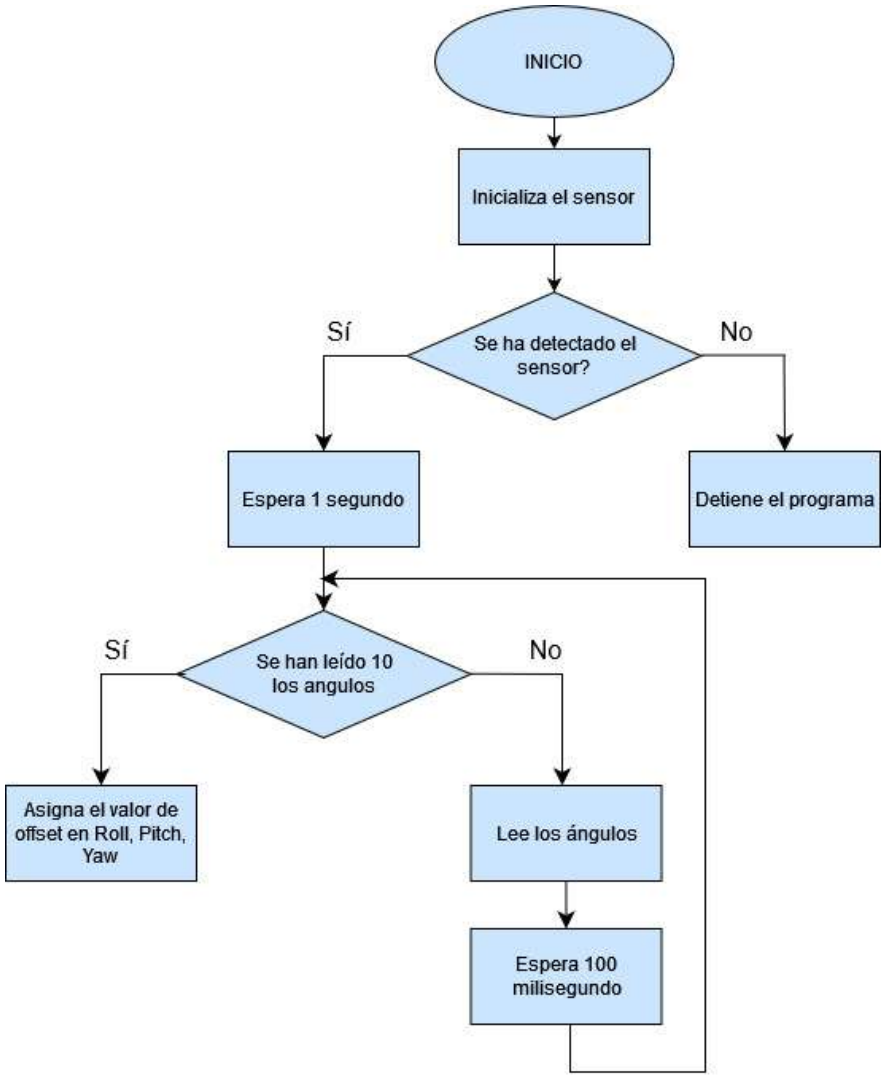


Figura 55: Diagram de flujo IMU_ini().

2.4.4.2 Código

```
void IMU_Ini(void){
  Serial.println("Orientation Sensor Test"); Serial.println("");

  /* Initialise the sensor */
  if(!bno.begin())
  {
    /* There was a problem detecting the BNO055 ... check your
connections */
    Serial.print("Ooops, no BNO055 detected ... Check your wiring or
I2C ADDR!");
    while(1);
  }
```

```
  delay(1000);

  bno.setExtCrystalUse(true);

  for (int cal_int = 0; cal_int < 10 ; cal_int ++) {
    IMU_read_abspos();
    /*
    Serial.print("X: ");
    Serial.print(Yaw, 4);
    Serial.print("\tY: ");
    Serial.print(Pitch, 4);
    Serial.print("\tZ: ");
    Serial.print(Roll, 4);
    Serial.println("");
    */
    delay(100);
  }

  Roll_cal=Roll;
  Pitch_cal=Pitch;
  Yaw_cal=Yaw;
}
```

2.4.5 IMU_read_abspos()

Esta función, se encarga de leer los valores de orientación angular absoluta de la IMU mediante la comunicación I2C. Esta función parte de un ejemplo que viene incluido en las librerías del propio sensor.

2.4.5.1 Diagrama de flujo

Figura 56: Diagrama de flujo de la función IMU_read_abspos().

2.4.5.2 Código

```
void IMU_read_abspos(){
  /* Get a new sensor event */
  sensors_event_t event;
  bno.getEvent(&event);

  /* Display the floating point data */
 // Serial.print("X: ");
  Yaw=event.orientation.x;
  if (Yaw>=180) Yaw= Yaw-360;
  //Serial.println(Yaw, 4);
  //Serial.print("\tY: ");
  Pitch=event.orientation.y;
  //Serial.print(Pitch, 4);
  //Serial.print("\tZ: ");
  Roll=event.orientation.z;
  //Serial.print(Roll, 4);
  //Serial.println("");
}
```

2.4.6 Control_altura()

Esta función obtiene el ancho de pulso necesario para generar el empuje que permita seguir la consigna de altura los primeros 15 segundos de vuelo, luego reduce la velocidad progresivamente para aterrizar.

2.4.6.1 Diagrama de flujo

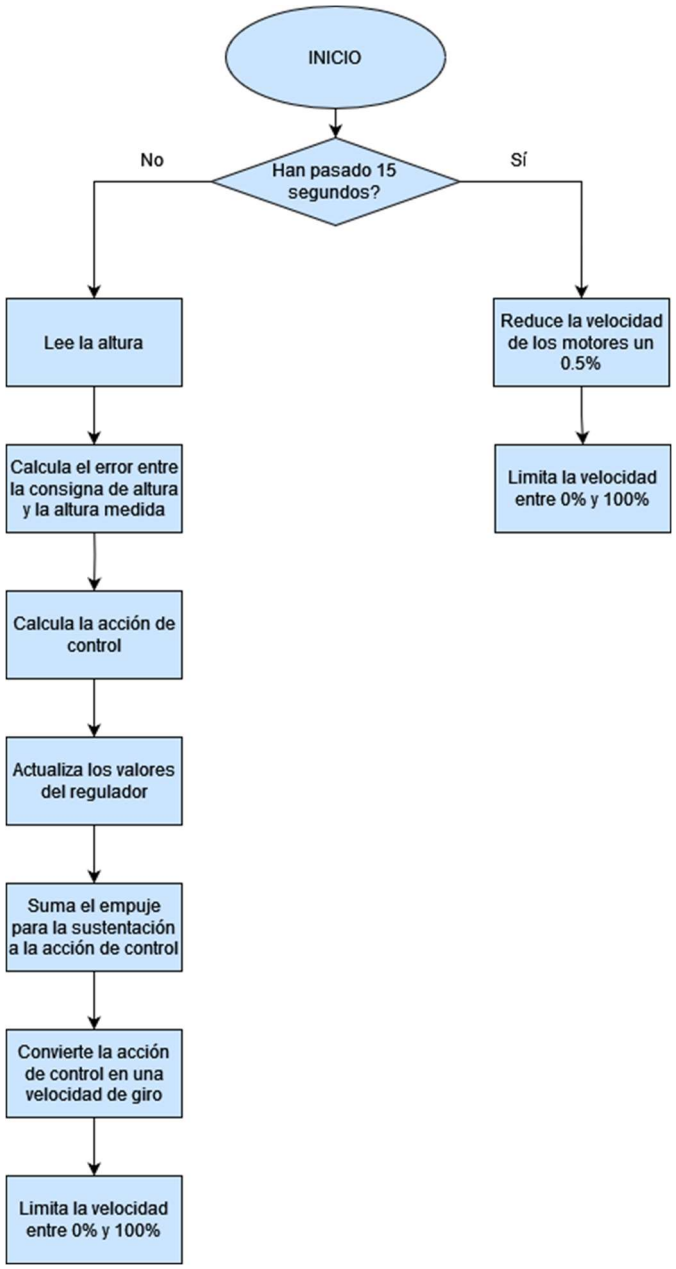


Figura 57: Diagrama de flujo Control_altura.

2.4.6.2 Código

```
void control_altura(){

  //Lectura de la variable a controlar
    //Serial.print("Altura: ");
    altura=altimetro();
    //Serial.print(altura);
```

```
    //Serial.print("\t");

 //Si no han pasado 15 segundos se activará el control de altura
   if(num_ciclos<15*10*2.5*2){

      //Actualizacón del error de altura
      //Serial.print("Error: ");
      ek=zref-altura;
      //Serial.println(ek);

      //Calculo de la acción de control
      //Serial.print("Uk: ");
      uk=0.6248*uk1+3.129*ek-3.0001*ek1;
      //Serial.println(uk);

      //Actualización de los valores del regulador
      ek1=ek;
      uk1=uk;

      //Suma del peso de la aeronave a la acción de control
      //Serial.print("Uk 1: ");
      uk=uk+peso_nave;
      //Serial.println(uk);

      //Conversion a Empuje a ancho de pulso de la señal PWM
      pulso=(uk/Pmax)*1000+1000;
      //Serial.println(pulso);

      //Limitación del ancho de pulso de la señal PWM
      //Serial.print("Pulso Constraint: ");
      pulso= constrain(pulso,min_pul,max_pul);
      //Serial.println(pulso);
    }

    //Si han pasado 15 segundos se aterrizará
    else {

      //Si el ancho de pulso es superior a 1000 ms
      if(pulso>1000){

        //Reduce la velocidad progrsivamente con cada ciclo
        pulso=pulso-5;
        pulso= constrain(pulso,min_pul,max_pul);
        //Serial.print("Pulso: ");
        //Serial.println(pulso);
      }
    }
}
```

2.4.7 Control_estabilidad()

Esta función calcula las velocidades de giro en cada motor que permita compensar las desviaciones en los ángulos Roll y Pitch. Estas velocidades se consiguen a partir de la velocidad resultante del control de altura y la acción de control para compensar el cabeceo y alabeo.

2.4.7.1 Diagrama de flujo

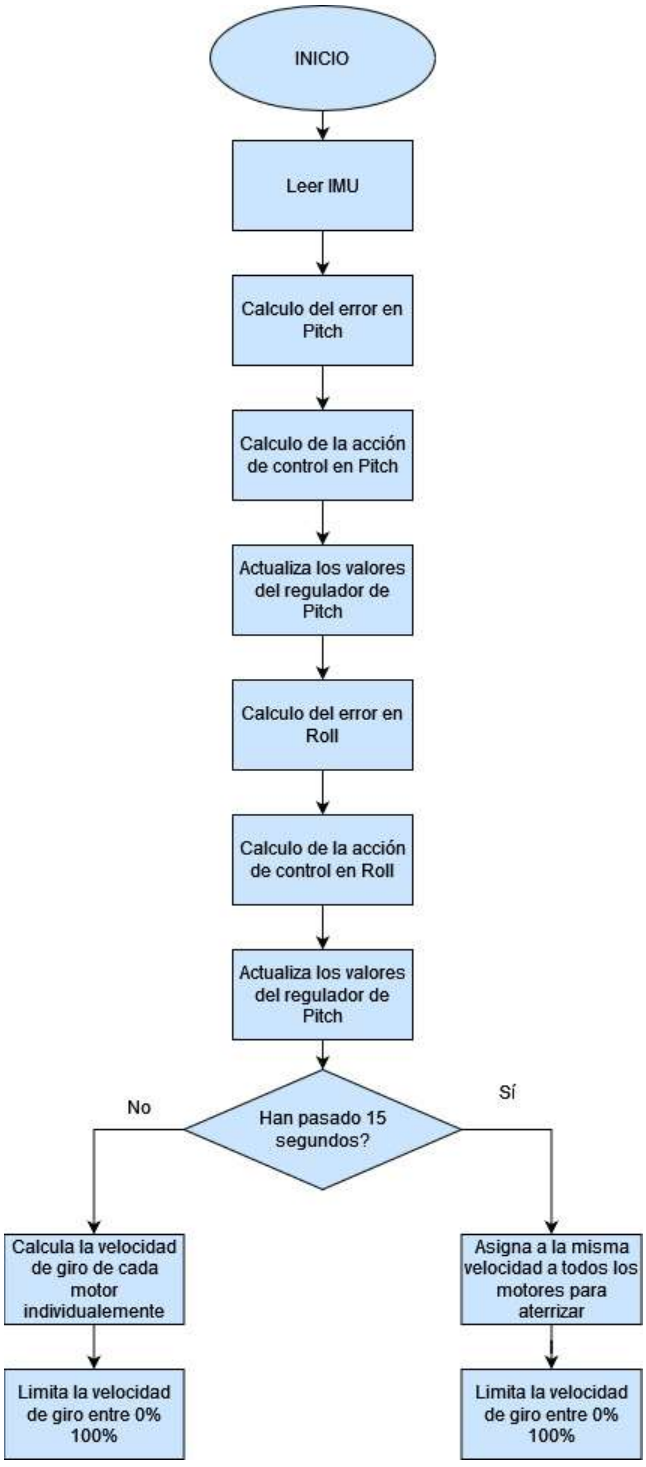


Figura 58: Diagrama de flujo Control_estabilidad.

2.4.7.2 Código

```
void control_estabilidad(){
  //Lectura de la IMU

  IMU_read_abspos();


  //Calculo de los ángulos

  Yaw=Yaw-Yaw_cal;
  Roll=Roll-Roll_cal;
  Pitch=Pitch-Pitch_cal;


  //PID Pitch

   //Caluclo del error en Pitch
    er_pitch=Pitch-Pitch_ref;

    //Calculo de la acción de control
   //pitch_PID=pitch_PID_1 +
(pitch_KP+pitch_KI*T/(2*1000)+pitch_KD*1000/T)*er_pitch+(-
pitch_KP+pitch_KI*T/(2*1000)-
2*pitch_KD*1000/T)*er_pitch_1+(pitch_KD*1000/T)*er_pitch_2;
    //pitch_PID=pitch_PID_1 +
q0p*er_pitch+q1p*er_pitch_1+q2p*er_pitch_2;
    //pitch_PID=pitch_PID_1 +
q0p*er_pitch+q1p*er_pitch_1+q2p*er_pitch_2;
    //pitch_PID= K1p*pitch_PID_1 + K2p*er_pitch -
K3p*er_pitch_1    pitch_P= pitch_KP*(er_pitch);
    pitch_P= pitch_KP*(er_pitch);
    pitch_I+=  pitch_KI*(er_pitch);
    pitch_I=constrain(pitch_I,-I_pitch_sat,I_pitch_sat);
    pitch_D= pitch_KD*(er_pitch-er_pitch_1);
    pitch_PID=pitch_P+pitch_I+pitch_D;
    pitch_PID=constrain(pitch_PID,-PID_pitch_sat,PID_pitch_sat);
//Limitación de la señal de control

    //Actualización de los valores del regulador
    pitch_PID_1=pitch_PID;
    er_pitch_2=er_pitch_1;
    er_pitch_1=er_pitch;


/*
    if(er_pirch>Pitch_ref+0.5){
      PID_pitch=5;
    }
    else if(er_pitch<Pitch_ref-0.5){
      PID_pitch=-5;
    }
    else{
      PID_pitch=0;
    }
    */
  //PID Roll
```

```
    //Calculo del error en Roll
    er_roll=Roll-Roll_ref;

    //Calculo de la acción de control
    //roll_PID=roll_PID_1 +
(roll_KP+roll_KI*T/(2*1000)+roll_KD*1000/T)*er_roll+(-
roll_KP+roll_KI*T/(2*1000)-
2*roll_KD*1000/T)*er_roll_1+(roll_KD*1000/T)*er_roll_2;
    //roll_PID=roll_PID_1 + q0r*er_roll + q1r*er_roll_1 +
q2r*er_roll_2;
    //roll_PID= K1r*roll_PID_1 + K2r*er_roll - K3r*er_roll_1;
    roll_P= roll_KP*(er_roll);
    roll_I+= roll_KI*(er_roll);
    roll_I=constrain(roll_I,-I_roll_sat,I_roll_sat);
    roll_D= roll_KD*(er_roll-er_roll_1);
    roll_PID=roll_P+roll_I+roll_D;
    roll_PID=constrain(roll_PID,-PID_roll_sat,PID_roll_sat);
//Limitación de la señal de control

    //Actualización de los valores del regulador
    roll_PID_1=roll_PID;
    er_roll_2=er_roll_1;
    er_roll_1=er_roll;

    //PID Yaw

    //Calculo del error en Roll
    er_yaw=Yaw_ref-Yaw;

    //Calculo de la acción de control
    yaw_P= yaw_KP*(er_yaw);
    yaw_I+= yaw_KI*(er_yaw);
    yaw_I=constrain(yaw_I,-I_yaw_sat,I_yaw_sat);
    yaw_D= yaw_KD*(er_yaw-er_yaw_1);
    yaw_PID=yaw_P+yaw_I+yaw_D;
    yaw_PID=constrain(yaw_PID,-PID_yaw_sat,PID_yaw_sat);
//Limitación de la señal de control

    //Actualización de los valores del regulador
    yaw_PID_1=yaw_PID;
    er_yaw_2=er_yaw_1;
    er_yaw_1=er_yaw;



    /*if(er_roll>Roll_ref+0.5){
      PID_roll=5;
    }
    else if(er_roll<Roll_ref-0.5){
      PID_roll=-5;
    }
    else{
      PID_roll=0;
    }
    */
```

```
  //Ajuste de la velocidad de giro de los motores

    //Si nos encontramos en los primeros 15 segundos se utiliza el
control de estabilidad
    if(num_ciclos<15*10*2.5*2){
      //Calculo de la velocidad de giro de cada motor
individualmente
      pulsoE1=pulso + pitch_PID + roll_PID + yaw_PID;
      pulsoE2=pulso - pitch_PID + roll_PID - yaw_PID;
      pulsoE3=pulso + pitch_PID - roll_PID - yaw_PID;
      pulsoE4=pulso - pitch_PID - roll_PID + yaw_PID;
     }

    //Si han pasado 15 segundos se aterrizará
    else{

      //Todos los motores girarán a la misma velocidad
      pulsoE1=pulso;
      pulsoE2=pulso;
      pulsoE3=pulso;
      pulsoE4=pulso;
    }

    //Limitación del <ancho de pulso de la señal PWM de los ESC
    pulsoE1= constrain(pulsoE1,min_pul,max_pul);
    pulsoE2= constrain(pulsoE2,min_pul,max_pul);
    pulsoE3= constrain(pulsoE3,min_pul,max_pul);
    pulsoE4= constrain(pulsoE4,min_pul,max_pul);
/*
    Serial.print("ESC1: ");
    Serial.println(pulsoE1);
    Serial.print("ESC2: ");
    Serial.println(pulsoE2);
    Serial.print("ESC3: ");
    Serial.println(pulsoE3);
    Serial.print("ESC4: ");
    Serial.println(pulsoE4);*/

}
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje e implementación del control de altura de

un cuadricóptero

Anexo 4: Relación del trabajo con los objetivos de desarrollo sostenible de la agenda 2030.

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Fontalba Roca, Marc

Tutor/a: González Sorribes, Antonio

CURSO ACADÉMICO: 2022 / 2023

Índice Anexo 4:

1. Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS)

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**	-	-	-	
ODS 2. **Hambre cero.**	-	-	-	
ODS 3. **Salud y bienestar.**	-	-	-	
ODS 4. **Educación de calidad.**	-	-	-	
ODS 5. **Igualdad de género.**	-	-	-	
ODS 6. **Agua limpia y saneamiento.**	-	-	-	
ODS 7. **Energía asequible y no contaminante.**	-	-	-	
ODS 8. **Trabajo decente y crecimiento económico.**	-	-	-	
ODS 9. **Industria, innovación e infraestructuras.**	-		-	-
ODS 10. **Reducción de las desigualdades.**	-	-	-	
ODS 11. **Ciudades y comunidades sostenibles.**	-	-	-	
ODS 12. **Producción y consumo responsables.**	-		-	-
ODS 13. **Acción por el clima.**	-	-	-	
ODS 14. **Vida submarina.**	-	-	-	
ODS 15. **Vida de ecosistemas terrestres.**	-	-	-	
ODS 16. **Paz, justicia e instituciones sólidas.**	-	-	-	
ODS 17. **Alianzas para lograr objetivos.**	-	-	-	

2. ODS 9. Industria, innovación e infraestructuras

El proyecto cumple con el siguiente objetivo puesto que promueve y facilita el acceso a los conocimientos sobre la fabricación de cuadricópteros. De esta forma, es más sencillo acceder a estas tecnologías.

3. ODS 12. Producción y consumo responsables

El dispositivo cumple con el siguiente objetivo por varias razones. En primer lugar, en la construcción del dispositivo se emplearon materiales que se reutilizaron de anteriores proyectos como el Arduino o el sensor de ultrasonidos, por otro lado, también se reutilizó la espuma del encapsulado de la IMU. En segundo lugar, al emplear un fuselaje de fibra de carbono, el prototipo es resistente a impactos y roturas reduciendo así el número de veces que se necesitará alguna pieza nueva para sustituir las dañadas. En tercer lugar, se utiliza una batería LiPo que en comparación a otras alternativas como las NiMH que son menos contaminantes y permiten más ciclos de recarga.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje e implementación del control de altura de

un cuadricóptero

Anexo 5: Fichas técnicas.

Grado en Ingeniería Electrónica Industrial y Automática

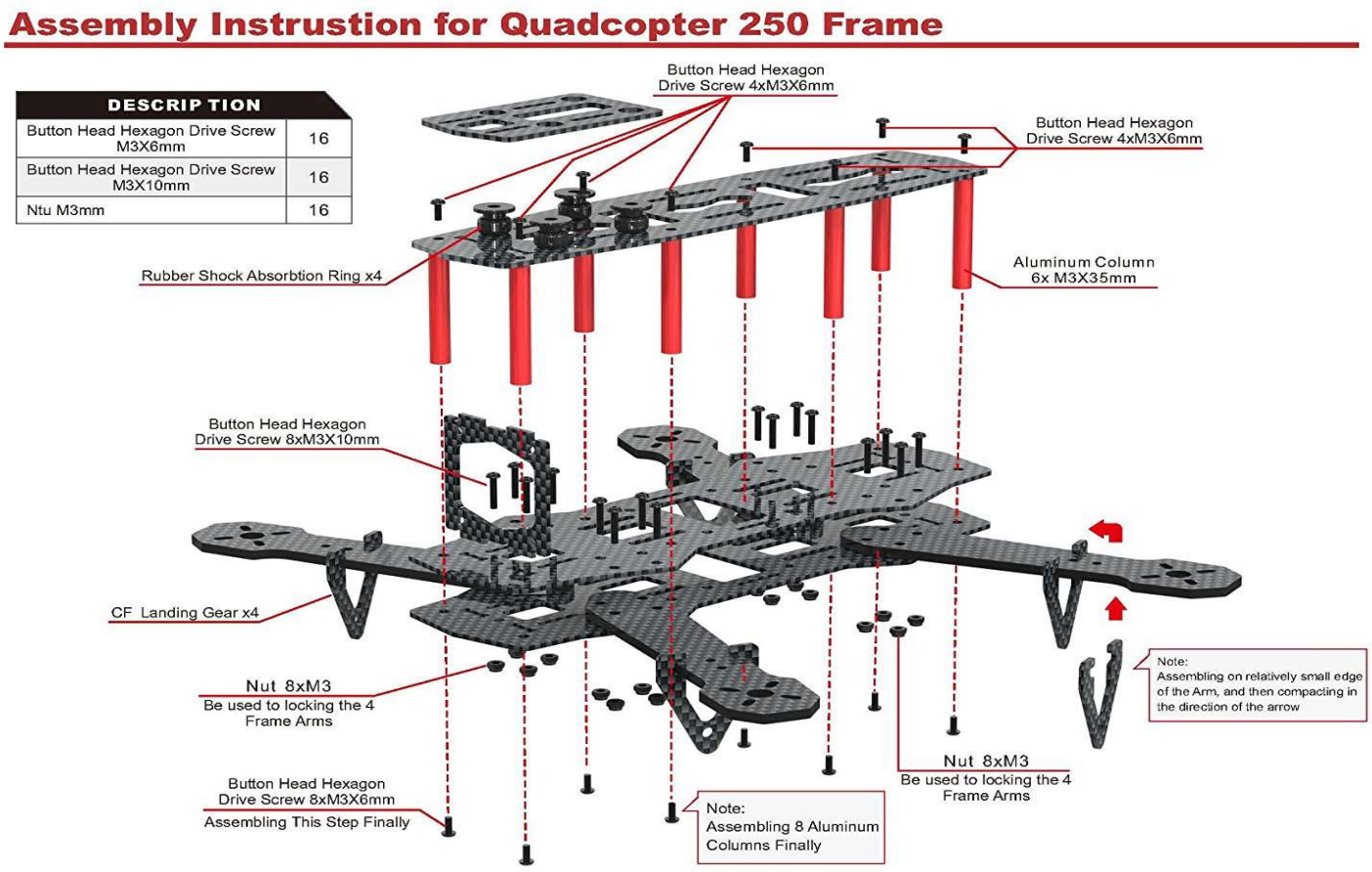
AUTOR/A: Fontalba Roca, Marc

Tutor/a: González Sorribes, Antonio

CURSO ACADÉMICO: 2022 / 2023

Índice Anexo 5:

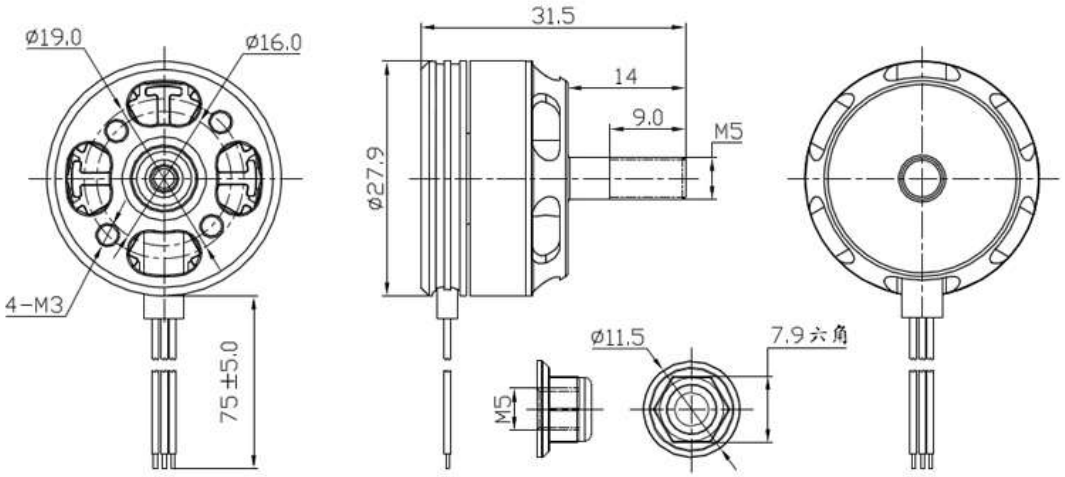
1. Frame


Assembly Instrustion for Quadcopter 250 Frame
DESCRIP TION
Button Head Hexagon Drive Screw M3X6mm 16
Button Head Hexagon Drive Screw M3X10mm 16
Ntu M3mm 16
Button Head Hexagon Drive Screw 4xM3X6mm
Button Head Hexagon Drive Screw 4xM3X6mm
Rubber Shock Absorbtion Ring x4
Aluminum Column 6x M3X35mm
Button Head Hexagon Drive Screw 8xM3X10mm
CF Landing Gear x4
Nut 8xM3 Be used to locking the 4 Frame Arms
Button Head Hexagon Drive Screw 8xM3X6mm Assembling This Step Finally
Note: Assembling 8 Aluminum Columns Finally
Nut 8xM3 Be used to locking the 4 Frame Arms
Note: Assembling on relatively small edge of the Arm, and then compacting in the direction of the arrow

2. Motor DX2205 2300KV

GoolRC DX2205 2300KV
CW/CCW Brushless Motor Kit

MOTOR OUTLINE DRAWING :



Flange Lock Nuts

MOTOR PERFORMANCE DATA :

MODEL	KV (rpm/V)	Voltage (V)	Prop	Load Current (A)	Pull (g)	Power (W)	Efficiency (g/W)	Lipo Cell	Weight (g) Approx
DX2205	2300	11.1	5045	19.2	660	213	3.1	2-4S	28
		14.8		27.6	950	408	2.3		
	2600	11.1	4045	18.5	530	205	2.6		
		14.8		23.2	710	343	2.1		

3. ESC BLHELI EMAX 12A

EMAX User Instruction for BLHeli Series ESC

Thank you for purchasing EMAX ESC, please read this manual carefully before you use the ESC and strictly follow the instructions. EMAX accepts no liability for damage(s) or injuries incurred directly or indirectly from the use of this product, or modification of this product. Due to unforeseen changes or product upgrades in design, appearance, performance, the information contained in this manual is subject to change without notice.

A. Features

A1: Use authentic electronic components to ensure high quality and enhance the current endurance ability of the ESC.

A2: Based on BLHeli firmware, optimized for high performance with great linearity and much quicker throttle response.

A3: Special designed for multirotors, and compatible with fixed-wing aircrafts and helicopters.

A4: Multiple protection features including Low-voltage cut-off protection / over-heat protection / throttle signal loss protection.

A5: Throttle range can be configured and is fully compatible with all receivers, providing smooth, linear and precise throttle response.

A6: All parameters can be programmed via using a transmitter, including default settings.

B.Product specification

Item	Continuous Current	Burst current (10S)	Li-xx Battery (cell)	Dimension L*W*H(mm)	Weight (g) wires Included	BEC Mode	BEC Output	Programmable
EMAX BLHeli-6A	6A	8A	1-2	22×13×5.5	6	Linear	0.8A/5V	YES
EMAX BLHeli -12A	12A	15A	2-4	42×20×8	11	Linear	1A/5V	YES
EMAX BLHeli -20A	20A	25A	2-4	52×26×7	28	Linear	2A/5V	YES
EMAX BLHeli -25A	25A	30A	2-4	52×26×7	28	Linear	2A/5V	YES
EMAX BLHeli -30A	30A	40A	2-4	52×26×7	28	Linear	2A/5V	YES
EMAX BLHeli -30A-OPTO	30A	40A	2-6	67×26×10	25	---	---	YES
EMAX BLHeli -40A-UBEC	40A	50A	2-6	73×28×12	41	Switch	3A/5V	YES
EMAX BLHeli -50A-UBEC	50A	60A	2-6	73×28×12	41	Switch	3A/5V	YES
EMAX BLHeli -60A-UBEC	60A	80A	2-6	73×36×12	63	Switch	5A/5V	YES
EMAX BLHeli -80A-UBEC	80A	100A	2-6	86×38×12	81	Switch	5A/5V	YES

C. Instructions

C1.Normal startup procedures

Move throttle stick to the bottom position and then switch on transmitter→ Connect battery pack to ESC→ The long "beep" sound should be emitted , means the bottom point of throttle range has been detected→ Several "beep" tones should be emitted to present the amount of battery cells→ When self-test is finished, a "♪ 1 2 3" tune should be emitted→ Move throttle stick upwards to go flying.

C2.Throttle range setting procedures (when users change a transmitter, throttle range setting is recommended.)

Switch on the transmitter, move throttle stick to the top position→ Connect battery pack to ESC→ Two "beep" sounds should be emitted, means the top point of throttle range has been confirmed and saved→ Move throttle stick to the bottom position (within 2s), a long "beep" sound should be emitted , means the bottom point of throttle range has been detected→ Several "beep" tones should be emitted to present the amount of battery cells→ When self-test is finished, a "♪ 1 2 3" tune should be emitted, Move throttle stick upwards to go flying.

If the throttle stick is neither at the bottom position nor the top position after powered on, it will constantly make "beep" sounds.

D. Programmable parameters

D1. Brake Type: There are two options: **OFF**, ON. The default is OFF.

D2. Timing Mode: There are five options: Low: 0°, Mid-low: 8°, **Middle:15°**, Mid-high:23° and High:30°. The default is Middle: 15°. Low advance timing is recommended for high inductance and low KV motors. High advance timing is recommended for low inductance and high KV outrunner motors. For some high KV motors, if it shakes while rotating in high speed, the High timing mode is recommended.

D3. Start Force: There are 13 options: 0.031, 0.047, 0.063, 0.094, 0.125, 0.188, 0.25, 0.38, 0.50, **0.75**, 1.00, 1.25, 1.50. The default is 0.75. Select the corresponding start force according to the load of motor.

D4. Curve Mode: There are 4 options: **OFF**, Low, Middle and High. The default is OFF.

D5. Control Frequency : 2 options: **8KHz** and 22KHz. The default is 8KHz. This option is the drive frequency of the motors.

D6. Low-voltage Protection: 4 options: OFF, 2.8V/cell, **3.0V/cell**, 3.2V/cell. The default is 3.0V/cell. the system will automatically identify the battery cell. E.g. suppose there're 3 cells, if the voltage is lower than 9V, the system will work according to the current cutoff option.

D7. Cutoff Mode: There are two options: **Soft-Cut** and Cut-Off. The default is Soft-Cut. Soft-Cut option: Gradually reduce throttle power to 31% of the current power when the voltage is lower than the programmed low-voltage protection threshold. Cut-Off Option: immediate motor shutdown occurs in low-voltage.

When low-voltage protection, Push the throttle stick to the bottom position and then to the top position, the motor will be restarted. But since it is low-voltage condition, the output power is low or stopped at once.

D8. Rotation Direction: There are 3 options: **Normal**, Reverse, Bidirectional. The default is Normal.

E. Protection setting

E1. Low-voltage Protection: Whether to shut down the motor immediately or to lower the power when the input voltage drops below the programmed low-voltage protection threshold depends on the values set as Cutoff Mode. (Please refer to D7 for Cutoff Mode)

E2. Loss of Signal Protection: Power will be gradually lower to 0 when signal lost, and motor stops. Motor will resume to the current power when the signal is detected again.

E3. Over-heat Protection: When the temperature of the ESC MOSFETS exceeds 100 Celsius degree, power will be lowered gradually and will resume when the temperature decreases.

F. Programming via Transmitter

Step 1: Enter program mode

Switch the transmitter on→Pull the throttle stick to the top position→Switch the ESC on, wait 2 seconds, you will hear two "beep" sounds, which denotes that Max. throttle has been confirmed→Hold the throttle stick at the top position, and then wait 2 seconds until you hear tune " ♪ 1 2 3 ♪ 1 2 3", that means you have entered the transmitter programming mode.

Step 2: Select program parameters

Hold the throttle stick on top position, there're 7 parameters can be set by using your transmitter. You would hear 7 different indicating sounds which correspond to 7 different parameters. Pull the throttle stick to the bottom position (full Off throttle) within 2 seconds after you hear the correspondent sound will brings you to the correspondent parameter setting status. The indicating sounds will repeat in turn as follow.

1. "beep-" (a short sound) which indicates the Brake Type
2. "beep-beep-" (two short sounds) which indicates the Timing Mode
3. "beep-beep-beep-" (three short sounds) which indicates the Start Force
4. "beep-beep-beep-beep-" (four short sounds) which indicates the Curve Mode
5. "beep-----" (a long sound) which indicates the Control Frequency
6. "beep-----beep-" (a long sound and a short) which indicates the Low-voltage Protection
7. "beep-----beep-beep-" (a long sound and two short) which indicates the Cutoff Mode
8. "beep-----beep-beep-beep-" (a long sound and three short) which indicates the Rotation Direction

Step 3: Select program values

After entering parameter setting status, hold the throttle stick on the bottom position, you will be led to the repeat selection of that parameter setting status. Each sound likes 4 short sounds and one long sound (1 long sound=5short sounds), and by that analogy. After some sound, pull the throttle stick to the top position in 2 seconds, after you hear a tune "♪3 2 1 ♪3 2 1", which means the correspondent value has been chosen and saved. Hold the stick on the top position, return to the second step and continue programming.

Step 4: Exit program

Pull the throttle stick to the bottom position within 2 seconds and hold on after saving parameters, until you hear a tune "beep---- beep- beep- beep- ♪ 1 2 3". Set the Min. Throttle at this moment and exit program and operate as normal.(beep----means Loading parameter. beep- beep- beep-means numbers of cells and ♪1 2 3 means ready.)

Restore Factory settings

To restore Factory settings, pull the throttle stick to the bottom position (full Off throttle) within 1s after entering program mode (Step 1)→Pull the throttle stick to the top position within 2s, then you will hear a tune, which means that the factory settings have been restored. Pull the throttle stick to the bottom position within 2s, the ESC is ready with factory settings.

If the Bidiretional mode is chosen, the Minimum Throttle actually means Middle Throttle position. Factory settings can be restored under Bidirectional mode.

Note: ESC program card is no longer produced, future ESCs will not support program card any more.

4. Batería RoarigTop 2200mAh 35 C

5. Arduino Nano IoT 33

Arduino® Nano 33 IoT

Product Reference Manual
SKU: ABX00027

Description

The Arduino Nano 33 IoT and Arduino Nano 33 IoT with headers are a miniature sized module containing a Cortex M0+ SAMD21 processor, a Wi-Fi®+Bluetooth® module based on ESP32, a crypto chip which can securely store certificates and pre-shared keys and a 6 axis IMU. The module can either be mounted as a DIP component (when mounting pin headers), or as a SMT component, directly soldering it via the castellated pads.

Target areas:

Maker, enhancements, basic IoT application scenarios

Features

- **SAMD21G18A**
  - **Processor**
    - 256KB Flash
    - 32KB Flash
    - Power On Reset (POR) and Brown Out Detection (BOD)
  - **Peripherals**
    - 12 channel DMA
    - 12 channel event system
    - 5x 16 bit Timer/Counter
    - 3x 24 bit timer/counter with extended functions
    - 32 bit RTC
    - Watchdog Time
    - CRC-32 generator
    - Full speed Host/Device USB with 8 end points
    - 6x SERCOM (USART, I^2C, SPI, LIN)
    - Two channel I^2S
    - 12 bit 350ksps ADC (up to 16 bit with oversampling)
    - 10 bit 350ksps DAC
    - External Interrupt Controller (up to 16 lines)

- **Nina W102**
  - **Module**
    - Dual Core Tensilica LX6 CPU at up to 240MHz
    - 448 KB ROM, 520KB SRAM, 2MB Flash
  - **WiFi**
    - IEEE 802.11b up to 11Mbit
    - IEEE 802.11g up to 54MBit
    - IEEE 802.11n up to 72MBit
    - 2.4 GHz, 13 channels
    - -96 dBm sensitivity
  - **Bluetooth® BR/EDR**
    - Max 7 peripherals
    - 2.4 GHz, 79 channels
    - Up to 3 Mbit/s
    - 8 dBm output power at 2/3 Mbit/s
    - 11 dBm EIRP at 2/3 Mbit/s
    - -88 dBm sensitivity
  - **Bluetooth® Low Energy**
    - Bluetooth® 4.2 dual mode
    - 2.4GHz 40 channels
    - 6 dBm output power
    - 9 dBm EIRP
    - -88 dBm sensitivity
    - Up to 1 Mbit/
  - **MPM3610** (DC-DC)
    - Regulates input voltage from up to 21V with a minimum of 65% efficiency @minimum load
    - More than 85% efficiency @12V
  - **ATECC608A** (Crypto Chip)
    - Cryptographic co-processor with secure hardware based key storage
    - Protected storage for up to 16 keys, certificates or data
    - ECDH: FIPS SP800-56A Elliptic Curve Diffie-Hellman
    - NIST standard P256 elliptic curve support
    - SHA-256 & HMAC hash including off-chip context save/restore
    - AES-128 encrypt/decrypt, galois field multiply for GCM
  - **LSM6DSL** (6 axis IMU)
    - Always-on 3D accelerometer and 3D gyroscope
    - Smart FIFO up to 4 KByte based
    - ±2/±4/±8/±16 g full scale
    - ±125/±250/±500/±1000/±2000 dps full scale

Contents

1 The Board

As all Nano form factor boards, Nano 33 IoT and Nano 33 IoT with headers does not have a battery charger but can be powered through USB or headers.

NOTE: Arduino Nano 33 IoT and Nano 33 IoT with headers only supports 3.3V I/Os and is **NOT** 5V tolerant so please make sure you are not directly connecting 5V signals to this board or it will be damaged. Also, as opposed to Arduino Nano boards that support 5V operation, the 5V pin does NOT supply voltage but is rather connected, through a jumper, to the USB power input.

1.1 Application Examples

Weather station: Using the Arduino Nano 33 IoT or Nano 33 IoT with headers together with a sensor and a OLED display, we can create a small weather station communicating temperature, humidity etc. directly to your phone.

Air quality monitor: Bad air quality may have serious effects on your health. By assembling the board, with a sensor and monitor you can make sure that the air quality is kept in indoor-environments. By connecting the hardware assembly to an IoT application/API, you will receive real time values.

Air drum: A quick and fun project is to create a small air drum. Connect your board and upload your sketch from the Create Web Editor and start creating beats with your audio workstation of your choice.

2 Ratings

2.1 Recommended Operating Conditions

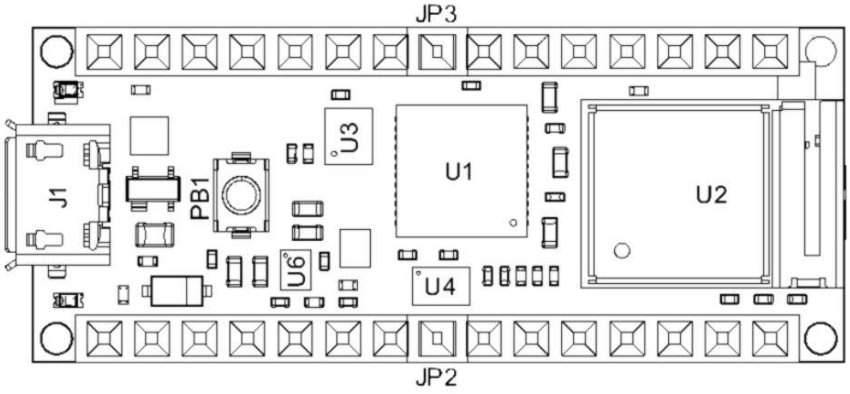
Symbol	Description	Min	Max
	Conservative thermal limits for the whole board:	-40 °C (40 °F)	85°C (185 °F)

2.2 Power Consumption

Symbol	Description	Min	Typ	Max	Unit
VINMax	Maximum input voltage from VIN pad	-0.3	-	21	V
VUSBMax	Maximum input voltage from USB connector	-0.3	-	21	V
PMax	Maximum Power Consumption	-	-	TBC	mW

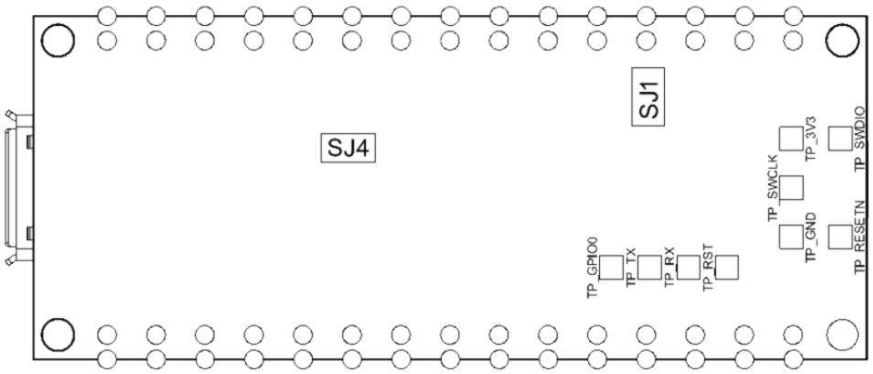
3 Functional Overview

3.1 Board Topology



Board topology top

Ref.	Description	Ref.	Description
U1	ATSAMD21G18A Controller	U3	LSM6DSOXTR IMU Sensor
U2	NINA-W102-00B WiFi/BLE Module	U4	ATECC608A-MAHDA-T Crypto Chip
J1	Micro USB Connector	PB1	IT-1185-160G-GTR Push button



Board topology bottom

Ref.	Description	Ref.	Description
SJ1	Open solder bridge (VUSB)	SJ4	Closed solder bridge (+3V3)
TP	Test points	xx	Lorem Ipsum

3.2 Processor

The Main Processor is a Cortex M0+ running at up to 48MHz. Most of its pins are connected to the external headers, however some are reserved for internal communication with the wireless module and the on-board internal I^2C peripherals (IMU and Crypto).

NOTE: As opposed to other Arduino Nano boards, pins A4 and A5 have an internal pull up and default to be used as an I^2C Bus so usage as analog inputs is not recommended.

Communication with NINA W102 happens through a serial port and a SPI bus through the following pins.

SAMD21 Pin	SAMD21 Acronym	NINA Pin	NINA Acronym	Description
13	PA08	19	RESET_N	Reset
39	PA27	27	GPIO0	Attention Request
41	PA28	7	GPIO33	Acknowledge
23	PA14	28	GPIO5	SPI CS
21	GPIO19	UART RTS		
24	PA15	29	GPIO18	SPI CLK
20	GPIO22	UART CTS		
22	PA13	1	GPIO21	SPI MISO
21	PA12	36	GPIO12	SPI MOSI
31	PA22	23	GPIO3	Processor TX Nina RX
32	PA23	22	GPIO1	Processor RX Nina TX

3.3 WiFi/BT Communication Module

Nina W102 is based on ESP32 and is delivered with a pre-certified software stack from Arduino. Source code for the firmware is available **[9]**.

NOTE: Reprogramming the wireless module's firmware with a custom one will invalidate compliance with radio standards as certified by Arduino, hence this is not recommended unless the application is used in private laboratories far from other electronic equipment and people. Usage of custom firmware on radio modules is the sole responsibility of the user.

Some of the module's pins are connected to the external headers and can be directly driven by ESP32 provided SAMD21's corresponding pins are aptly tri-stated. Below is a list of such signals:

SAMD21 Pin	SAMD21 Acronym	NINA Pin	NINA Acronym	Description
48	PB03	8	GPIO21	A7
14	PA09	5	GPIO32	A6
8	PB09	31	GPIO14	A5/SCL
7	PB08	35	GPIO13	A4/SDA

3.4 Crypto

The crypto chip in Arduino IoT boards is what makes the difference with other less secure boards as it provides a secure way to store secrets (such as certificates) and accelerates secure protocols while never exposing secrets in plain text.

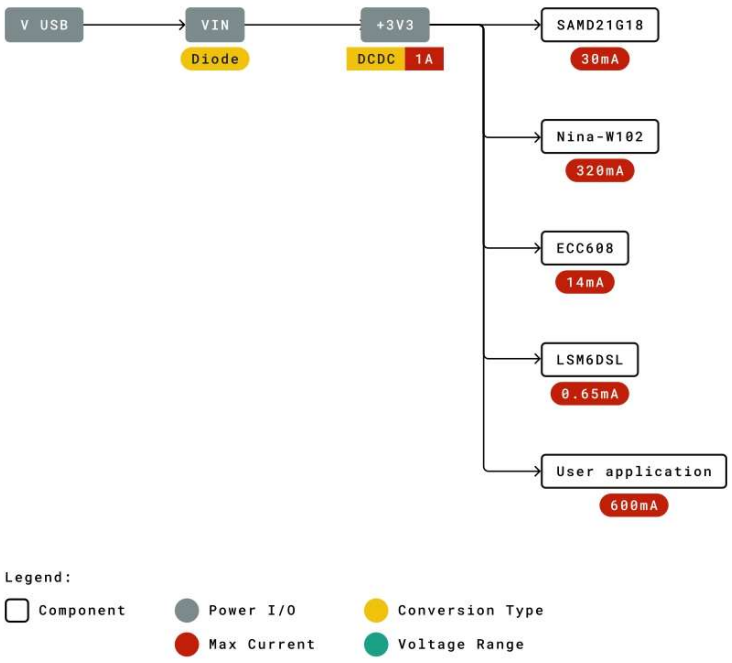
Source code for the Arduino Library that supports the Crypto is available **[10]**

3.5 IMU

The board has an embedded 6 axis IMU which can be used to measure board orientation (by checking the gravity acceleration vector orientation) or to measure shocks, vibration, acceleration and rotation speed.

Source code for the Arduino Library that supports the IMU is available **[11]**

3.6 Power Tree



Power tree

4 Board Operation

4.1 Getting Started - IDE

If you want to program your board while offline you need to install the Arduino Desktop IDE [1] To connect the Arduino 33 IoT to your computer, you'll need a Micro-B USB cable. This also provides power to the board, as indicated by the LED.

4.2 Getting Started - Arduino Web Editor

All Arduino boards, including this one, work out-of-the-box on the Arduino Web Editor [2], by just installing a simple plugin.

The Arduino Web Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow **[3]** to start coding on the browser and upload your sketches onto your board.

4.3 Getting Started - Arduino IoT Cloud

All Arduino IoT enabled products are supported on Arduino IoT Cloud which allows you to Log, graph and analyze sensor data, trigger events, and automate your home or business.

4.4 Sample Sketches

Sample sketches for the Arduino 33 IoT can be found either in the "Examples" menu in the Arduino IDE or in the "Documentation" section of the Arduino Pro website [4]

4.5 Online Resources

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking exciting projects on ProjectHub **[5]**, the Arduino Library Reference **[6]** and the online store **[7]** where you will be able to complement your board with sensors, actuators and more

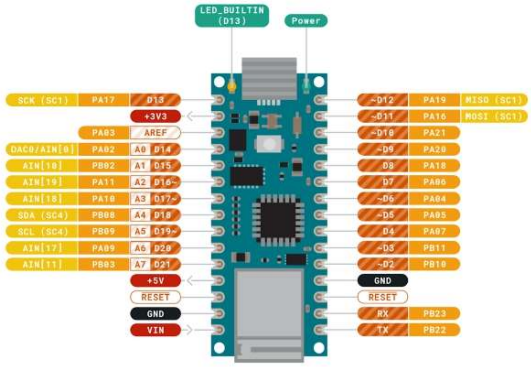
4.6 Board Recovery

All Arduino boards have a built-in bootloader which allows flashing the board via USB. In case a sketch locks up the processor and the board is not reachable anymore via USB it is possible to enter bootloader mode by double-tapping the reset button right after power up.

5 Connector Pinouts

ARDUINO

ARDUINO
NANO 33 IoT



Pinout

5.1 USB

Pin	Function	Type	Description
1	VUSB	Power	Power Supply Input. If board is powered via VUSB from header this is an Output **(1)**
2	D-	Differential	USB differential data -
3	D+	Differential	USB differential data +
4	ID	Analog	Selects Host/Device functionality
5	GND	Power	Power Ground

1. The board can support USB host mode only if powered via the V_{USB} pin and if the jumper close to the VUSB pin is shorted.

5.2 Headers

The board exposes two 15 pin connectors which can either be assembled with pin headers or soldered through castellated vias.

Pin	Function	Type	Description
1	D13	Digital	GPIO
2	+3V3	Power Out	Internally generated power output to external devices
3	AREF	Analog	Analog Reference; can be used as GPIO
4	A0/DAC0	Analog	ADC in/DAC out; can be used as GPIO
5	A1	Analog	ADC in; can be used as GPIO
6	A2	Analog	ADC in; can be used as GPIO
7	A3	Analog	ADC in; can be used as GPIO
8	A4/SDA	Analog	ADC in; I2C SDA; Can be used as GPIO **(1)**
9	A5/SCL	Analog	ADC in; I2C SCL; Can be used as GPIO **(1)**
10	A6	Analog	ADC in; can be used as GPIO
11	A7	Analog	ADC in; can be used as GPIO
12	VUSB	Power In/Out	Normally NC; can be connected to VUSB pin of the USB connector by shorting a jumper
13	RST	Digital In	Active low reset input (duplicate of pin 18)
14	GND	Power	Power Ground
15	VIN	Power In	Vin Power input
16	TX	Digital	USART TX; can be used as GPIO
17	RX	Digital	USART RX; can be used as GPIO
18	RST	Digital	Active low reset input (duplicate of pin 13)
19	GND	Power	Power Ground
20	D2	Digital	GPIO
21	D3/PWM	Digital	GPIO; can be used as PWM
22	D4	Digital	GPIO
23	D5/PWM	Digital	GPIO; can be used as PWM
24	D6/PWM	Digital	GPIO, can be used as PWM
25	D7	Digital	GPIO
26	D8	Digital	GPIO
27	D9/PWM	Digital	GPIO; can be used as PWM
28	D10/PWM	Digital	GPIO; can be used as PWM
29	D11/MOSI	Digital	SPI MOSI; can be used as GPIO
30	D12/MISO	Digital	SPI MISO; can be used as GPIO

5.3 Debug

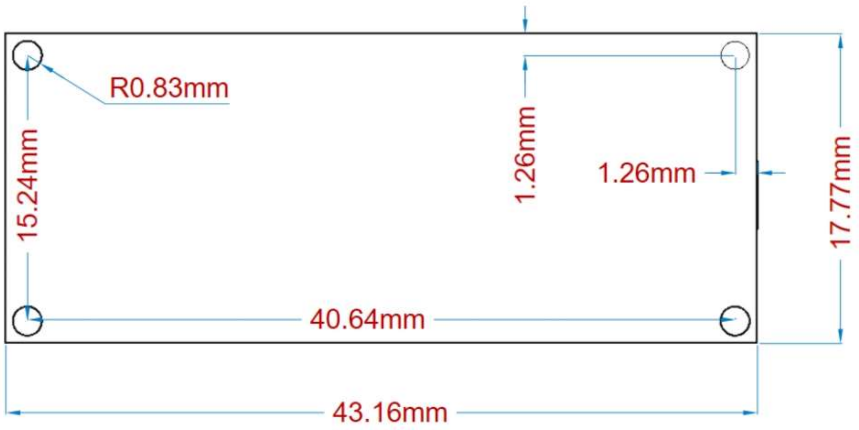
On the bottom side of the board, under the communication module, debug signals are arranged as 3x2 test pads with 100 mil pitch. Pin 1 is depicted in Figure 3 – Connector Positions

Pin	Function	Type	Description
1	+3V3	Power Out	Internally generated power output to be used as voltage reference
2	SWD	Digital	SAMD11 Single Wire Debug Data
3	SWCLK	Digital In	SAMD11 Single Wire Debug Clock
4	UPDI	Digital	ATMega4809 update interface
5	GND	Power	Power Ground
6	RST	Digital In	Active low reset input

6 Mechanical Information

6.1 Board Outline and Mounting Holes

The board measures are mixed between metric and imperial. Imperial measures are used to maintain a 100 mil pitch grid between pin rows to allow them to fit a breadboard whereas board length is Metric.

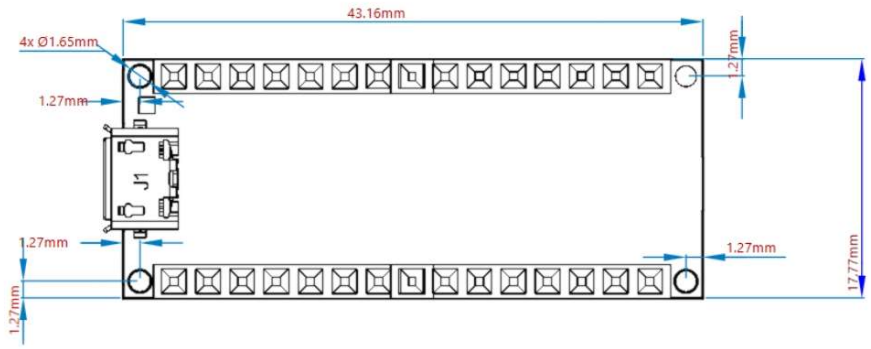


Layout

6.2 Connector Positions

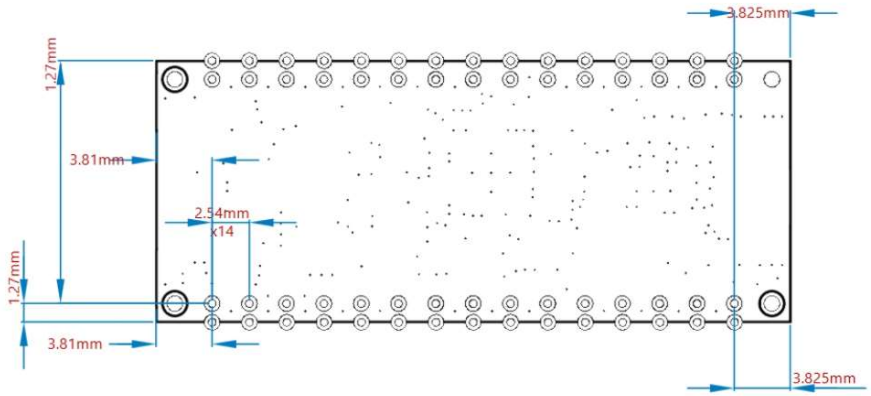
The view below is from top however it shows Debug connector pads which are on the bottom side. Highlighted pins are pin 1 for each connector'

Top view:



Top side connectors

Bottom view:



Bottom side connectors

7 Certifications

7.1 Declaration of Conformity CE DoC (EU)

We declare under our sole responsibility that the products above are in conformity with the essential requirements of the following EU Directives and therefore qualify for free movement within markets comprising the European Union (EU) and European Economic Area (EEA).

7.2 Declaration of Conformity to EU RoHS & REACH 211 01/19/2021

Arduino boards are in compliance with RoHS 2 Directive 2011/65/EU of the European Parliament and RoHS 3 Directive 2015/863/EU of the Council of 4 June 2015 on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

Substance	Maximum limit (ppm)
Lead (Pb)	1000
Cadmium (Cd)	100
Mercury (Hg)	1000
Hexavalent Chromium (Cr6+)	1000
Poly Brominated Biphenyls (PBB)	1000
Poly Brominated Diphenyl ethers (PBDE)	1000
Bis(2-Ethylhexyl} phthalate (DEHP)	1000
Benzyl butyl phthalate (BBP)	1000
Dibutyl phthalate (DBP)	1000
Diisobutyl phthalate (DIBP)	1000

Exemptions : No exemptions are claimed.

Arduino Boards are fully compliant with the related requirements of European Union Regulation (EC) 1907 /2006 concerning the Registration, Evaluation, Authorization and Restriction of Chemicals (REACH). We declare none of the SVHCs (https://echa.europa.eu/web/guest/candidate-list-table), the Candidate List of Substances of Very High Concern for authorization currently released by ECHA, is present in all products (and also package) in quantities totaling in a concentration equal or above 0.1%. To the best of our knowledge, we also declare that our products do not contain any of the substances listed on the "Authorization List" (Annex XIV of the REACH regulations) and Substances of Very High Concern (SVHC) in any significant amounts as specified by the Annex XVII of Candidate list published by ECHA (European Chemical Agency) 1907 /2006/EC.

7.3 Conflict Minerals Declaration

As a global supplier of electronic and electrical components, Arduino is aware of our obligations with regards to laws and regulations regarding Conflict Minerals, specifically the Dodd-Frank Wall Street Reform and Consumer Protection Act, Section 1502. Arduino does not directly source or process conflict minerals such as Tin, Tantalum, Tungsten, or Gold. Conflict minerals are contained in our products in the form of solder, or as a component in metal alloys. As part of our reasonable due diligence Arduino has contacted component suppliers within our supply chain to verify their continued compliance with the regulations. Based on the information received thus far we declare that our products contain Conflict Minerals sourced from conflict-free areas.

8 FCC Caution

Any Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference

(2) this device must accept any interference received, including interference that may cause undesired operation.

FCC RF Radiation Exposure Statement:

1. This Transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.
2. This equipment complies with RF radiation exposure limits set forth for an uncontrolled environment.
3. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

English: User manuals for license-exempt radio apparatus shall contain the following or equivalent notice in a conspicuous location in the user manual or alternatively on the device or both. This device complies with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions:

(1) this device may not cause interference

(2) this device must accept any interference, including interference that may cause undesired operation of the device.

French: Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes :

(1) l' appareil nedoit pas produire de brouillage

(2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

IC SAR Warning:

English This equipment should be installed and operated with minimum distance 20 cm between the radiator and your body.

French: Lors de l' installation et de l' exploitation de ce dispositif, la distance entre le radiateur et le corps est d 'au moins 20 cm.

Important: The operating temperature of the EUT can't exceed 85°C and shouldn't be lower than -40°C.

Hereby, Arduino S.r.l. declares that this product is in compliance with essential requirements and other relevant provisions of Directive 2014/53/EU. This product is allowed to be used in all EU member states.

Frequency bands	Maximum output power (EIRP)
2402-2480MHz(EDR)	6.24 dBm
2402-2480MHz(BLE)	6.30 dBm
2412-2472MHz(2.4G WiFi)	13.61 dBm

9 Company Information

Company name	Arduino S.r.l
Company Address	Via Andrea Appiani,2520900 MONZA

10 Reference Documentation

Reference	Link
Arduino IDE (Desktop)	https://www.arduino.cc/en/software
Arduino IDE (Cloud)	https://create.arduino.cc/editor
Cloud IDE Getting Started	https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-4b3e4a
Forum	http://forum.arduino.cc/
SAMD21G18	https://ww1.microchip.com/downloads/aemDocuments/documents/MCU32/ProductDocuments/DataSheets/SAM-D21DA1-Family-Data-Sheet-DS40001882G.pdf
NINA W102	https://content.u-blox.com/sites/default/files/NINA-W10_DataSheet_UBX-17065507.pdf
ECC608	https://ww1.microchip.com/downloads/aemDocuments/documents/SCBU/ProductDocuments/DataSheets/ATECC608A-CryptoAuthentication-Device-Summary-Data-Sheet-DS40001977B.pdf
MPM3610	https://www.monolithicpower.com/pub/media/document/MPM3610_r1.01.pdf
NINA Firmware	https://github.com/arduino/nina-fw
ECC608 Library	https://github.com/arduino-libraries/ArduinoECCX08
LSM6DSL Library	https://github.com/stm32duino/LSM6DSL
ProjectHub	https://create.arduino.cc/projecthub?by=part&part_id=11332&sort=trending
Library Reference	https://www.arduino.cc/reference/en/
Arduino Store	https://store.arduino.cc/

11 Revision History

Date	Revision	Changes
03/08/2022	2	Reference documentation links updates
15/04/2021	1	General datasheet updates

6. HC-SR04

ELEC Freaks
www.elecfreaks.com

HC-SR04 User Guide

Part 1 Ultrasonic Introduction

1. 1 Ultrasonic Definition

The human ear can hear sound frequency around 20HZ ~ 20KHZ, and ultrasonic is the sound wave beyond the human ability of 20KHZ .

1.2 Ultrasonic distance measurement principle

Ultrasonic transmitter emitted an ultrasonic wave in one direction, and started timing when it launched. Ultrasonic spread in the air, and would return immediately when it encountered obstacles on the way. At last, the ultrasonic receiver would stop timing when it received the reflected wave. As Ultrasonic spread velocity is 340m / s in the air, based on the timer record **t**, we can calculate the distance (s) between the obstacle and transmitter, namely: s = 340t / 2, which is so- called time difference distance measurement principle

The principle of ultrasonic distance measurement used the already-known air spreading velocity, measuring the time from launch to reflection when it encountered obstacle, and then calculate the distance between the transmitter and the obstacle according to the time and the velocity. Thus, the principle of ultrasonic distance measurement is the same with radar.
Distance Measurement formula is expressed as: L = C X T
In the formula, L is the measured distance, and C is the ultrasonic spreading velocity in air, also, T represents time (T is half the time value from transmitting to receiving).

1.3 Ultrasonic Application

Ultrasonic Application Technology is the thing which developed in recent decades. With the ultrasonic advance, and the electronic technology development, especially as high-power semiconductor device technology matures, the application of ultrasonic has become increasingly widespread:

- Ultrasonic measurement of distance, depth and thickness;
- Ultrasonic testing;
- Ultrasound imaging;
- Ultrasonic machining, such as polishing, drilling;
- Ultrasonic cleaning;
- Ultrasonic welding;

Part 2 HC-SR04 Ultrasonic Module Introduction

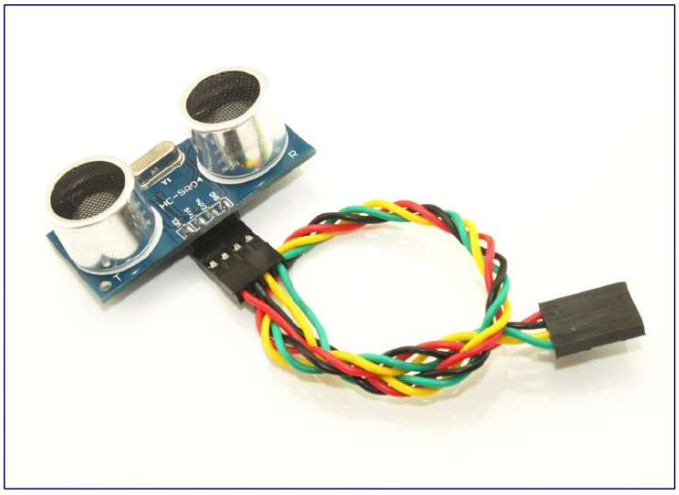
2.1 Product Features

- Stable performance
- Accurate distance measurement
- High-density
- Small blind

Application Areas:

- Robotics barrier
- Object distance measurement
- Level detection
- Public security
- Parking detection

2.2 Product Image



2.3、Module pin definitions

Types	Pin Symbol	Pin Function Description
HC-SR04	VCC	5V power supply
	Trig	Trigger pin
	Echo	Receive pin
	GND	Power ground

2.4、Electrical parameters

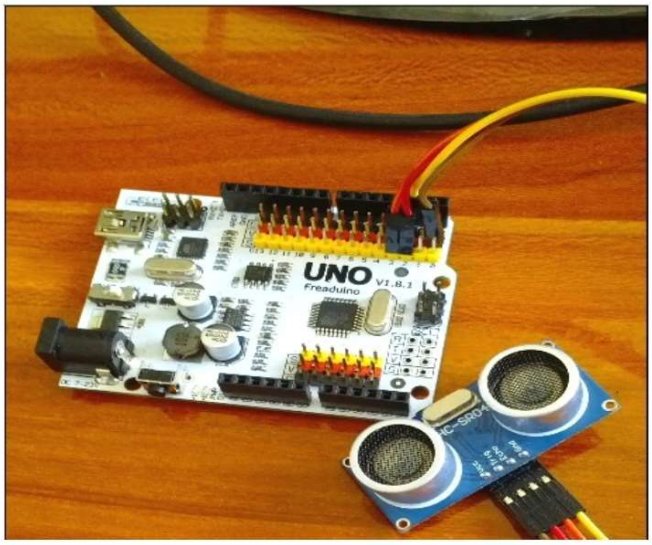
Electrical Parameters	HC-SR04 Ultrasonic Module
Operating Voltage	DC-5V
Operating Current	15mA
Operating Frequency	40KHZ
Farthest Range	4m
Nearest Range	2cm
Measuring Angle	15 Degree
Input Trigger Signal	10us TTL pulse
Output Echo Signal	Output TTL level signal, proportional with range
Dimensions	45*20*15mm

2.5 Module operating Principle

Set low the Trig and Echo port when the module initializes , firstly, transmit at least 10us high level pulse to the Trig pin (module automatically sends eight 40K square wave), and then wait to capture the rising edge output by echo port, at the same time, open the timer to start timing. Next, once again capture the falling edge output by echo port, at the same time, read the time of the counter, which is the ultrasonic running time in the air. According to the formular: test distance = (high level time * ultrasonic spreading velocity in air) / 2, you can calculate the distance to the obstacle.

Part3 Use Freaduino UNO to test HC-SR04

3.1 Freaduino uno and HC-SR04 Connection



Connection Description：D2<------>Trig D3<------>Echo (The users can define the connection pin by themselves)

Note：You need to set the Freaduino UNO switch in 5V Side when use together with HC-SR04 Module.

3.2 HCSR04 library function description

Long timing()

Function name: timing
Parameters: None
Return Value: the time of ultrasonic from the transmitter to the receiver

float CalcDistance(long microsec,int metric)

Function name：CalcDistance

- microsec: the time of ultrasonic from the transmitter to the receiver
- metric: Set the unit of the return value (the value of 1 for cm, and the value of 0 for in)

Return Value: the measured distance

3.3 Add the HC-SR04 Library

Step1:Download the Demo Code of HCSR04 Ultrasonic from address http://www.elecfreaks.com/store/download/product/Sensor/HC-SR04/HCSR04Ultrasonic_demo.zip and then unpack it to get the file of HCSR04 Ultrasonic.

Step2: Add the file of HCSR04 Ultrasonic in the file of Arduino-1.0.X / libraries.

Step3:If you can see the Example of HCSR04 Ultrasonic in Arduino IDE, the adding of HC-SR04 library has been successful.

3.4 Test the Module with the Examples of Library File

1. Open Arduino IDE 1.0.X, and choose the corresponding board and serial port.
2. Click file/ examples/ HCSR04Ultrasonic until the code pop up.
3. Compiling sketch until Done uploading appears, which represents the uploading has been successful.
4. Open serial monitor and set the corresponding BaudRate.
5. If you see similar information in serial monitor as below, you succeeded.

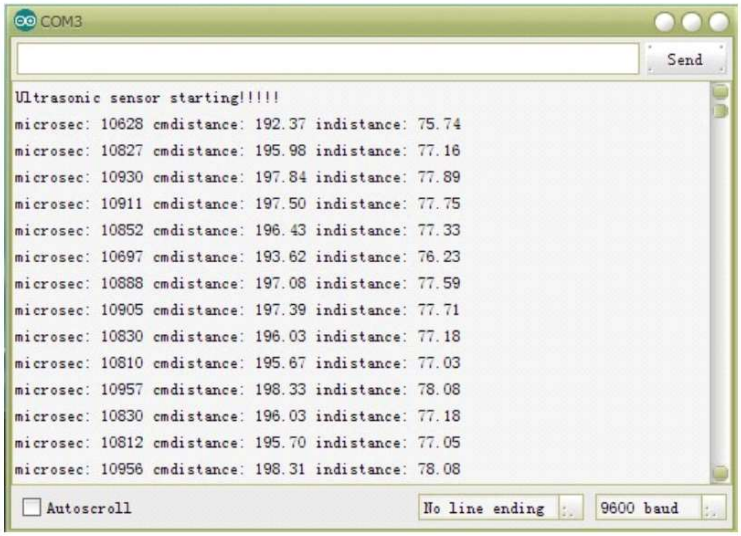
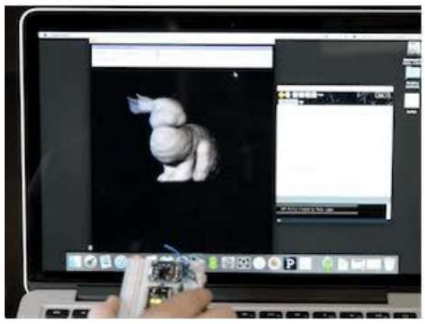


Chart 3、HC-SR04 testing results

7. BNO055

Adafruit BNO055 Absolute Orientation Sensor

Created by Kevin Townsend



https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor

Last updated on 2023-04-13 04:30:03 PM EDT

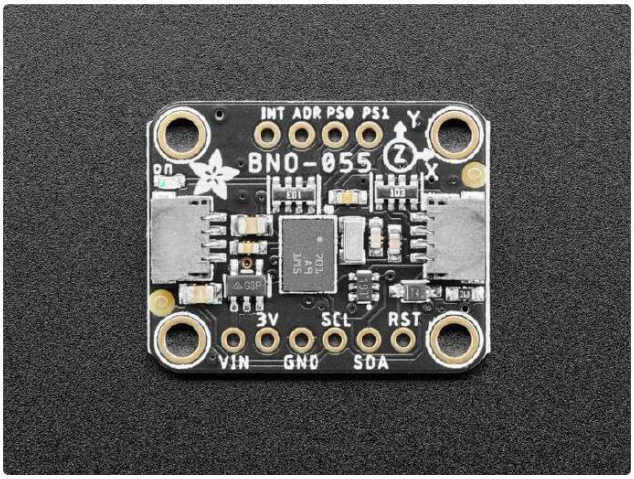
Table of Contents

Overview

If you've ever ordered and wire up a 9-DOF sensor, chances are you've also realized the challenge of turning the sensor data from an accelerometer, gyroscope and magnetometer into actual "3D space orientation"! Orientation is a hard problem to solve. The sensor fusion algorithms (the secret sauce that blends accelerometer, magnetometer and gyroscope data into stable three-axis orientation output) can be mind-numbingly difficult to get right and implement on low cost real time systems.

Bosch is the first company to get this right by taking a MEMS accelerometer, magnetometer and gyroscope and putting them on a single die with a high speed ARM Cortex-M0 based processor to digest all the sensor data, abstract the sensor fusion and real time requirements away, and spit out data you can use in quaternions, Euler angles or vectors.

The BNO055 I2C implementation violates the I2C protocol in some circumstances. This causes it not to work well with certain chip families. It does not work well with Espressif ESP32, ESP32-S3, and NXP i.MX RT1011, and it does not work well with I2C multiplexers. Operation with SAMD51, RP2040, STM32F4, and nRF52840 is more reliable.



The new version of the board includes SparkFun qwiic () compatible STEMMA QT () connectors for the I2C bus so you don't even need to solder! Use a a plug-and-play STEMMA QT cable to get 9 DoF data ASAP.

Rather than spending weeks or months fiddling with algorithms of varying accuracy and complexity, you can have meaningful sensor data in minutes thanks to the BNO055 - a smart 9-DOF sensor that does the sensor fusion all on its own!

Data Output

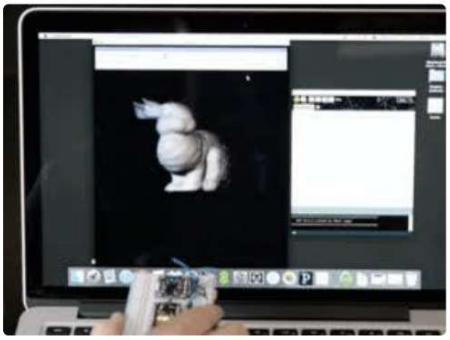
The BNO055 can output the following sensor data:

- Absolute Orientation (Euler Vector, 100Hz)
  Three axis orientation data based on a 360° sphere
- Absolute Orientation (Quaterion, 100Hz)
  Four point quaternion output for more accurate data manipulation
- Angular Velocity Vector (100Hz)
  Three axis of 'rotation speed' in rad/s
- Acceleration Vector (100Hz)
  Three axis of acceleration (gravity + linear motion) in m/s^2
- Magnetic Field Strength Vector (20Hz)
  Three axis of magnetic field sensing in micro Tesla (uT)
- Linear Acceleration Vector (100Hz)
  Three axis of linear acceleration data (acceleration minus gravity) in m/s^2

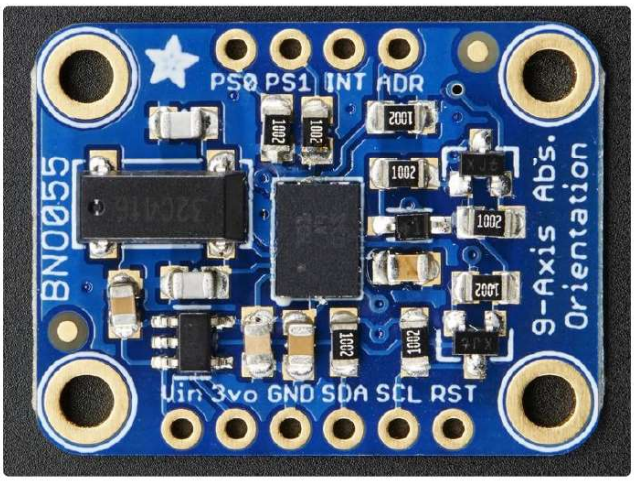
- Gravity Vector (100Hz)
  Three axis of gravitational acceleration (minus any movement) in m/s^2
- Temperature (1Hz)
  Ambient temperature in degrees celsius

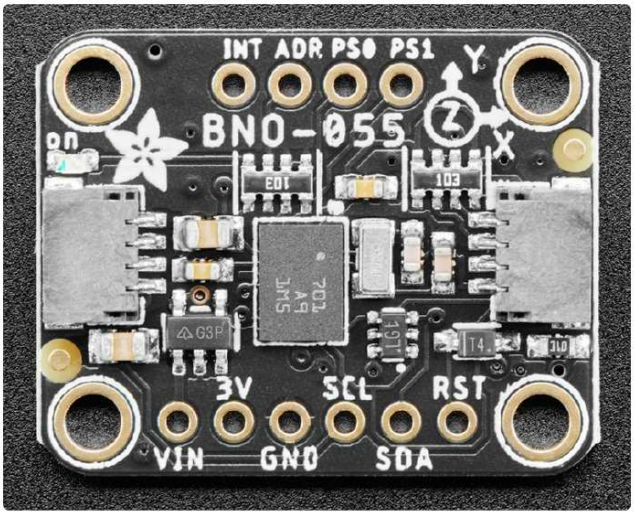
Related Resources

- Datasheet ()
- Adafruit BNO055 Library () (Github)



Pinouts





Note: The pin order on the STEMMA QT version of the board is not the same as the original version. The pins are the same otherwise.

The BNO055 I2C implementation violates the I2C protocol in some circumstances. This causes it not to work well with certain chip families. It does not work well with Espressif ESP32, ESP32-S3, and NXP i.MX RT1011, and it does not work well with I2C multiplexers. Operation with SAMD51, RP2040, STM32F4, and nRF52840 is more reliable.

Power Pins

- VIN: 3.3-5.0V power supply input
- 3VO: 3.3V output from the on-board linear voltage regulator, you can grab up to about 50mA as necessary
- GND: The common/GND pin for power and logic

I2C Pins

- SCL - I2C clock pin, connect to your microcontrollers I2C clock line. This pin can be used with 3V or 5V logic, and there's a 10K pullup on this pin.
- SDA - I2C data pin, connect to your microcontrollers I2C data line. This pin can be used with 3V or 5V logic, and there's a 10K pullup on this pin.

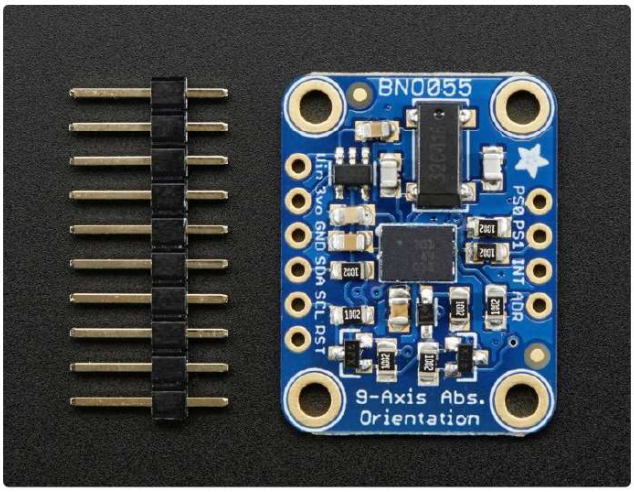
STEMMA QT version

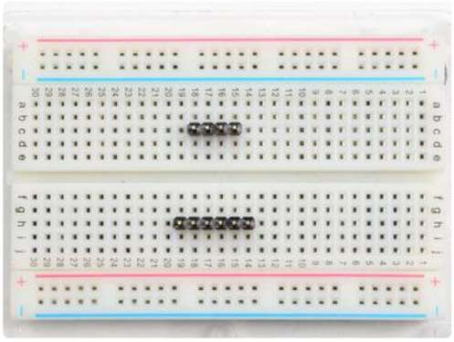
- STEMMA QT () - These connectors allow you to connectors to dev boards with S TEMMA QT connectors or to other things with various associated accessories ()

Other Pins

- RST: Hardware reset pin. Set this pin low then high to cause a reset on the sensor. This pin is 5V safe.
- INT: The HW interrupt output pin, which can be configured to generate an interrupt signal when certain events occur like movement detected by the accelerometer, etc. (not currently supported in the Adafruit library, but the chip and HW is capable of generating this signal). The voltage level out is 3V
- ADR: Set this pin high to change the default I2C address for the BNO055 if you need to connect two ICs on the same I2C bus. The default address is 0x28. If this pin is connected to 3V, the address will be 0x29
- PS0 and PS1: These pins can be used to change the mode of the device (it can also do HID-I2C and UART) and also are provided in case Bosch provides a firmware update at some point for the ARM Cortex M0 MCU inside the sensor. They should normally be left unconnected.

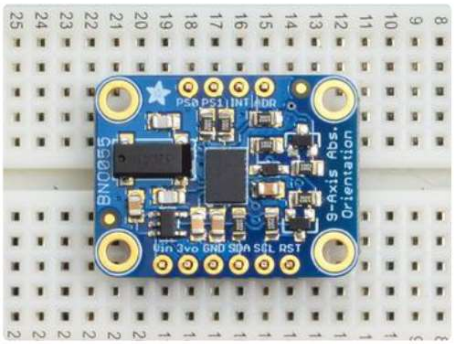
Assembly





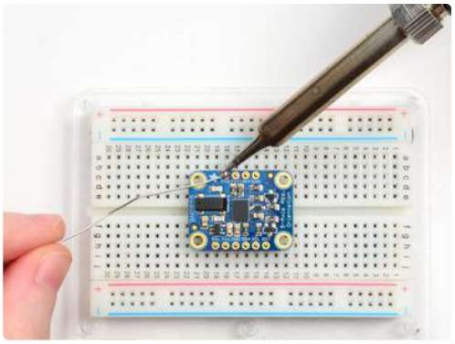
Prepare the header strip:

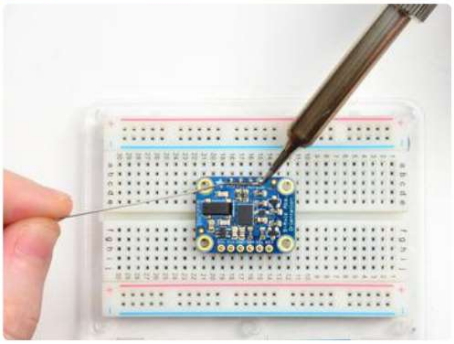
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - long pins down

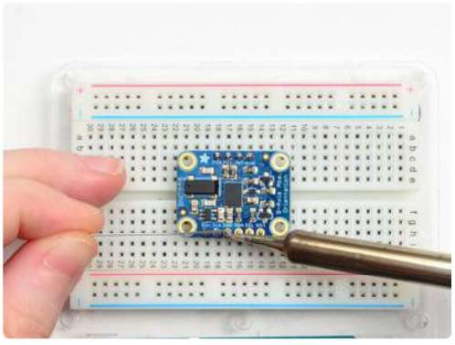


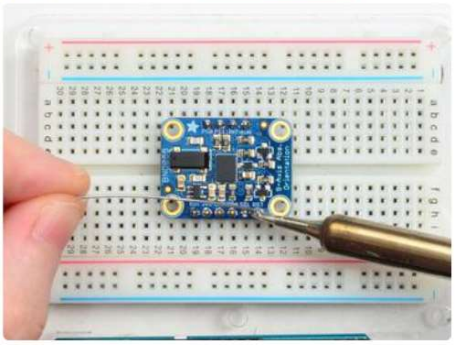
Add the breakout board:

Place the breakout board over the pins so that the short pins poke through the breakout pads







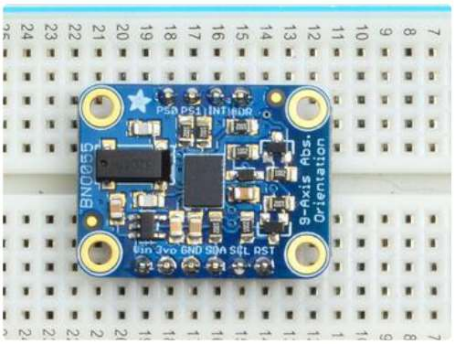


And Solder!

Be sure to solder all pins for reliable electrical contact.

Solder the longer power/data strip first

(For tips on soldering, be sure to check out our Guide to Excellent Soldering ()).

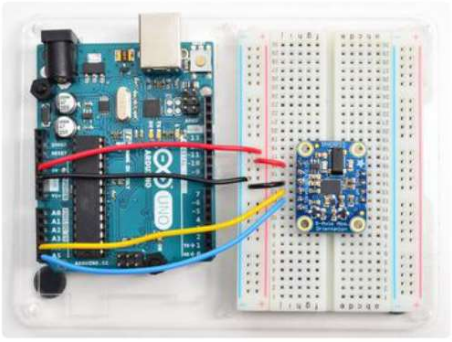


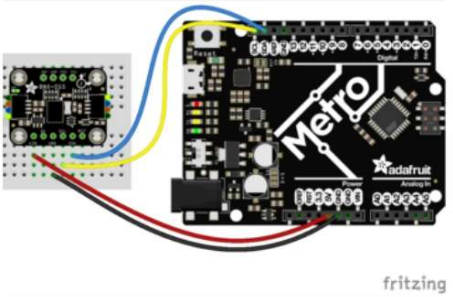
You're done! Check your solder joints visually and continue onto the next steps

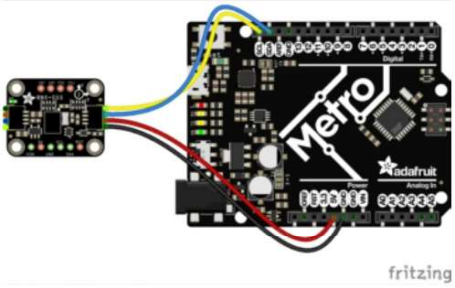
Arduino Code

Wiring for Arduino

You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, just make sure it has I2C capability, then port the code - its pretty simple stuff!







To connect the assembled BNO055 breakout to an Arduino Uno, follow the wiring diagram.

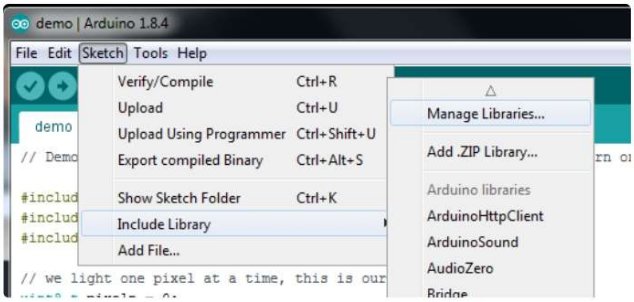
Connect Vin (red wire) to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
Connect GND (black wire) to common power/data ground
Connect the SCL (yellow wire) pin to the I2C clock SCL pin on your Arduino. On an UNO & '328 based Arduino, this is also known as A5, on a Mega it is also known as digital 21 and on a Leonardo/Micro, digital 3
Connect the SDA (blue wire) pin to the I2C data SDA pin on your Arduino. On an UNO & '328 based Arduino, this is also known as A4, on a Mega it is also known as digital 20 and on a Leonardo/Micro, digital 2

If you're using a Genuino Zero or Arduino Zero with the built in EDBG interface you may need to use I2C address 0x29 since 0x28 is 'taken' by the DBG chip

Software

The Adafruit_BNO055 driver () supports reading raw sensor data, or you can use the Adafruit Unified Sensor () system to retrieve orientation data in a standard data format.

Open up the Arduino library manager:



Search for the Adafruit Sensor library and install it

Search for the Adafruit BNO055 library and install it

We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use ()

Adafruit Unified Sensor System

Since the Adafruit_BNO055 driver is based on the Adafruit Unified Sensor system, you can retrieve your three axis orientation data (in Euler angles) using the standard types and functions described in the Adafruit Sensor learning guide () (.getEvent (), .getSensor (), etc.).

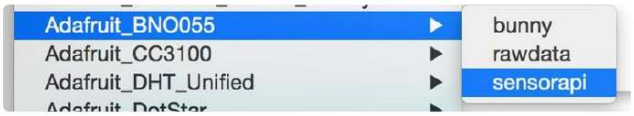
This is probably the easiest option if all you care about is absolute orientation data across three axis.

For example, the following code snippet shows the core of what is needed to start reading data using the Unified Sensor System:

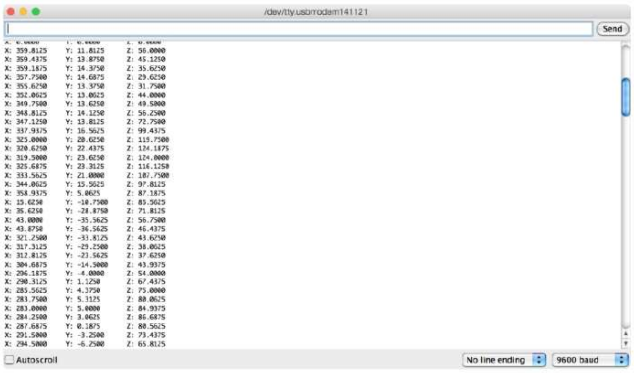
```
#include &lt;Wire.h&gt;
#include &lt;Adafruit_Sensor.h&gt;
#include &lt;Adafruit_BNO055.h&gt;
#include &lt;utility/imumaths.h&gt;

Adafruit_BNO055 bno = Adafruit_BNO055(55);

void setup(void)
{
  Serial.begin(9600);
  Serial.println("Orientation Sensor Test"); Serial.println("");

  /* Initialise the sensor */
  if(!bno.begin())
  {
    /* There was a problem detecting the BNO055 ... check your connections */
    Serial.print("Ooops, no BNO055 detected ... Check your wiring or I2C ADDR!");
    while(1);
  }

  delay(1000);

  bno.setExtCrystalUse(true);
}

void loop(void)
{
  /* Get a new sensor event */
  sensors_event_t event;
  bno.getEvent(&amp;event);

  /* Display the floating point data */
  Serial.print("X: ");
  Serial.print(event.orientation.x, 4);
  Serial.print("\tY: ");
  Serial.print(event.orientation.y, 4);
  Serial.print("\tZ: ");
  Serial.print(event.orientation.z, 4);
  Serial.println("");

  delay(100);
}
```

'sensorapi' Example

To test the Unified Sensor System output, open the sensorapi demo in the Adafruit_BNO055 examples folder:



This should produce the following output on the Serial Monitor:



Raw Sensor Data

If you don't want to use the Adafruit Unified Sensor system (for example if you want to access the raw accelerometer, magnetometer or gyroscope data directly before the sensor fusion algorithms process it), you can use the raw helper functions in the driver.

The key raw data functions are:

- getVector (adafruit_vector_type_t vector_type)
- getQuat (void)
- getTemp (void)

.getVector (adafruit_vector_type_t vector_type)

The .getVector function accepts a single parameter (vector_type), which indicates what type of 3-axis vector data to return.

The vector_type field can be one of the following values:

- VECTOR_MAGNETOMETER (values in uT, micro Teslas)
- VECTOR_GYROSCOPE (values in rps, radians per second)
- VECTOR_EULER (values in Euler angles or 'degrees', from 0..359)
- VECTOR_ACCELEROMETER (values in m/s^2)
- VECTOR_LINEARACCEL (values in m/s^2)
- VECTOR_GRAVITY (values in m/s^2)

For example, to get the Euler angles vector, we could run the following code:

```
imu::Vector&lt;3&gt; euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);

/* Display the floating point data */
Serial.print("X: ");
Serial.print(euler.x());
Serial.print(" Y: ");
Serial.print(euler.y());
Serial.print(" Z: ");
Serial.print(euler.z());
Serial.println("");
```

.getQuat(void)

The .getQuat function returns a Quaternion, which is often easier and more accurate to work with than Euler angles when doing sensor fusion or data manipulation with raw sensor data.

You can get a quaternion data sample via the following code:

```
imu::Quaternion quat = bno.getQuat();

/* Display the quat data */
Serial.print("qW: ");
Serial.print(quat.w(), 4);
Serial.print(" qX: ");
Serial.print(quat.y(), 4);
Serial.print(" qY: ");
Serial.print(quat.x(), 4);
Serial.print(" qZ: ");
Serial.print(quat.z(), 4);
Serial.println("");
```

.getTemp(void)

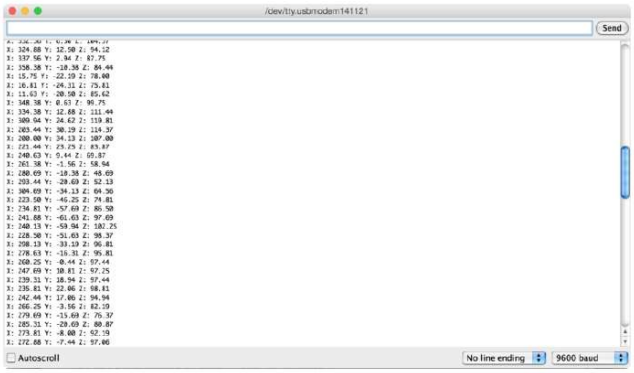
The .getTemp helper returns the current ambient temperature in degrees celsius, and can be read via the following function call:

```
/* Display the current temperature */
int8_t temp = bno.getTemp();

Serial.print("Current Temperature: ");
Serial.print(temp);
Serial.println(" C");
Serial.println("");
```

'rawdata' Example

To test the raw data ouput, open the rawdata demo in the Adafruit_BNO055 examples folder:

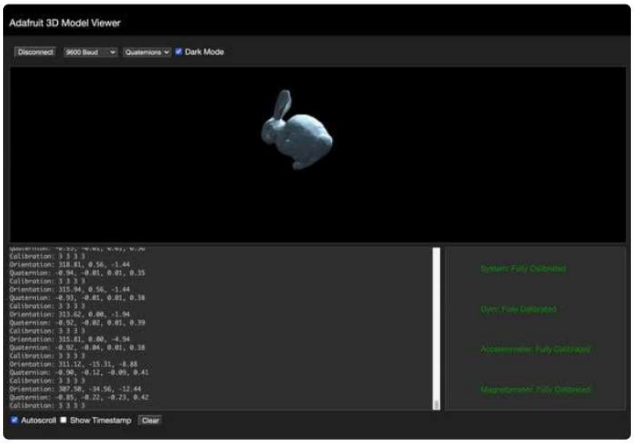
This should produce the following output on the Serial Monitor:



By default, the sketch generates Euler angle absolute orientation data, but you can easily modify the data displayed by changing the value provided to .getVector below:

```
// Possible vector values can be:
// - VECTOR_ACCELEROMETER - m/s^2
// - VECTOR_MAGNETOMETER  - uT
// - VECTOR_GYROSCOPE     - rad/s
// - VECTOR_EULER         - degrees
// - VECTOR_LINEARACCEL   - m/s^2
// - VECTOR_GRAVITY       - m/s^2
imu::Vector&lt;3&gt; euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);

/* Display the floating point data */
Serial.print("X: ");
Serial.print(euler.x());
Serial.print(" Y: ");
Serial.print(euler.y());
Serial.print(" Z: ");
Serial.print(euler.z());
Serial.println("");
```

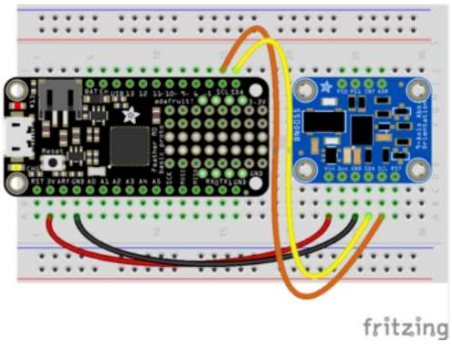
WebSerial Visualizer



That raw data is all fine and good, but we want to see what they mean in 3D space, right? Traditionally, a Processing sketch would be used to read the serial data and convert it to a 3D rotation - but thanks to Web Serial API we can use any Chrome browser - a lot easier than installing Processing! ()

Step 1 - Wire up the BNO055 to your Microcontroller using I2C

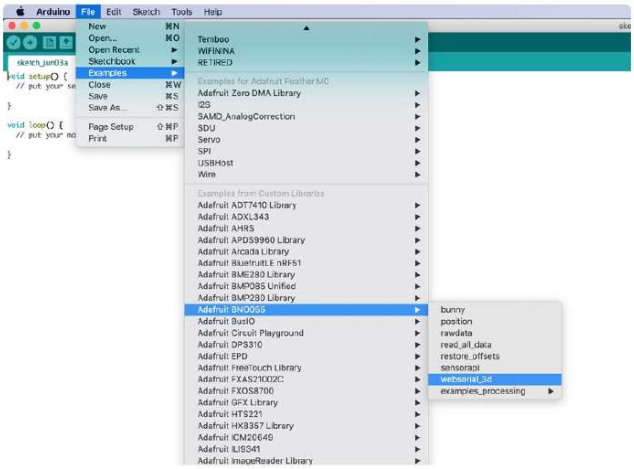
First wire up a BNO055 to your board exactly as shown on the previous pages using the I2C interface. Here's an example of wiring a Feather M0 to the sensor with I2C:



Board 3V to sensor VIN
Board GND to sensor GND
Board SCL to sensor SCL
Board SDA to sensor SDA

Step 2 - Load the Sketch onto your device

Continue by making sure you still have the Arduino IDE open and have the latest version of the Adafruit BNO055 library installed. Open the sketch at Examples → Adafruit BNO055 → webserial_3d



Upload the sketch to your Microcontroller Board.

Step 3 - Install Chrome

Start by installing the Chrome browser if you haven't yet. ()

Step 4 - Enable Web Serial API if necessary

As of Chrome 89, Web Serial is enabled by default.

At the time of this tutorial, you'll need to enable the Serial API, which is really easy.

Visit chrome://flags from within Chrome. Find and enable the Experimental Web Platform features

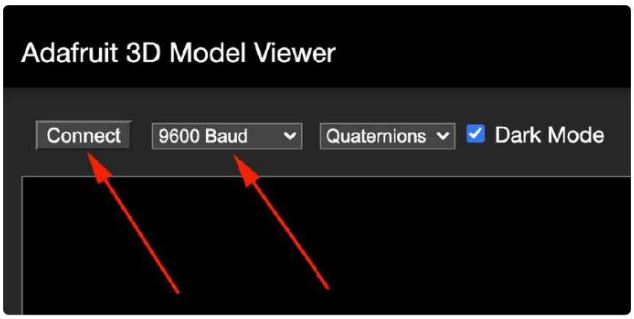
Restart Chrome

Step 5 - Visit the Adafruit 3D Model viewer

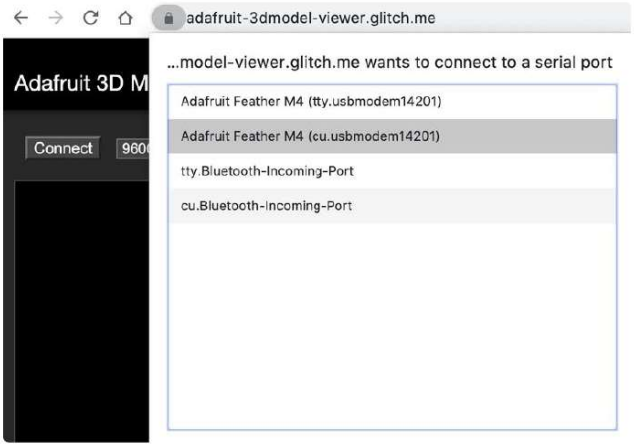
In Chrome, visit https://adafruit.github.io/Adafruit_WebSerial_3DModelViewer/ ()

Verify you have 9600 Baud selected (it only really matters for non-native-serial devices but might as well make sure its right). If you changed it in the sketch, be sure it matches.

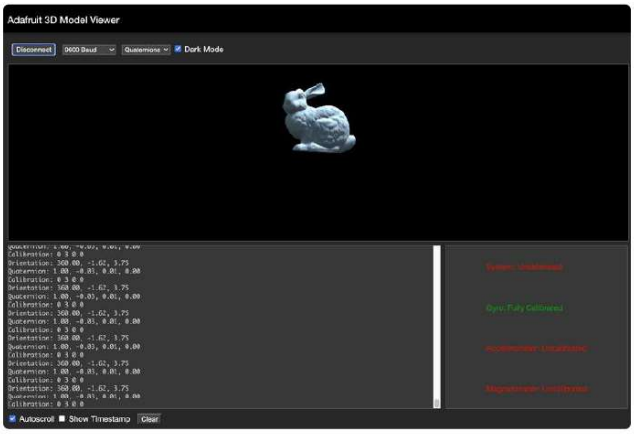
Click Connect



When the security window pops up, pick the matching Serial/COM port for your board running the AHRS sketches. Make sure the serial port isn't open in Arduino or something else.



You'll see the serial port monitor on the bottom and a 3D bunny on the top. Try rotating and twisting the sensor to see it move!



Step 6 - Calibration

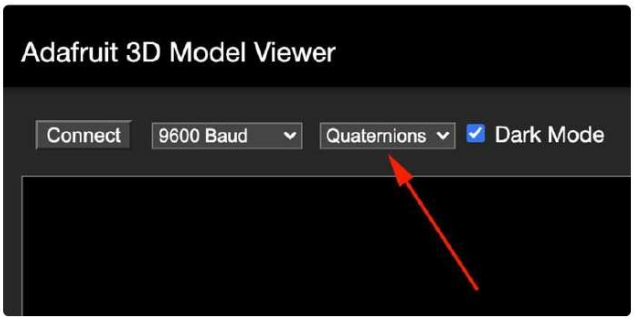
The devices will need to be calibrated each time it is powered up. You can see the Device Calibration page for more details on performing the actual calibration, but the WebSerial interface provides a convenient way to check the current calibration status.

When you first connect, you'll see that most of the calibration registers show as Uncalibrated.

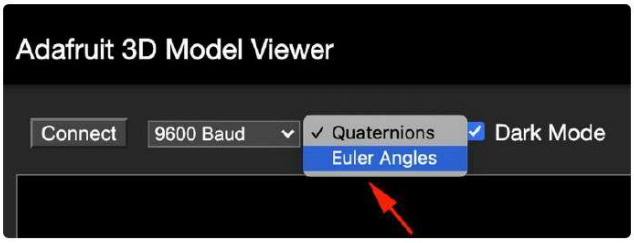
Once you have gone through the calibration steps, you will see that they are all fully calibrated.

Step 7 - Euler Angles or Quaternions

The WebSerial interface is also able to use both Euler Angles and Quaternions. Euler angles represent the X, Y, and Z axes and are easier to understand, but also have the disadvantage of "Gimbal Lock" at certain angles. To get around that, quaternions can be used. The angle type selection is at the top.



You can choose between using Euler Angles and Quaternions.

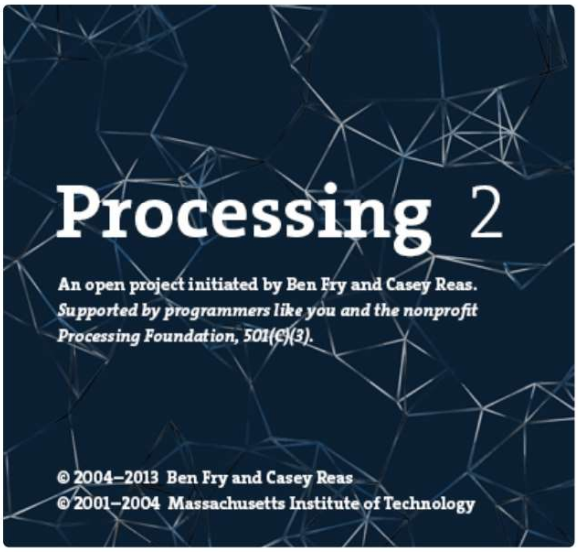


Try playing around with both by moving the bunny around and see if you can see the differences!

Processing Test

We don't recommend using processing for visualization, as its not easy. Check the previous page for how to use a Chrome browser

To help you visualize the data, we've put together a basic Processing sketch that loads a 3D model (in the .obj file format) and renders it using the data generated by the BNO055 sketch on the Uno. The "bunny" sketch on the uno published data over UART, which the Processing sketch reads in, rotating the 3D model based on the incoming orientation data.



Requirements

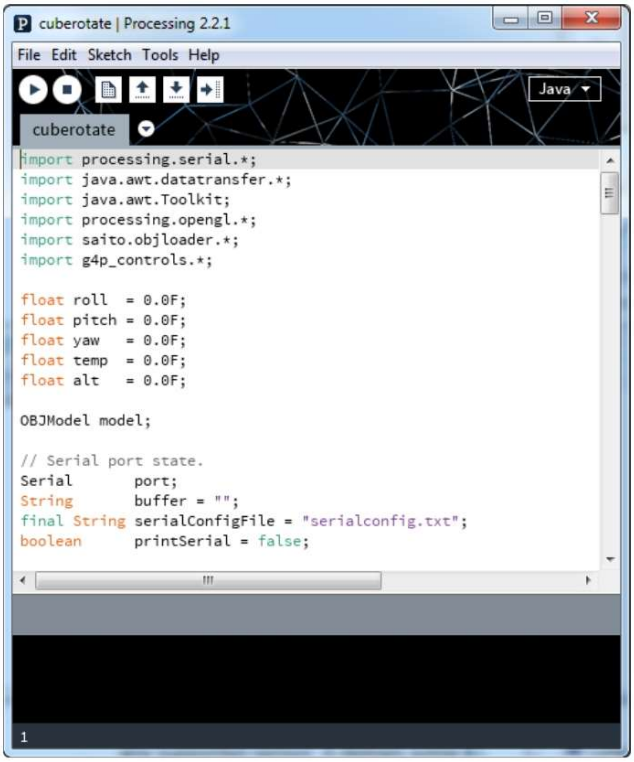
- Processing 2.x ()
  - Note that you can try later Processing versions like 3.0+ too. On some platforms Processing 2.2.1 has issues with supporting 3D acceleration (you might see 'NoClassDefFoundError: processing/awt/PGraphicsJava2D' errors). In those cases grab the later Processing 3.0+ release and use it instead of 2.x.
- Saito's OBJ Loader () library for Processing (included as part of the Adafruit repo since Google Code is now 'End of Life').
- G4P GUI library () for Processing (download the latest version here () and copy the zip into the processing libraries folder along with the OBJ loader library above). Version 3.5.2 was used in this guide.

The OBJ library is required to load 3D models. It isn't strictly necessary and you could also render a boring cube in Processing, but why play with cubes when you have rabbits?!

Opening the Processing Sketch

The processing sketch to render the 3D model is contained in the sample folder as the ahrs sketch for the Uno.

With Processing open, navigate to you Adafruit_BNO055 library folder (ex.: 'libraries/Adafruit_BNO055'), and open 'examples/bunny/processing/cuberotate/cuberotate.pde'. You should see something like this in Processing:



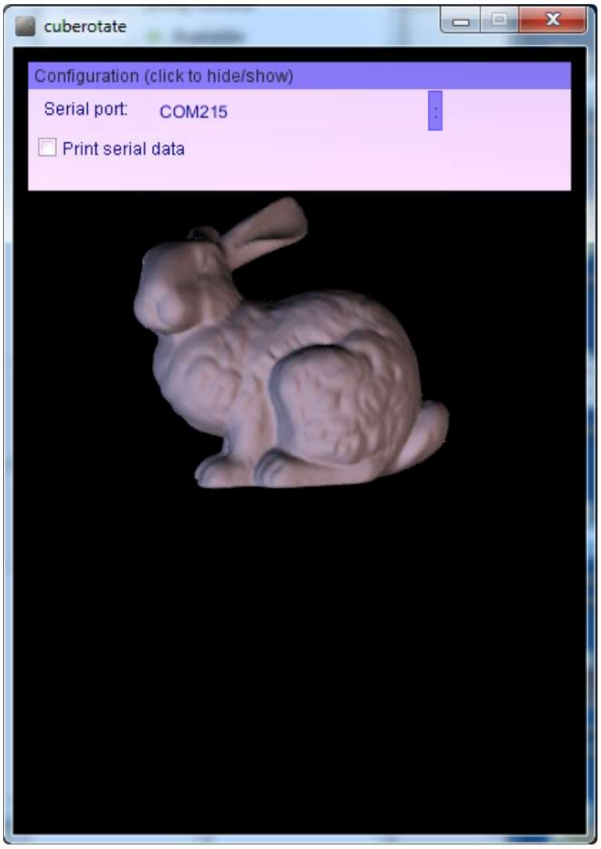
Run the Bunny Sketch on the Uno

Make sure that the "bunny" example sketch is running on the Uno, and that the Serial Monitor is closed.

With the sample sketch running on the Uno, click the triangular 'play' icon in Processing to start the sketch.

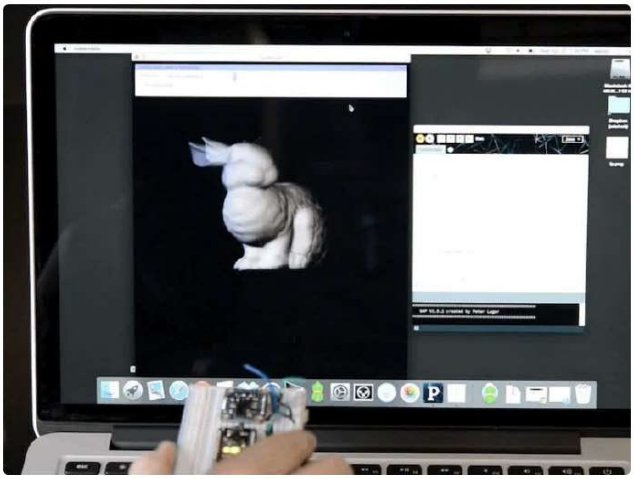
Rabbit Disco!

You should see a rabbit similar to the following image:



Before the rabbit will rotate you will need to click the : to the right of the serial port name. This will open a list of available serial ports, and you will need to click the appropriate serial port that your Arduino uses (check the Arduino IDE to see the port name if you're unsure). The chosen serial port should be remembered if you later run the sketch again.

As you rotate your breakout board, the rabbit should rotate to reflect the movement of the breakout in 3D-space, as seen in the video below



Also notice in the upper right corner of the dialog box at the top that the calibration of each sensor is displayed. It's important to calibrate the BNO055 sensor so that the most accurate readings are retrieved. Each sensor on the board has a separate calibration status from 0 (uncalibrated) up to 3 (fully calibrated). Check out the video and information from this guide for how to best calibrate the BNO055 sensor ().

Device Calibration

The BNO055 includes internal algorithms to constantly calibrate the gyroscope, accelerometer and magnetometer inside the device.

The exact nature of the calibration process is a black box and not fully documented, but you can read the calibration status of each sensor using the .getCalibration function in the Adafruit_BNO055 () library. An example showing how to use this function can be found in the sensorapi demo, though the code is also shown below for convenience sake.

The four calibration registers -- an overall system calibration status, as well individual gyroscope, magnetometer and accelerometer values -- will return a value between '0' (uncalibrated data) and '3' (fully calibrated). The higher the number the better the data will be.

```
/**************************************************************************/
/*
    Display sensor calibration status
*/
/**************************************************************************/
void displayCalStatus(void)
```

```
{
  /* Get the four calibration values (0..3) */
  /* Any sensor data reporting 0 should be ignored, */
  /* 3 means 'fully calibrated" */
  uint8_t system, gyro, accel, mag;
  system = gyro = accel = mag = 0;
  bno.getCalibration(&amp;system, &amp;gyro, &amp;accel, &amp;mag);

  /* The data should be ignored until the system calibration is &gt; 0 */
  Serial.print("\t");
  if (!system)
  {
    Serial.print("! ");
  }

  /* Display the individual values */
  Serial.print("Sys:");
  Serial.print(system, DEC);
  Serial.print(" G:");
  Serial.print(gyro, DEC);
  Serial.print(" A:");
  Serial.print(accel, DEC);
  Serial.print(" M:");
  Serial.println(mag, DEC);
}
```

Interpretting Data

The BNO055 will start supplying sensor data as soon as it is powered on. The sensors are factory trimmed to reasonably tight offsets, meaning you can get valid data even before the calibration process is complete, but particularly in NDOF mode y ou should discard data as long as the system calibration status is 0 if you have the choice.

The reason is that system cal '0' in NDOF mode means that the device has not yet found the 'north pole', and orientation values will be off The heading will jump to an absolute value once the BNO finds magnetic north (the system calibration status jumps to 1 or higher).

When running in NDOF mode, any data where the system calibration value is '0' should generally be ignored

Generating Calibration Data

To generate valid calibration data, the following criteria should be met:

- Gyroscope: The device must be standing still in any position

- Magnetometer: In the past 'figure 8' motions were required in 3 dimensions, but with recent devices fast magnetic compensation takes place with sufficient normal movement of the device
- Accelerometer: The BNO055 must be placed in 6 standing positions for +X, -X, +Y, -Y, +Z and -Z. This is the most onerous sensor to calibrate, but the best solution to generate the calibration data is to find a block of wood or similar object, and place the sensor on each of the 6 'faces' of the block, which will help to maintain sensor alignment during the calibration process. You should still be able to get reasonable quality data from the BNO055, however, even if the accelerometer isn't entirely or perfectly calibrated.

Persisting Calibration Data

Once the device is calibrated, the calibration data will be kept until the BNO is powered off.

The BNO doesn't contain any internal EEPROM, though, so you will need to perform a new calibration every time the device starts up, or manually restore previous calibration values yourself.

Bosch Video

Here's a video from the BNO055 makers on calibration!

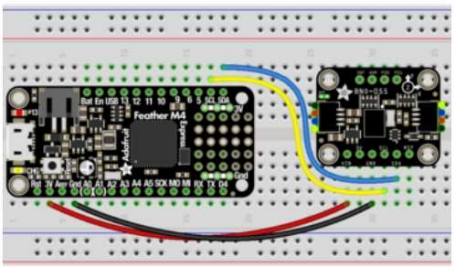
Python & CircuitPython

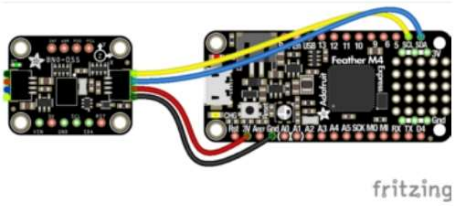
It's easy to use the BNO055 sensor with Python and CircuitPython, and the Adafruit CircuitPython BNO055 () library. This library allows you to easily write Python code that reads the acceleration and orientation of the sensor.

You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library ().

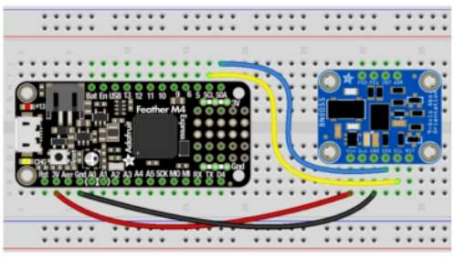
CircuitPython Microcontroller Wiring - I2C

First wire up a BNO055 to your board exactly as shown on the previous pages for Arduino using the I2C interface. Here's an example of wiring a Feather M4 to the sensor with I2C:



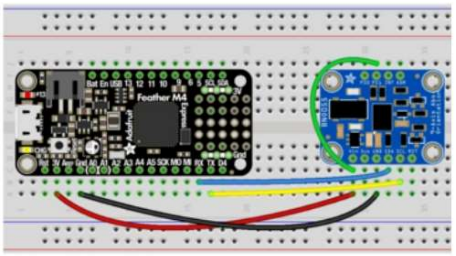


Board 3V to sensor VIN (red wire for STEMMA QT)
Board GND to sensor GND (black wire for STEMMA QT)
Board SCL to sensor SCL (yellow wire for STEMMA QT)
Board SDA to sensor SDA (blue wire for STEMMA QT)



CircuitPython Microcontroller Wiring - UART

Here's an example of wiring a Feather M4 to the sensor with UART:

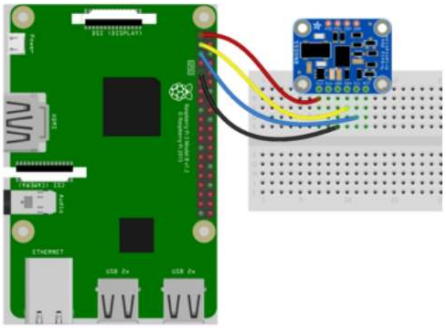


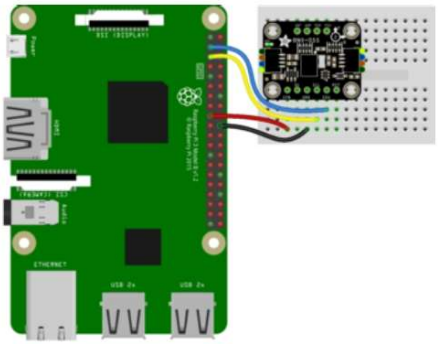
Board 3V to sensor VIN
Board GND to sensor GND
Board TX to sensor SCL
Board RX to sensor SDA
sensor PS1 to sensor VIN

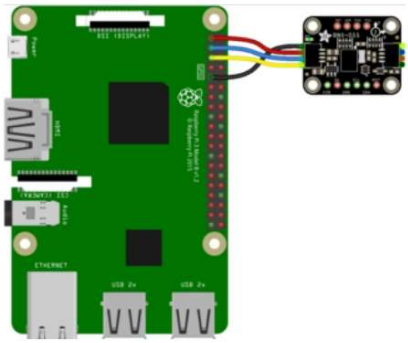
Python Computer Wiring - I2C

Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported ().

Here's the Raspberry Pi wired with I2C:





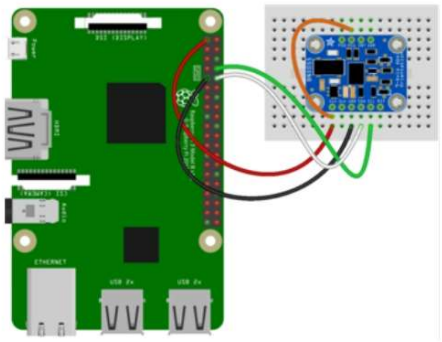


Pi 3V3 to sensor VIN (red wire for STEMMA QT)
Pi GND to sensor GND (black wire for STEMMA QT)
Pi SCL to sensor SCL (yellow wire for STEMMA QT)
Pi SDA to sensor SDA (blue wire for STEMMA QT)

Older versions of the Raspberry Pi firmware do not have I2C clock stretching support so they don't work well with the BNO. Please ensure your firmware is updated to the latest version before continuing and slow down the I2C as explained here https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/i2c-clock-stretching

Python Computer Wiring - UART

Here's the Raspberry Pi wired with UART:



Pi 3V3 to sensor VIN
Pi GND to sensor GND
Pi TXD to sensor SCL
Pi RXD to sensor SDA
sensor PS1 to sensor VIN

You will also need to configure the Pi to enable the UART and disable the login console from using it.

CircuitPython Installation of BNO055 Library

Next you'll need to install the Adafruit CircuitPython BNO055 () library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython () for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (). For example the Circuit Playground Express guide has a great page on how to install the library bundle () for express boards.

The lib folder on your CIRCUITPY drive should contain at least the following libraries:

- adafruit_bno055.mpy
- adafruit_bus_device
- adafruit_register

Before continuing make sure your board's lib folder or root filesystem has the adafruit_bno055.mpy, adafruit_bus_device, and adafruit_register files and folders copied over.

Next connect to the board's serial REPL () so you are at the CircuitPython >>> prompt.

The BNO055 CircuitPython library is large, and cannot be imported on a SAMD21 due to lack of RAM space.

Python Installation of BNO055 Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready ()!

Once that's done, from your command line run the following command:

- `sudo pip3 install adafruit-circuitpython-bno055`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

Older versions of the Raspberry Pi firmware do not have I2C clock stretching support so they don't work well with the BNO. Please ensure your firmware is updated to the latest version before continuing and slow down the I2C as explained here https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/i2c-clock-stretching

To use this sensor, you must enable i2c slowdown on the Raspberry Pi device tree overlay. Check out this guide for instructions! ()

CircuitPython & Python Usage

To demonstrate the usage of the sensor we'll initialize it and read the acceleration, orientation (in Euler angles), and more from the board's Python REPL. The difference between I2C and UART is only with the initialization. After that, you can use the sensor the same with either connection.

I2C Initialization

If you are using the I2C connection, create you sensor object as follows:

```
import board
import busio
import adafruit_bno055
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_bno055.BNO055_I2C(i2c)
```

UART Initialization - CircuitPython

If you are using the UART connection with a board (like a Feather) running CircuitPython, create your sensor object as follows:

```
import board
import busio
import adafruit_bno055
uart = busio.UART(board.TX, board.RX)
sensor = adafruit_bno055.BNO055_UART(uart)
```

UART Initialization - Python

Check how you specific board supports UART and where the port entry is created and named. For the Raspberry Pi, this is done using the pyserial module and the UART used is `/dev/serial0`. Then you create your sensor object as follows:

```
import serial
import adafruit_bno055
uart = serial.Serial("/dev/serial0")
sensor = adafruit_bno055.BNO055_UART(uart)
```

Usage

Now you're ready to read values from the sensor using any of these properties:

- temperature - The sensor temperature in degrees Celsius.
- acceleration - This is a 3-tuple of X, Y, Z axis accelerometer values in meters per second squared.
- magnetic - This is a 3-tuple of X, Y, Z axis magnetometer values in microteslas.
- gyro - This is a 3-tuple of X, Y, Z axis gyroscope values in degrees per second.
- euler - This is a 3-tuple of orientation Euler angle values.
- quaternion - This is a 4-tuple of orientation quaternion values.

- linear_acceleration - This is a 3-tuple of X, Y, Z linear acceleration values (i.e. without effect of gravity) in meters per second squared.
- gravity - This is a 3-tuple of X, Y, Z gravity acceleration values (i.e. without the effect of linear acceleration) in meters per second squared.

```
>>> print('Temperature: {} degrees C'.format(sensor.temperature))
Temperature: 23 degrees C
>>> print('Accelerometer (m/s^2): {}'.format(sensor.acceleration))
Accelerometer (m/s^2): (0.0, -0.6, 9.47)
>>> print('Magnetometer (microteslas): {}'.format(sensor.magnetic))
Magnetometer (microteslas): (-16.75, -7.1875, -42.8625)
>>> print('Gyroscope (deg/sec): {}'.format(sensor.gyro))
Gyroscope (deg/sec): (-0.00109083, 0.00218166, 0.0)
>>> print('Euler angle: {}'.format(sensor.euler))
Euler angle: (117.937, -0.25, 3.1875)
>>> print('Quaternion: {}'.format(sensor.quaternion))
Quaternion: (0.514832, -0.0124512, 0.0251465, -0.856812)
>>> print('Linear acceleration (m/s^2): {}'.format(sensor.linear_acceleration))
Linear acceleration (m/s^2): (0.01, 0.0, -0.31)
>>> print('Gravity (m/s^2): {}'.format(sensor.gravity))
Gravity (m/s^2): (-0.04, -0.54, 9.79)
>>>
```

That's all there is to using the BNO055 sensor with CircuitPython!

Here's a complete example that prints each of the properties every second. Save this as code.py on your board and look for the output in the serial REPL.

Full Example Code

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_bno055


i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
sensor = adafruit_bno055.BNO055_I2C(i2c)

# If you are going to use UART uncomment these lines
# uart = board.UART()
# sensor = adafruit_bno055.BNO055_UART(uart)

last_val = 0xFFFF


def temperature():
    global last_val  # pylint: disable=global-statement
    result = sensor.temperature
    if abs(result - last_val) == 128:
        result = sensor.temperature
        if abs(result - last_val) == 128:
            return 0b00111111 & result
    last_val = result
    return result
```

```
while True:
    print("Temperature: {} degrees C".format(sensor.temperature))
    """
    print(
        "Temperature: {} degrees C".format(temperature())
    )  # Uncomment if using a Raspberry Pi
    """
    print("Accelerometer (m/s^2): {}".format(sensor.acceleration))
    print("Magnetometer (microteslas): {}".format(sensor.magnetic))
    print("Gyroscope (rad/sec): {}".format(sensor.gyro))
    print("Euler angle: {}".format(sensor.euler))
    print("Quaternion: {}".format(sensor.quaternion))
    print("Linear acceleration (m/s^2): {}".format(sensor.linear_acceleration))
    print("Gravity (m/s^2): {}".format(sensor.gravity))
    print()

    time.sleep(1)
```

Python Docs

Python Docs ()

WebGL Example

Included with the BNO055 library is an example of how to send orientation readings to a webpage and use it to rotate a 3D model. Follow the steps below to setup and run this example.

This example is for use with Raspberry Pi and other Linux computers - flask doesn't run on CircuitPython yet!

Dependencies

In addition to the BNO055 libary, you'll need to install the flask Python web framework ().

Connect to your board in a command terminal and run the following commands:

```
sudo apt-get update
sudo apt-get install python3-flask
```

You will also need to be using a web browser that supports WebGL () on your computer or laptop. I recommend and have tested the code for this project with the latest version of Chrome ().

Download the WebGL Example

The example can be found in the library repo here (). There are various ways you can get all the code. The easiest is probably to just clone the repo to your Pi:

```
git clone https://github.com/adafruit/Adafruit_CircuitPython_BNO055.git
```

And then you can navigate to the examples folder in the repo. Or copy the example to another location, etc.

Start Server

Navigate to the webgl_demo example folder that you downloaded above. Then you can start the server by running:

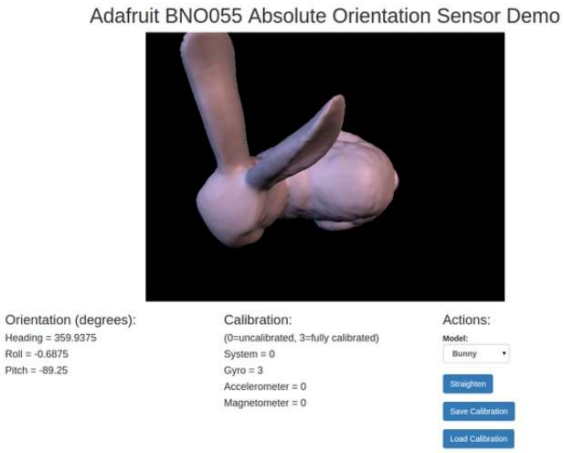
```
sudo python3 server.py
```

You should see text like the following after the server starts running:

```
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
```

Now open a web browser on your computer and navigate to your board's IP address or hostname on port 5000. For example on a Raspberry Pi http://raspberrypi.local:5000/ () might work, or on a BeagleBone Black http://beaglebone:5000/ () is the URL to try. If neither URL works you'll need to look up the IP address of your device () and then access it on port 5000. For example if your board has the IP address 192.168.1.5 you would access http://192.168.1.5:5000/ ().

Once the page loads you should see something like the following:



If you move the BNO055 sensor you should see the 3D model move too. However when the demo first runs the sensor will be uncalibrated and likely not providing good orientation data. Follow the next section to learn how to calibrate the sensor.

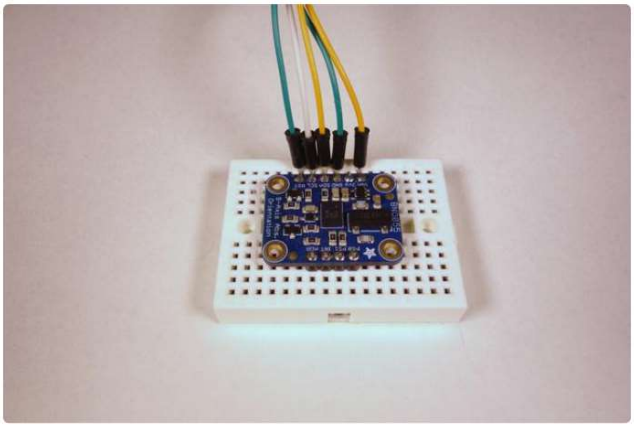
Sensor Calibration

This feature is currently not active. Once the library has been updated to allow calibration data to be saved and loaded, this section will get updated.

For now, just have fun spinning the little 3D rabbit around.

Usage

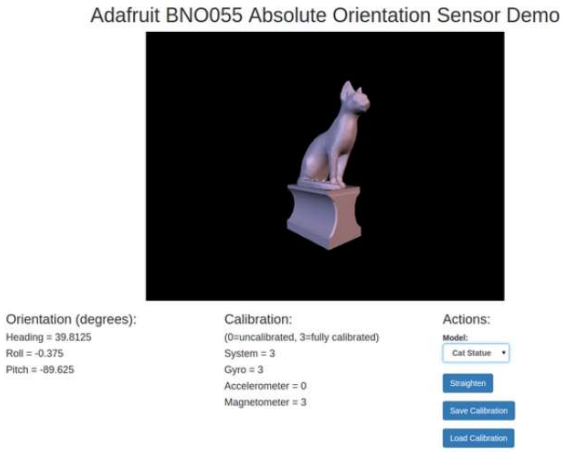
You can line up the axes of the sensor and 3D model by using the Straighten button. First you'll need to place the BNO sensor in a very specific orientation. Place the sensor flat in front of you and with the row of SDA, SCL, etc. pins facing away from you like shown below:



Then click the Straighten button and you should see the 3D model snap into its normal position:

Now move the BNO055 sensor around and you should see your movements exactly matched by the 3D model!

You can also change the 3D model by clicking the Model drop down on the right and changing to a different model, like a cat statue:



That's all there is to using the BNO055 WebGL demo!

To stop the server go back to the terminal where it was started and press Ctrl-C.

More Info

Describing how all of the WebGL code works is a little too complex for this guide, however the high level components of the example are:

- flask web service framework (): This is a great, simple web framework that is used by server.py to serve the main index.html page and expose a few web service endpoints to read BNO sensor data and save/load calibration data.
- HTML5 server sent events (): This is how data is sent from the server to the webpage. With SSE a connection is kept open and data is pushed to the client web page. BNO sensor readings are taken and sent over SSE where they're use to update the orientation of the model. This page () has a little more info on how to use HTML5 SSE with the flask framework (although it uses a more complex multiprocessing framework called gevent () that isn't necessary for simple apps like this demo).
- Three.js (): This is the JavaScript library that handles all the 3D model rendering.
- Bootstrap () & jQuery (): These are a couple other JavaScript libraries that are used for the layout and some core functionality of the page.

That's all there is to using the BNO055 WebGL demo. Enjoy using the BNO055 absolute orientation sensor in your own projects!

FAQs

Can I manually set the calibration constants?

Yes you can save and restore the calibration of the sensor, check out the restore_offsets example: https://github.com/adafruit/Adafruit_BN ... ffsets.ino ()

One thing to keep in mind though is that the sensor isn't necessarily 'plug and play' with loading the calibration data, in particular the magnetometer needs to be recalibrated even if the offsets are loaded. The magnetometer calibration is very dynamic so saving the values once might not really help when they're reloaded and the EMF around the sensor has changed.

For further details check out the datasheet and Bosch's info on the sensor for calibration info: https://www.bosch-sensortec.com/en/home ... 1/bno055_4 ()

Does the device make any assumptions about its initial orientation?

You can customize how the axes are oriented (i.e. swap them around, etc.) but the Adafruit Arduino library doesn't expose it right now. Check out section 3.4 Axis Remap of the BNO055 datasheet for info on the registers to adjust its orientation: https://www.adafruit.com/datasheets/BST ... 000_12.pdf ()

Another thing to be aware of is that until the sensor calibrates it has a relative orientation output (i.e. orientation will be relative to where the sensor was when it powered on).

A system status value of '0' in NDOF mode means that the device has not yet found the 'north pole', and orientation values will be relative not absolute. Once calibration and setup is complete (system status > '0') the heading will jump to an absolute value since the BNO has found magnetic north (the system calibration status jumps to 1 or higher). See the Device Calibration page in this learning guide for further details.

Why doesn't Euler output seem to match the Quaternion output?

The Euler angles coming out of the chip are based on 'automatic orientation detection', which has the drawback of not being continuous for all angles and situations.

According to Bosch BNO055 Euler angle output should only be used for eCompass, where pitch and roll stay below 45 degrees.

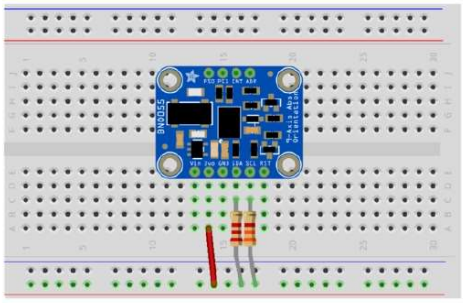
For absolute orientation, quaternions should always be used, and they can be converted to Euler angles at the last moment via the .toEuler() helper function in quaternion.h ().

Why do I get Input/Output or I2C errors when trying to use the sensor in CircuitPython?

The BNO055 I2C implementation violates the I2C protocol in some circumstances. This causes it not to work well with certain chip families. It does not work well with Espressif ESP32, ESP32-S3, and NXP i.MX RT1011, and it does not work well with I2C multiplexers. Operation with SAMD51, RP2040, STM32F4, and nRF52840 is more reliable.

I'm sometimes losing data over I2C, what can I do about this?

Depending on your system setup, you might need to adjust the pullups on the SCL and SDA lines to be a bit stronger. The BNO055 has very tight timing requirements on the I2C bus, requiring short setup and rise times on the signals. By default the breakout board has 10K pullups, which might be too weak on some setups. You can shorten the rise times and extend the setup time on the I2C lines with 'stronger' pullups. To do this simply add a 3.3K pullup on SCL and a 2.2K pullup on the SDA line with a breadboard or perma-proto board, which will override to weaker 10K pullups that are populated by default. See the image below for details:



I have some high frequency (> 2MHz) wires running near the BNO055 and I'm getting unusual results/hanging behavior

Turns out the BNO0055 breakout board is quite sensitive to RF interference from nearby wires with higher frequency square waves. ()

Try to keep high frequency lines/wires away from the BNO055!

Downloads

Files

- Arduino Library ()
- EagleCAD PCB files on GitHub ()
- BNO055 Stemma 3D models on GitHub ()
- BNO055 Datasheet ()
- Fritzing object in the Adafruit Fritzing Library ()

Pre-Compiled Bunny Rotate Binaries

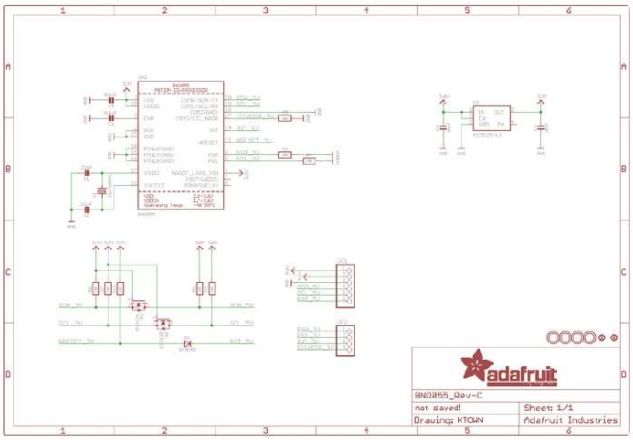
The following binary images can be used in place of running the Processing Sketch, and may help avoid the frequent API and plugin changes.

For OS X download cuberotate.app.zip, which was built on OS X 10.11.6 based on Processing 2.2.1:

cuberotate.app.zip

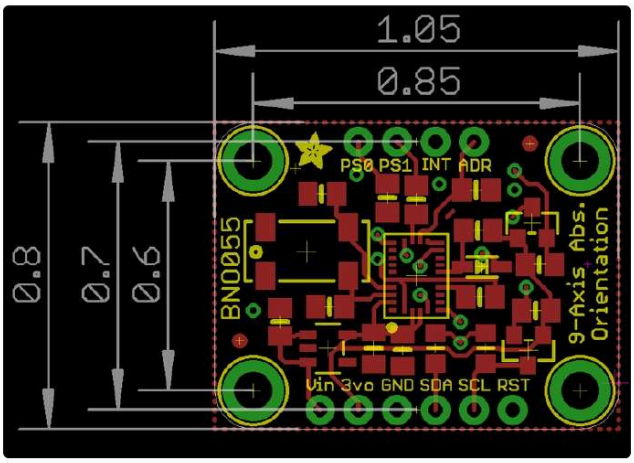
Schematic

The latest version of the Adafruit BNO055 breakout can be seen below (click the image to view the schematic in full resolution):

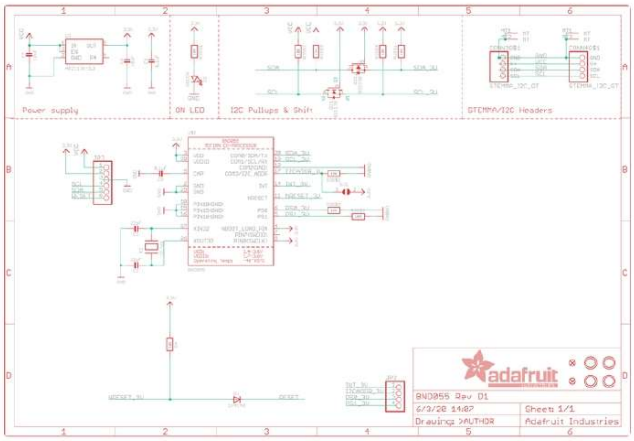


Board Dimensions

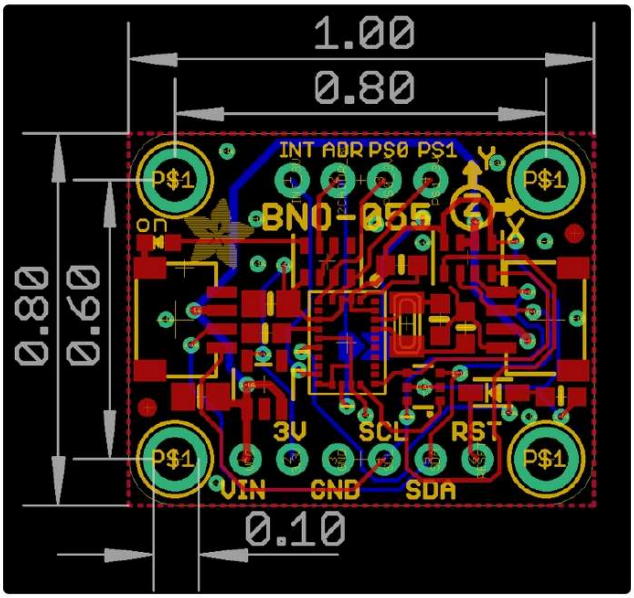
The BNO055 breakout has the following dimensions (in inches):



Schematic for STEMMA QT



Fab Print for STEMMA QT



8. L3G4200D

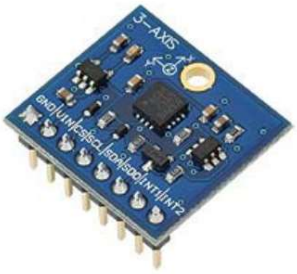
PARALLAX
Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com
Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

Gyroscope Module 3-Axis L3G4200D (#27911)

The Gyroscope Module is a low power 3-Axis angular rate sensor with temperature data. The gyroscope shows the rate of change in rotation on its X,Y and Z axes. Temperature output data and raw measured angular rate is accessed from the selectable digital interface (I²C or SPI). The module is a small package design and has an easy to access SIP interface with a mounting hole for quick connectivity to your projects. The module is designed for use with a large variety of microcontrollers with different voltage requirements.

Features

- 3-Axis angular rate sensor (yaw, pitch, and roll)
- Supports I^2C and SPI communications
- Three selectable scales: 250/500/2000 degrees/sec (dps)
- High shock survivability
- Embedded temperature sensor -40 to +185 °F (-40 to + 85 °C)
- Embedded power-down and sleep mode
- 16 bit-rate value data output
- 8-bit temperature data output



Key Specifications

- Power Requirements: 2.7 to 6.5 VDC
- Communication Interface: I^2C (up to 400 kHz) or SPI (10 MHz; 4 & 3 wire)
- Operating temperature: -40 to +185 °F (-40 to +85 °C)
- Dimensions: 0.85 X 0.80 in (2.16 X 2.03 cm)

Application Ideas

- Gaming
- 3D motion control
- Virtual reality input devices
- Robotics
- UAV, IMU systems

Downloads & Resources

More resources, example code, the device schematic, and open-source hardware files are available from the 27911 product page at www.parallax.com.

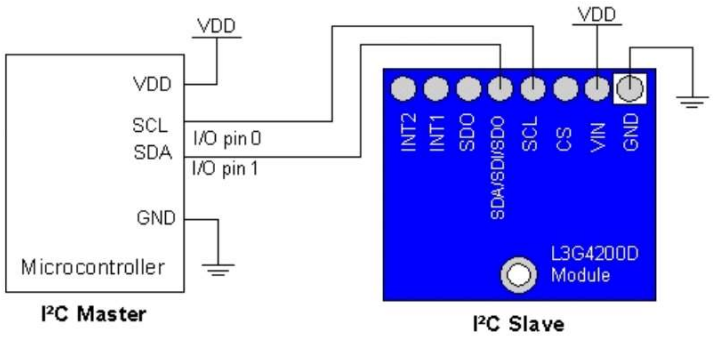
Quick-Start Guide

The following is a very basic procedure to get started initializing and reading values from the Gyroscope Module. Example test code for the Propeller and Basic Stamp can be found on the 27911 product page at www.parallax.com.

This module's default communication setup is I²C. Use of SPI communication is configured by pulling the CS line low. See the datasheet for 3-wire and 4-wire SPI configuration and use.

1. With main power off, make the proper connections between the module and a microcontroller as shown below. The voltage connected to the VIN pin should be the same as the voltage powering the microcontroller communicating with the device.

2. Power on the device and load the BASIC Stamp or Propeller sample code provided on the 27911 product page.



NOTE: The SDA pin is connected to a bi-directional level shifting IC, used to translate the lower I/O voltage of the L3G4200D to the possibly higher voltage used by an externally connected device. This particular level shifter has built-in pull-up resistors to each supply rail so it can be driven by open drain outputs for I^2C; but can also be driven low/high for SPI protocol. The CLK and CS pins operate exclusively as inputs, so a simpler level shifting circuit with a resistor and Schottky diode is used. For the SDO pin another simple level shifting circuit is used along with a MOSFET and two resistors.

In summary, there is no need for any external hardware to operate.

Calibration

Each L3G4200D is factory tested and trimmed for zero-rate level and sensitivity. So, for most common applications, no further calibration is required. For details on the calibration, please reference page 31 of Gyro_app_Note1.pdf on the product page.

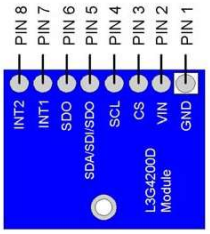
Device Information

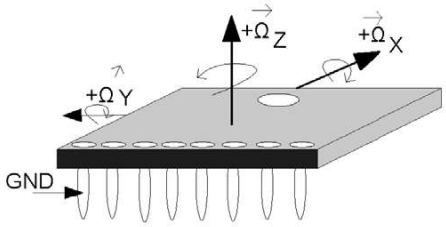
Complete device information for the L3G4200D can be found in the manufacturer's datasheet, which is available for download from the 27911 product page at www.parallax.com

Pin Definitions and Ratings

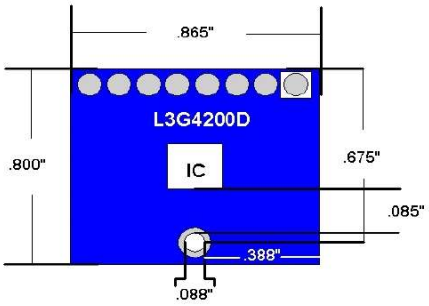
Pin	Name	Type	Function
1	GND	G	0V Supply, Ground Pin
2	VIN	P	Supply Voltage from +2.7 – +6.5VDC
3	CS	I	SPI enable (Default is I²C enabled) I²C/SPI mode selection (1: I²C communication enabled; 0: SPI communication mode / I2C disabled)
4	SCL	I	I²C & SPI serial clock (SCL)
5	SDA/SDI/SDO	IO	I²C serial data (SDA) SPI serial data input (SDI) 3-wire interface serial data output (SDO)
6	SDO	O	SPI serial data output (SDO) I²C least significant bit of the device address (SA0)
7	INT1	I	Programmable interrupt, see datasheet for more details
8	INT2	I	Data ready/FIFO interrupt, see datasheet for more details

Pin Type: P = Power, G = Ground, I = Input, O = Output





Module Dimensions



L3G4200D Mechanical Characteristics

Characterized @ Vdd = 3.0 V, T = 25 °C

Symbol	Parameter	Test condition	Typical	Units
FS	Measurement range	User-selectable	±250	dps
			±500	
			±2000	
So	Sensitivity	FS = 250 dps	8.75	Mdps/digit
		FS = 500 dps	17.50	
		FS = 2000 dps	70	
SoDr	Sensitivity change vs. temperature	From -40 °C to +85 °C	±2	%
DVoff	Digital zero-rate level	FS = 250 dps	±10	dps
		FS = 500 dps	±15	
		FS = 2000 dps	±75	
OffDr	Zero-rate level change vs. temperature	FS = 250 dps	±0.03	dps/°C
		FS = 500 dps	±0.04	
NL	Non linearity	Best fit straight line	0.2	% FS
DST	Self-test output change	FS = 250 dps	130	dps
		FS = 500 dps	200	
		FS = 2000 dps	530	
Rn	Rate noise density	BW = 50 Hz	0.03	dps/sqrt(Hz)
ODR	Digital output data rate		100/200/ 400/800	Hz

Data excerpt from the L3G4200D datasheet.

Communication Protocol

Note: details on the communication protocol and FIFO modes are taken from the L3G4200D datasheet. Please reference the datasheet for much more detailed explanations and configurations.

Communication Settings

You can select between I²C (2-wire) or SPI (3 or 4 wire) communication protocols; I²C is the default setting for this module. These serial interfaces are mapped onto the same pins.

I²C

The Gyroscope module I^2C is a bus slave. I²C communication is used to read and write to and from the Gyroscope's data registers.

The two signals need for I²C operation are the serial clock line (SCL) and the serial data line (SDA). The SDA line is bidirectional and used for sending and receiving the data to/from the interface.

SPI

The SPI is a bus slave. The SPI communication is used to read and write to and from the Gyroscope's data registers. The serial interface interacts with the external world through 4 wires: CS, SCL, SDI, and SDO (see Pin Definitions and Ratings for descriptions).

FIFO Modes

FIFO is an acronym for First In, First Out. It used to buffer data to help with flow of communication to devices.

There are 32 slots of FIFO data, for each of the three output channels: yaw, pitch, and roll (X,Y,Z). Each slot has 16 bits of data.

The great thing about having a FIFO is the host processor does not need to continuously poll data from the sensor. Instead, it can wake up only when needed and burst the significant data out from the FIFO. This buffer can work in five different modes.

There are five FIFO mode settings available; the default mode is "Bypass mode." To see how to access different modes, see FIFO_CTRL_REG and FIFO_SRC_REG in the datasheet posted to the 27911 product page at www.parallax.com.

Bypass Mode
In bypass mode, the FIFO is not operational and for this reason it remains empty.

FIFO Mode
In FIFO mode, data from the yaw, pitch, and roll channels are stored in the FIFO.

Stream Mode
In stream mode, data from yaw, pitch, and roll measurements are stored in the FIFO. The FIFO continues filling until full (32 slots of 16-bit data for yaw, pitch, and roll). When full, the FIFO discards the older data as the new data arrives.

Bypass-to-stream Mode
In bypass-to-stream mode, the FIFO starts operating in bypass mode, and once a trigger event occurs, the FIFO starts operating in stream mode.

Stream-to-FIFO Mode
In stream-to-FIFO mode, data from yaw, pitch, and roll measurements are stored in the
FIFO. An interrupt can be enabled on pin INT2, setting the I2_WTM bit in CTRL_REG3, which is triggered when the FIFO is filled to the level specified in the WTM. The FIFO continues filling until full (32 slots of 16-bit data for yaw, pitch, and roll).

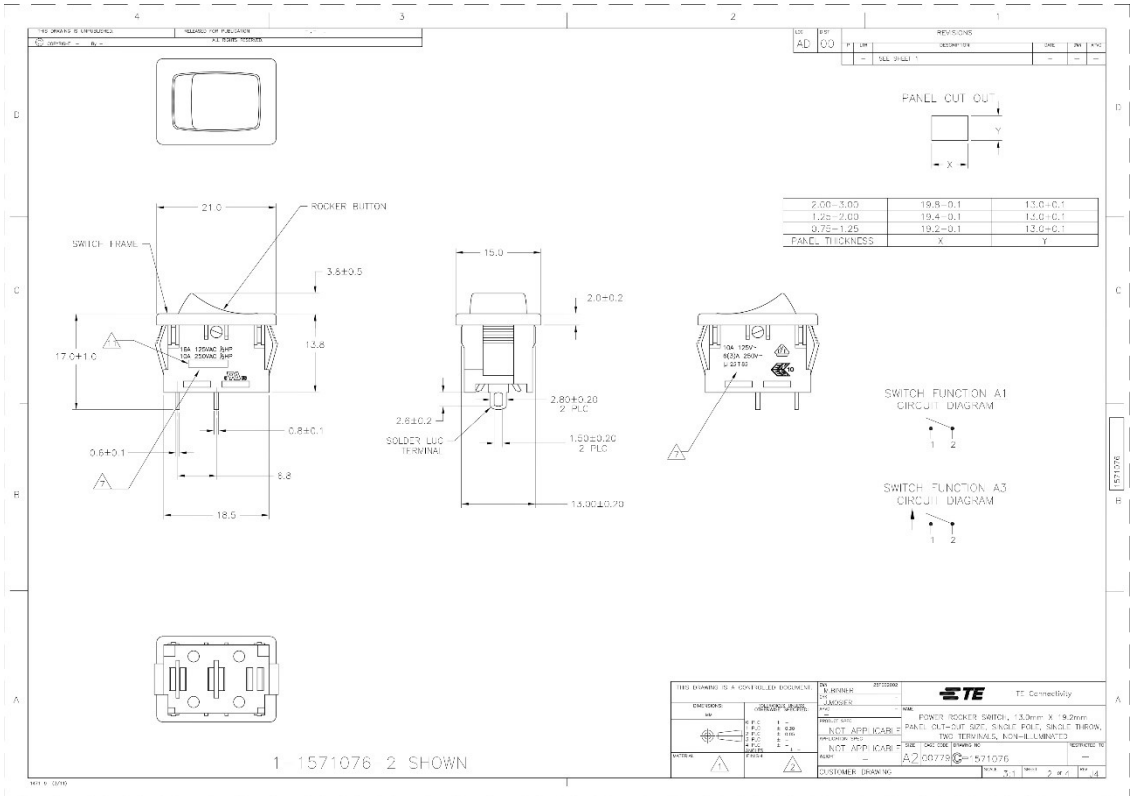
Example Code

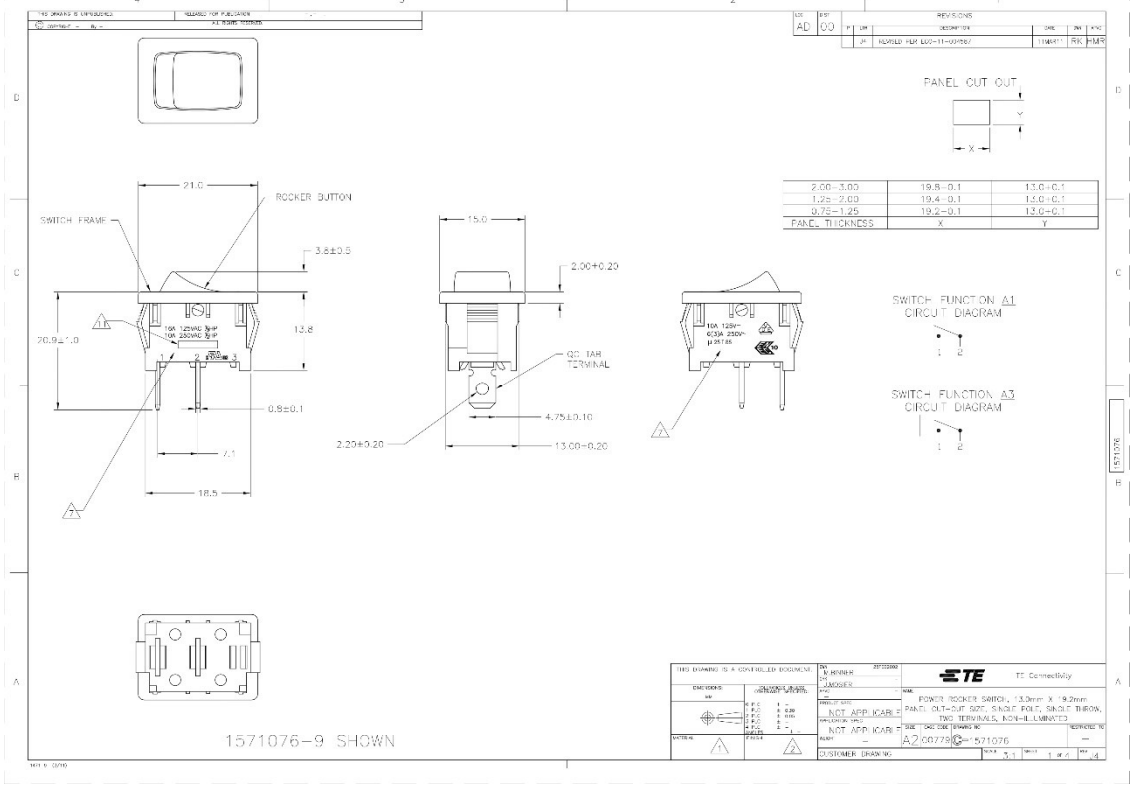
Example code is available for download from the 27911 product page at www.parallax.com

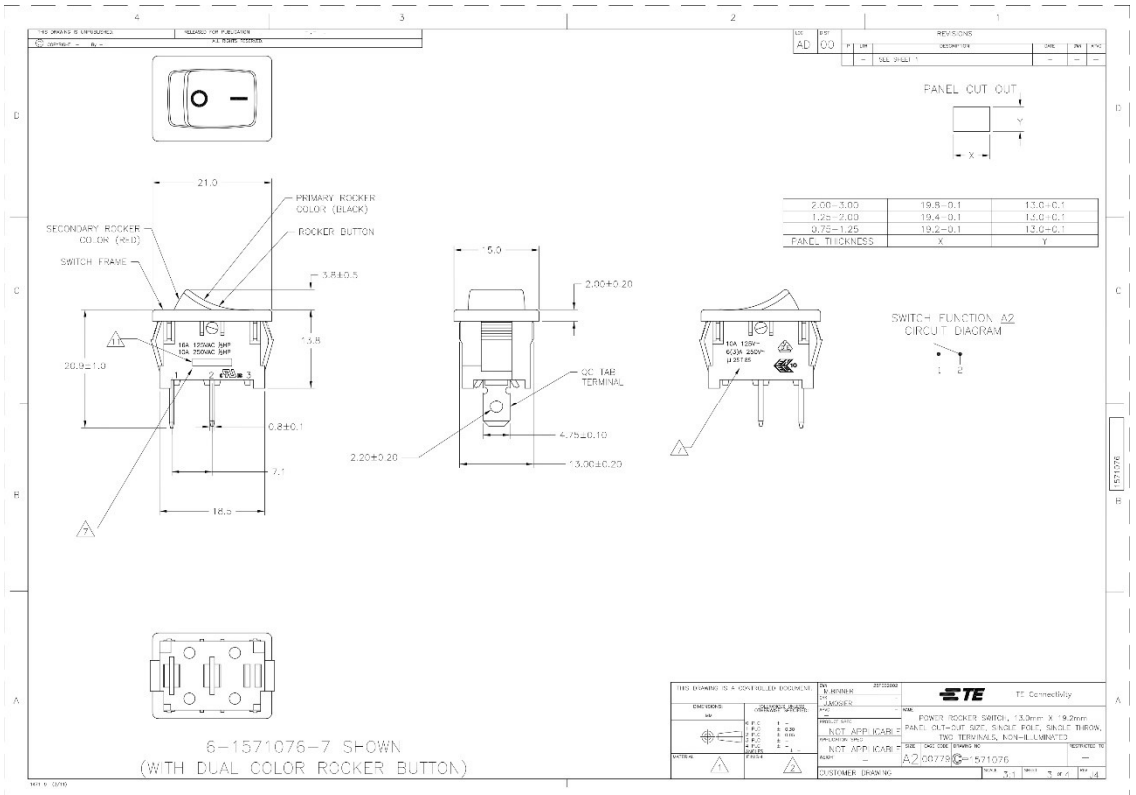
BASIC Stamp 2
The L3G4200D_Gyroscope_Demo.bs2 program reads raw X,Y,Z values from the Gyroscope module using the default I²C interface, and displays the values in the BASIC Stamp Editor's Debug Terminal. The software is a free download from www.parallax.com/basicstampsoftware.

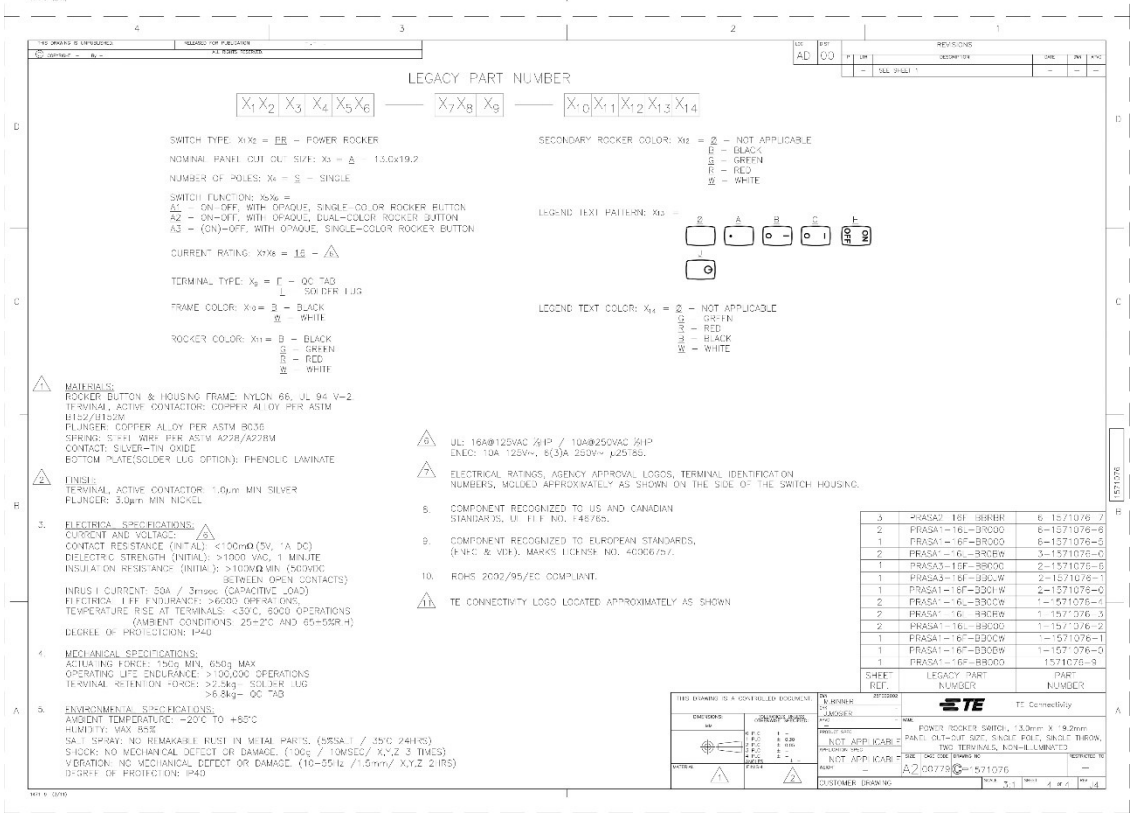
Propeller™ P8X32A
The L3G4200D_example_code.spin program reads raw X,Y,Z values from the Gyroscope module using the default I²C interface, and displays the values in a serial terminal. It calls FullDuplexSerial.spin, a library object of the Propeller Tool software, which is available from www.parallax.com/Propeller.

9. Interruptor PRASA1-16F-BB0BW





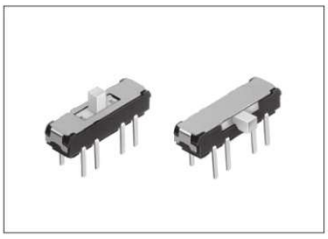

PANEL CUT OUT
21.0
PRIMARY ROCKER COLOR (BLACK)
SECONDARY ROCKER COLOR (RED)
ROCKER BUTTON
SWITCH FRAME
3.8±0.5
20.9±1.0
15.8
0.8±0.1
7.1
16.0
5.0
2.00±0.20
QC TAB TERMINAL
4.75±0.10
2.20±0.20
13.00±0.20
SWITCH FUNCTION: A2
CIRCUIT DIAGRAM
2.00~3.00 | 19.8~0.1 | 13.0+0.1
1.25~2.00 | 19.4~0.1 | 13.0+0.1
0.75~1.25 | 19.2~0.1 | 13.0+0.1
PANEL THICKNESS | X | Y
6-1571076-7 SHOWN
(WITH DUAL COLOR ROCKER BUTTON)
TE Connectivity
POWER ROCKER SWITCH, 13.0mm X 19.2mm
PANEL CUT-OUT SIZE, SINGLE POLE, SINGLE THROW,
TWO TERMINALS, NON-ILLUMINATED
A2 (0079) - 1571076


LEGACY PART NUMBER
X1X2 X3 X4 X5X6 — X7X8 X9 — X10 X11 X12 X13 X14
SWITCH TYPE: X1X2 = PR – POWER ROCKER
NOMINAL PANEL CUT OUT SIZE: X3 = A – 13.0x19.2
NUMBER OF POLES: X4 = S – SINGLE
SWITCH FUNCTION: X5X6 =
A1 – ON-OFF, WITH OPAQUE, SINGLE-COLOR ROCKER BUTTON
A2 – ON-OFF, WITH OPAQUE, DUAL-COLOR ROCKER BUTTON
A3 – (ON)-OFF, WITH OPAQUE, SINGLE-COLOR ROCKER BUTTON
CURRENT RATING: X7X8 = 16 – 5
TERMINAL TYPE: X9 = Q – QC TAB
L – SOLDER LUG
FRAME COLOR: X10 = B – BLACK
W – WHITE
ROCKER COLOR: X11 = B – BLACK
G – GREEN
R – RED
W – WHITE
SECONDARY ROCKER COLOR: X12 = N – NOT APPLICABLE
B – BLACK
G – GREEN
R – RED
W – WHITE
LEGEND TEXT PATTERN: X13 =
2 A B C F
1
LEGEND TEXT COLOR: X14 = N – NOT APPLICABLE
G – GREEN
R – RED
B – BLACK
W – WHITE
1 MATERIALS:
ROCKER BUTTON & HOUSING FRAME: NYLON 66, UL 94 V-2
TERMINAL, ACTIVE CONTACTOR: COPPER ALLOY PER ASTM B152/B152M
PLUNGER: COPPER ALLOY PER ASTM B036
SPRING: STEEL WIRE PER ASTM A228/A228M
CONTACT: SILVER-TIN OXIDE
BOTTOM PLATE(SOLDER LUG OPTION): PHENOLIC LAMINATE
2 FINISH:
TERMINAL, ACTIVE CONTACTOR: 1.0µm MIN SILVER
PLUNGER: 3.0µm MIN NICKEL
3. ELECTRICAL SPECIFICATIONS:
CURRENT AND VOLTAGE: – 6
CONTACT RESISTANCE (INITIAL): <100mΩ (5V, 1A DC)
DIELECTRIC STRENGTH (INITIAL): >1000 VAC, 1 MINUTE
INSULATION RESISTANCE (INITIAL): >100MΩ MIN. (500VDC BETWEEN OPEN CONTACTS)
INRUSH CURRENT: 30A / 3msec (CAPACITIVE LOAD)
ELECTRICAL LIFE ENDURANCE: >6000 OPERATIONS
TEMPERATURE RISE AT TERMINALS: <30°C, 6000 OPERATIONS (AMBIENT CONDITIONS: 25±2°C AND 65±5%R.H)
DEGREE OF PROTECTION: IP40
4. MECHANICAL SPECIFICATIONS:
ACTUATING FORCE: 150g MIN, 650g MAX
OPERATING LIFE ENDURANCE: >100,000 OPERATIONS
TERMINAL RETENTION FORCE: >2.5kg SOLDER LUG >6.8kg QC TAB
5. ENVIRONMENTAL SPECIFICATIONS:
AMBIENT TEMPERATURE: -20°C TO +85°C
HUMIDITY: MAX 85%
SALT SPRAY: NO REMARKABLE RUST IN METAL PARTS. (5%SALT / 35°C 24HRS)
SHOCK: NO MECHANICAL DEFECT OR DAMAGE. (100g / 10MSEC/ X,Y,Z 3 TIMES)
VIBRATION: NO MECHANICAL DEFECT OR DAMAGE. (10~55Hz /1.5mm/ X,Y,Z 2HRS)
DEGREE OF PROTECTION: IP40
8 UL: 16A@125VAC ½HP / 10A@250VAC ½HP
ENEC: 10A 125V~, 6(3)A 250V~ µ25T85.
7 ELECTRICAL RATINGS, AGENCY APPROVAL LOGOS, TERMINAL IDENTIFICATION NUMBERS, MOLDED APPROXIMATELY AS SHOWN ON THE SIDE OF THE SWITCH HOUSING.
6. COMPONENT RECOGNIZED TO US AND CANADIAN STANDARDS, UL FILE NO. E48765.
9. COMPONENT RECOGNIZED TO EUROPEAN STANDARDS, (ENEC & VDE). MARKS LICENSE NO. 40006757.
10. ROHS 2002/95/EC COMPLIANT.
11 TE CONNECTIVITY LOGO LOCATED APPROXIMATELY AS SHOWN
3 PRASA2-16F-BBBNR 6-1571076-7
2 PRASA1-16F-BRB00 6-1571076-6
1 PRASA1-16F-BRB00 6-1571076-5
2 PRASA1-16F-BRBBW 3-1571076-0
1 PRASA3-16F-BBB00 2-1571076-6
1 PRASA3-16F-BBB0W 2-1571076-1
1 PRASA1-16F-BBB0W 2-1571076-0
2 PRASA1-16F-BBB0W 1-1571076-4
2 PRASA1-16F-BRBBW 1-1571076-3
2 PRASA1-16F-BRB00 1-1571076-2
1 PRASA1-16F-BBB0W 1-1571076-1
1 PRASA1-16F-BBB0W 1-1571076-0
1 PRASA1-16F-BRB00 1571076-9
SHEET REF. | LEGACY PART NUMBER | PART NUMBER
TE Connectivity
POWER ROCKER SWITCH, 13.0mm X 19.2mm
PANEL CUT-OUT SIZE, SINGLE POLE, SINGLE THROW,
TWO TERMINALS, NON-ILLUMINATED
A2 (0079) - 1571076

10. Interruptor Micro

SSSS2 3.5(H)mm, 2mm-travel Type

Excellent application for designing compact and high density portable devices



Typical Specifications

Items		Specifications
Rating (max.)/(min.) (Resistive load)		0.3A 6V DC / 50μA 3V DC
Contact resistance (Initial performance / After lifetime)		70mΩ max. / 130mΩ max.
Operating force		Refer to the dimensions.
Operating life	Without load	10,000 cycles※
	With load	10,000 cycles (0.3A 6V DC)※

Note ※ Operating life for SSSS213202 is 100 cycles

Product Line

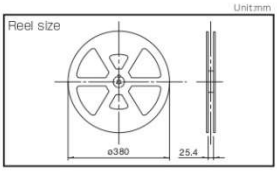
Travel (mm)	Actuator direction	Actuator length (mm)	Poles	Positions	Changeover timing	Soldering	Minimum order unit (pcs.) Japan	Minimum order unit (pcs.) Export	Products No.	Drawing No.
2	Vertical	2	1	2	Non shorting	Manual, Dip	100	10,000	SSSS213000	1
				3					SSSS211900	2
			2	2					SSSS222700	3
				3					SSSS223600	4
			1	2		Reflow	12,00	4,800	SSSS213202	5
				3			1,000	4,000	SSSS212901	6
	Horizontal			2		Manual, Dip	100	10,000	SSSS213100	7
				3					SSSS212200	8
				4					SSSS212400	9
			2	2					SSSS223200	10
				3					SSSS223900	11
				4					SSSS224100	12
			1	2		Reflow	1,400	5,600	SSSS211603	13
				3					SSSS212301	14
			2						SSSS224001	15

Packing Specifications

Taping

Product No.	Number of packages (pcs.) 1 reel	1 case /Japan	1 case /export packing	Tape width (mm)	Export package measurements (mm)
SSSS213202	1,200	2,400	4,800	24	428×413×172
SSSS212901	1,000	2,000	4,000		406×406×190
SSSS211603 SSSS212301 SSSS224001	1,400	2,800	5,600		

Unit:mm



Bulk

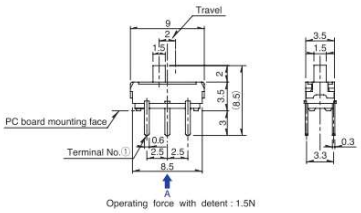
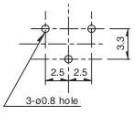
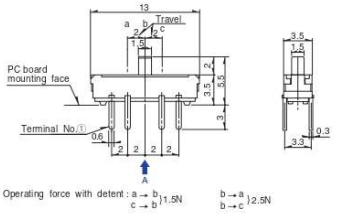
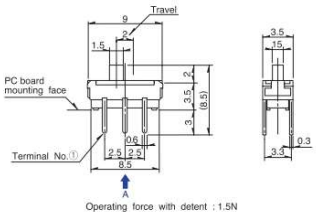
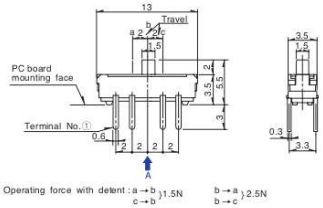
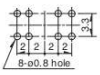
Product No.	Number of packages (pcs.) 1 case /Japan	1 case /export packing	Export package measurements (mm)
SSSS211900, SSSS212200, SSSS212400, SSSS213000, SSSS213100, SSSS222700, SSSS223200, SSSS223600, SSSS223900, SSSS224100	2,000	10,000	400×270×290

Refer to P.111 for soldering conditions.

Dimensions

Vertical Actuator Type

Unit:mm

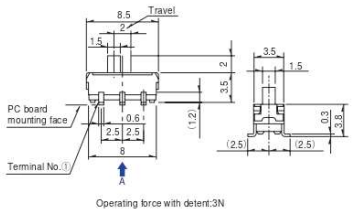
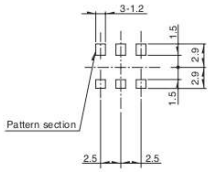
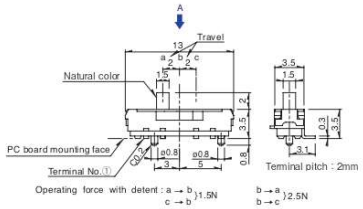
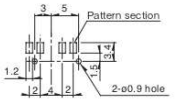
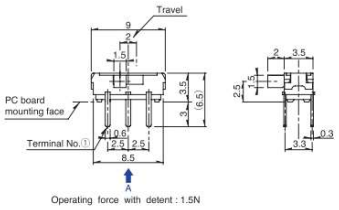
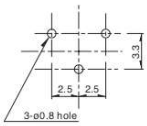
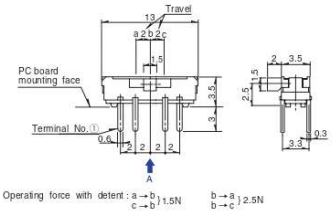
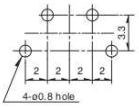
No.	Photo	Style	PC board mounting hole dimensions (Viewed from direction A)
1	**1-pole, 2-position** 		
2	**1-pole, 3-position** 		
3	**2-pole, 2-position** 		
4	**2-pole, 3-position** 		

SSSS2 3.5(H)mm, 2mm-travel Type

Dimensions

Vertical Actuator Type/Horizontal Actuator Type

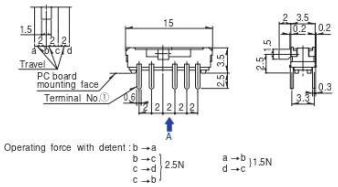
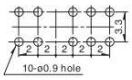
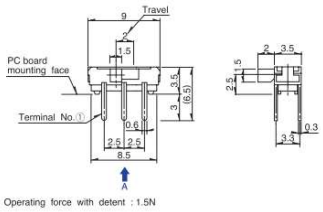
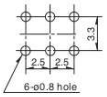
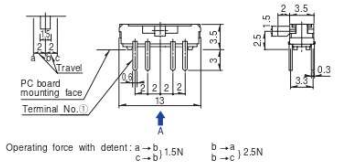
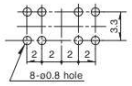
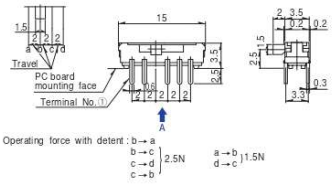
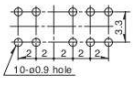
Unit:mm

No.	Photo	Style	PC board mounting hole dimensions (Viewed from direction A)
5	**1-pole, 2-position Reflow** 		
6	**1-pole, 3-position Reflow** 		
7	**1-pole, 2-position** 		
8	**1-pole, 3-position** 		

■Dimensions

Horizontal Actuator Type

Unit:mm

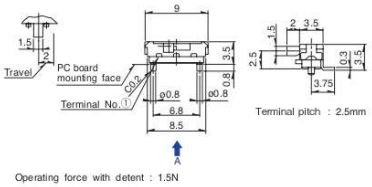
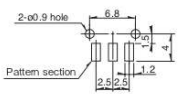
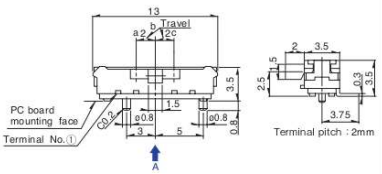
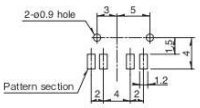
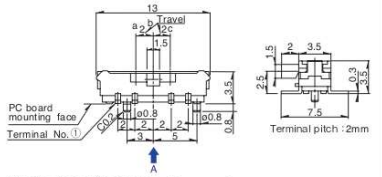
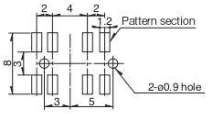
No.	Photo	Style	PC board mounting hole dimensions (Viewed from direction A)
9	**1-pole, 4-position** 		
10	**2-pole, 2-position** 		
11	**2-pole, 3-position** 		
12	**2-pole, 4-position** 		

SSSS2 3.5(H)mm, 2mm-travel Type

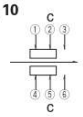
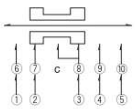
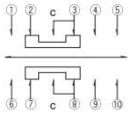
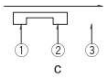
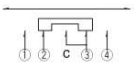
Dimensions

Horizontal Actuator Type/Reflow Type

Unit:mm

No.	Photo	Style	PC board mounting hole and land dimensions (Viewed from direction A)
13	**1-pole, 2-position** 	 Operating force with detent : 1.5N	
14	**1-pole, 3-position** 	 Operating force with detent : a→b, c→b 1.5N; b→a, b→c 2.5N	
15	**2-pole, 3-position** 	 Operating force with detent : a→b, c→b 1.5N; b→a, b→c 2.5N	

Circuit Diagram (Viewed from Direction A)

1-pole, 2-position **Drawing No.1, 5, 7** 	**1-pole, 3-position** **Drawing No.2, 6, 8** 	**2-pole, 2-position** **Drawing No.3, 10** 
2-pole, 3-position **Drawing No.4, 11, 15** 	**1-pole, 4-position** **Drawing No.9** 	**2-pole, 4-position** **Drawing No.12** 
1-pole, 2-position **Drawing No.13** 	**1-pole, 3-position** **Drawing No.14** 	

Slide Switches
List of Varieties

Series		SSSS2※1	SSSS9	SSAC	SSSF※2	SSSU※2
Photo						
Actuator direction	Horizontal	●	●	●	●	●
	Vertical	●	●	—	●	●
Poles-positions	1-2	●	●	—	●	●
	1-3	●	●	—	●	●
	1-4	●	—	—	—	—
	2-2	●	●	●	●	●
	2-3	●	●	●	●	●
	2-4	●	—	—	—	—
	4-2	—	—	—	●	●
Travel (mm)		2	2	1.5	2	3
Operating temperature range		−40℃ to +85℃		−10℃ to +60℃	−40℃ to +85℃	
Automotive use		—	—	—	—	—
Life cycle		★3	★3	★3	★3	★3
Rating (max.) (Resistive load)		0.3A 6V DC	0.1A 12V DC	1mA 5V DC	0.1A 30V DC	
Rating (min.) (Resistive load)		50μA 3V DC	1mA 5V DC	50μA 3V DC	10μA 1V DC	
Durability	Operating life without load	10,000 cycles 100mΩ max.※1	10,000 cycles 60mΩ max.	10,000 cycles 200mΩ max.	10,000 cycles 45mΩ max.	
	Operating life with load Load: as rating	10,000 cycles 130mΩ max.※1	10,000 cycles 80mΩ max.		10,000 cycles 65mΩ max.	
Electrical performance	Initial contact resistance	70mΩ max.	30mΩ max.	100mΩ max.	25mΩ max.	
	Insulation resistance	100MΩ min. 500V DC		100MΩ min. 100V DC	100MΩ min. 500V DC	
	Voltage proof	500V AC for 1minute		100V AC for 1minute	500V AC for 1minute	
Mechanical performance	Terminal strength	3N for 1minute		5N for 1minute		
	Actuator strength: Operating direction	20N	30N	5N	30N	
	Actuator strength: Pulling direction	10N				
Environmental performance	Cold	−20℃ 500h	−40℃ 500h	−20℃ 96h	−40℃ 500h	
	Dry heat	85℃ 500h		85℃ 96h	85℃ 500h	
	Damp heat	60℃, 90 to 95%RH 500h		40℃, 90 to 95%RH 96h	60℃, 90 to 95%RH 500h	
Page		90	95	101	103	107

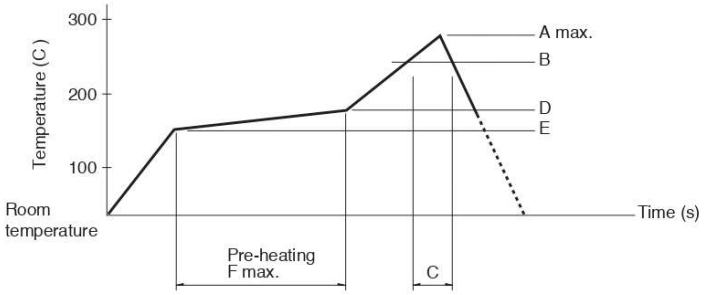
Note

1. ※1. Operating life for SSSS213202 is 100 cycles.
2. ※2. The operating temperature range for automotive applications can be raised upon request. Please contact us for details.
3. ● Indicates applicability to all products in the series.

Slide Switches Soldering Conditions

Example of Reflow Soldering Condition

1. Heating method: Double heating method with infrared heater.
2. Temperature measurement: Thermocouple ϕ 0.1 to 0.2 CA (K) or CC (T) at soldering portion (copper foil surface).
   A heat resisting tape should be used for fixed measurement.
3. Temperature profile



Series (Reflow type)			A (℃) 3s max.	B (℃)	C (s)	D (℃)	E (℃)	F (s)
SSSS2	Vertical	1-pole, 3-position	260	230	40	180	150	120
	Horizontal	1-pole, 2-position 1-pole, 3-position 2-pole, 3-position						
	Vertical	1-pole, 2-position	250					
SSSS7								
SSAH, SSAG, SSAJ, SSAL, SSSS8			260					

Notes

1. The condition mentioned above is the temperature on the mounting surface of a PC board. There are cases where the PC board's temperature greatly differs from that of the switch, depending on the PC board's material, size, thickness, etc. The above-stated conditions shall also apply to switch surface temperatures.
2. Soldering conditions differ depending on reflow soldering machines. Prior verification of soldering condition is highly recommended.

Reference for Hand Soldering

Series	Soldering temperature	Soldering time
SSSF, SSSU	350±10℃	3+1/0s
SSSS2	350±10℃	4s max.
SSSS9	350±10℃	3s max.
SSAH, SSAG, SSAJ, SSAL	350±5℃	3s max.
SSSS8	330±5℃	3s max.
SSSS7	320±5℃	3s max.
SSAC	300±10℃	2s max.

Reference for Dip Soldering

(For PC board terminal types)

Series	Items		Dip soldering	
	Preheating temperature	Preheating time	Soldering temperature	Duration of immersion
SSSS2	100℃ max.	60s max.	260±5℃	3±1s
SSSS9	120℃ max.	60s max.	260±5℃	5+0/-1s (2 times)
SSSF, SSSU	100℃ max.	60s max.	260±5℃	10±1s/5±1s
SSAC	100℃ max.	60s max.	260±5℃	5±1s

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje e implementación del control de altura de

un cuadricóptero

Documento Nº2: PLANOS

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Fontalba Roca, Marc

Tutor/a: González Sorribes, Antonio

CURSO ACADÉMICO: 2022 / 2023

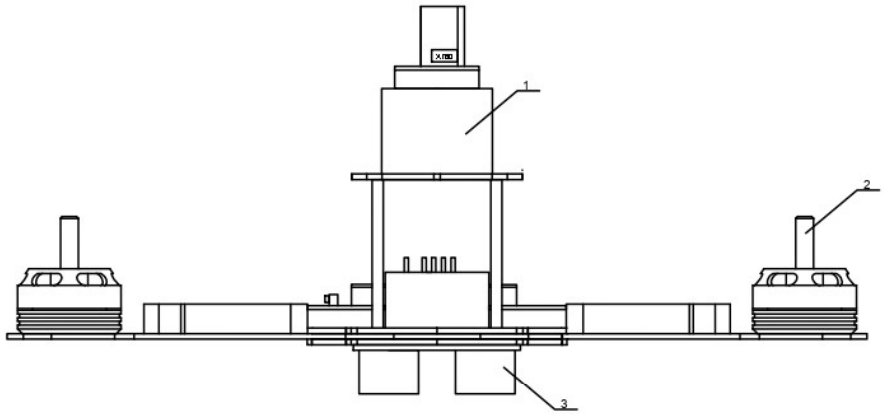
Índice de planos:

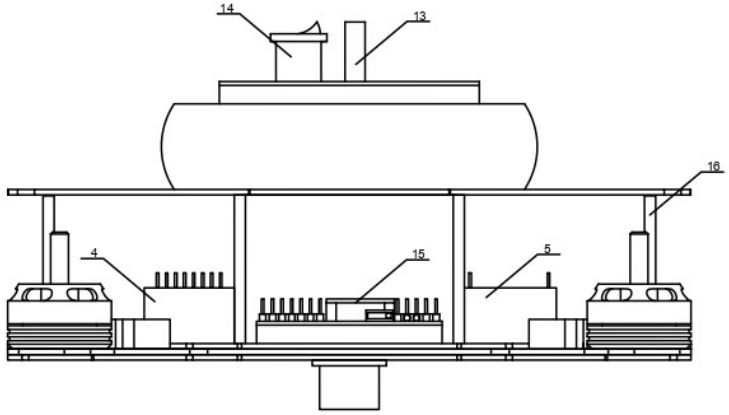
Introducción

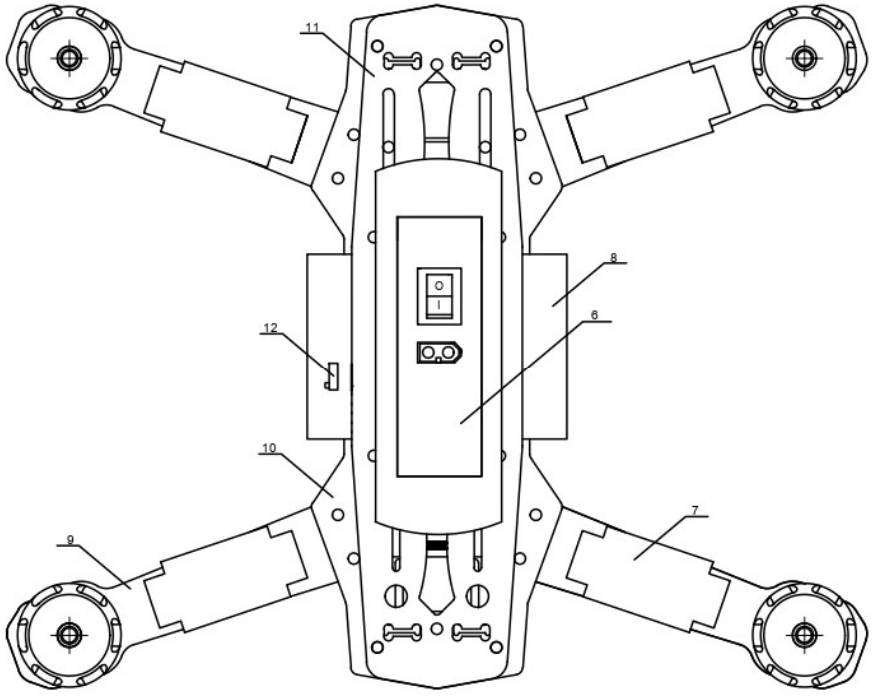
Se han representado los planos que se consideraban dentro del alcance del proyecto, siendo estos los referentes a la distribución de componentes, diagramas de conexiones y esquemas electrónicos del dispositivo. Por lo tanto, queda fuera del alcance tanto los planos referentes a la elaboración de los componentes tales como el fuselaje, motores, ESCs, microcontrolador entre otros. Para la elaboración de los planos se ha partido de los modelos en 3d del fuselaje y motores de los que se han extraído sus vistas mediante Solidworks.

Para el fuselaje se utilizo el modelo de Ravvinoff, «Racing drone zmr250», Escala 1:1, del 13 agosto de 2015, https://www.thingiverse.com/thing:966368, de https://www.thingiverse.com.

Para el motor se utilizó el modelo de G. Kress, «Crazepony DX2205 2300KV Brushless motor», Escala 1:1, del 14 de abril de 2021, https://grabcad.com/library/crazepony-dx2205-2300kv-brushless-motor-1 de https://grabcad.com.

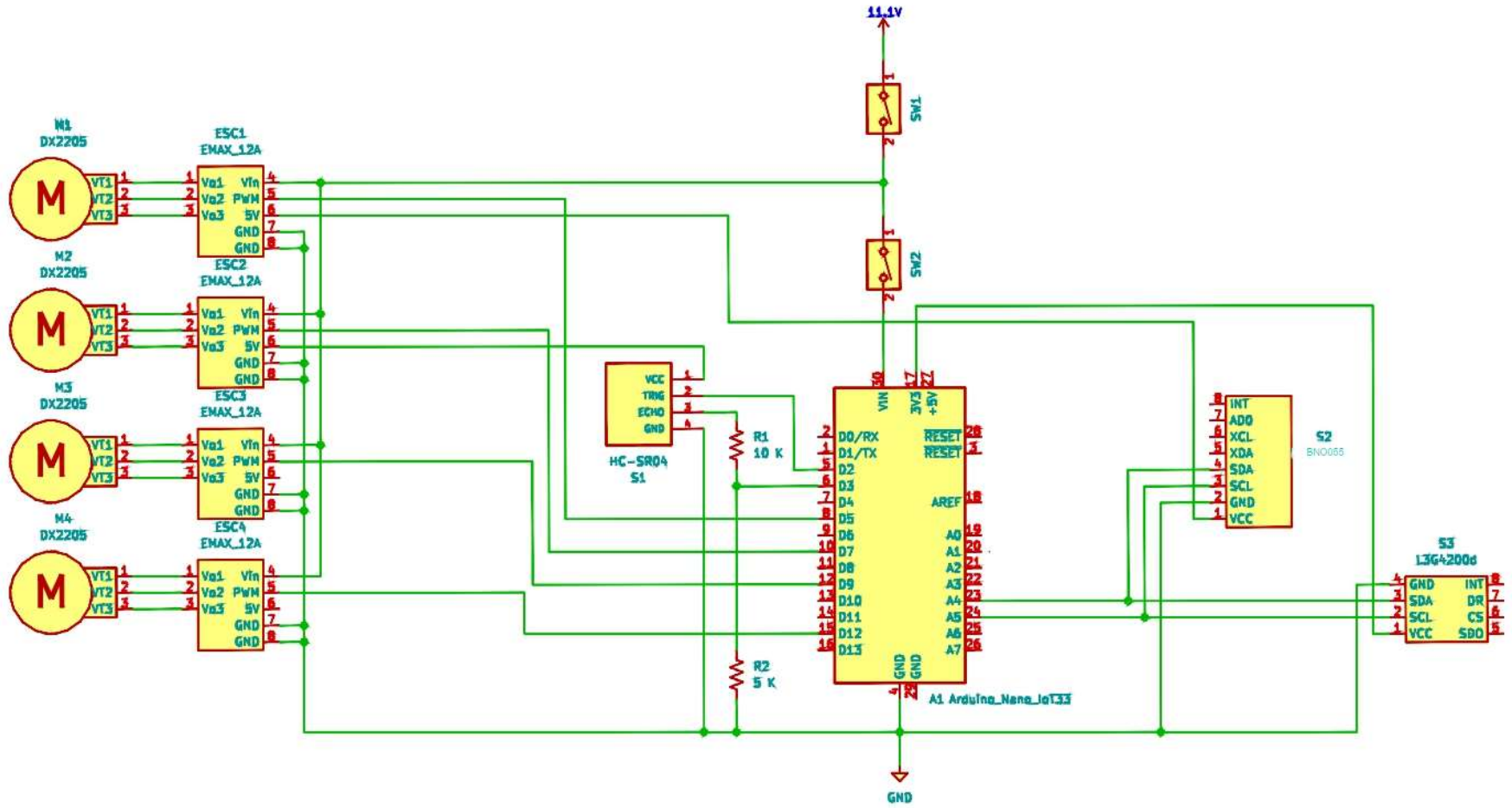






16	6	Varilla del frame QAV250	Nºref.fabric.: Dilwe79r0ews5np-01
15	1	Arduino Nano IOT 33	Nº ref. fabric.: ABX00027
14	1	Interruptor 12 V	Nºref. fabric.: INTERR-2PIN-ONOFF
13	1	Conector XT60 hembra	ASIN ref: B074RGT76J
12	1	Interruptor Micro	Nºref. fabric.: RUNCCI-YUN
11	1	Placa superior del frame QAV250	Nºref.fabric: Dilwe79r0ews5np-01
10	2	Placa inferior del frame QAV250	Nºref.fabric: Dilwe79r0ews5np-01
9	4	Brazo del frame QAV250	Nºref.fabric: Dilwe79r0ews5np-01
8	1	Placa principal	Plano 08
7	4	ESC EMAX BLHELI 12A	ASIN ref: B07M7DHJ2C
6	1	Placa de alimentación	Plano 05
5	1	Encapsulado y módulo L3G4200D giroscopio	ASIN ref: B07DJ4RWHH
4	1	Encapsulado y módulo BNO055 IMU	ASIN ref: B08GY7WKZ3
3	1	Módulo HC-SR04 ultrasonidos	ASIN ref: B07TKVPPHF
2	4	Motor Brushless DX22055 2300KV	ASIN ref: B075731ZJM
1	1	Batería LIPO 3S 11.1 V 35C 2200mAh	ASIN ref: B08H82KFS6
MARCA	CANTIDAD	DENOMINACIÓN	PLANO, NORMA,FABRICANTE

Titulo del proyecto: Titular:	Diseño, montaje e implementación del control de altura de un cuadricóptero	Fecha: 17/05/2023 Escala: 1:2
Autores: M. Fontalba	Denominación del plano Plano conjunto prototipo completo	Número del plano 01

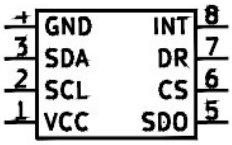


Titulo del proyecto: Titular:	Diseño, montaje e implementación del control de altura de un cuadricóptero	Fecha: 17/05/2023 Escala: SE
Autores: M. Fontalba	Denominación del plano: Esquemático electrónico completo	Número del plano: 02

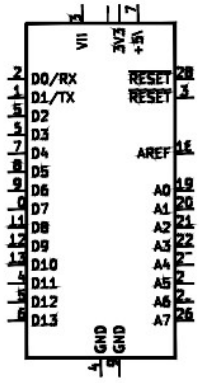
Conexión con alimentación

Conexión a masa

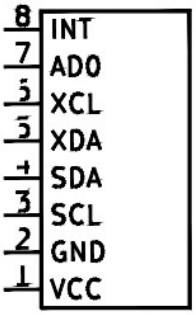
Resistencia



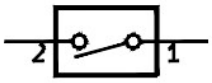
Gyro L3G4200D



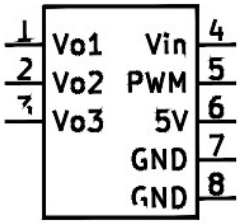
Arduino nano IOT 33



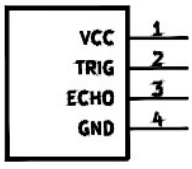
IMU BNO055



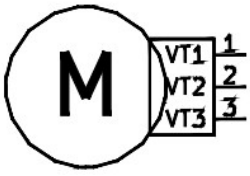
Interruptor



ESC

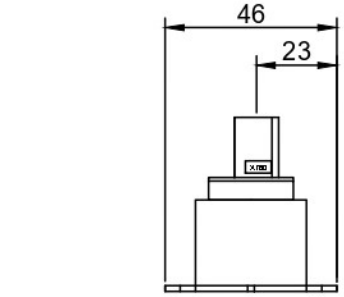


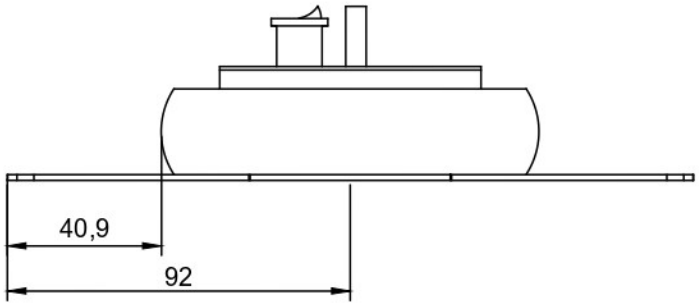
Ultrasonidos HC-SR04

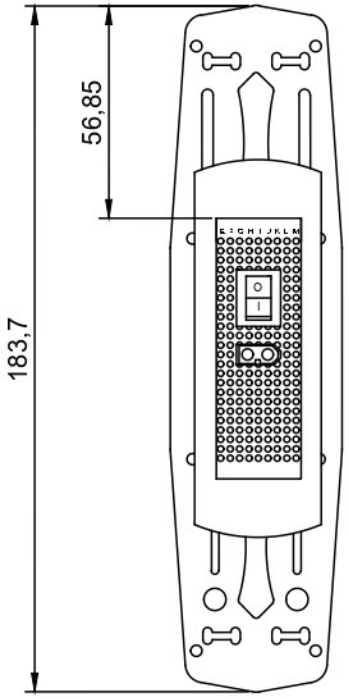


Motor brushless

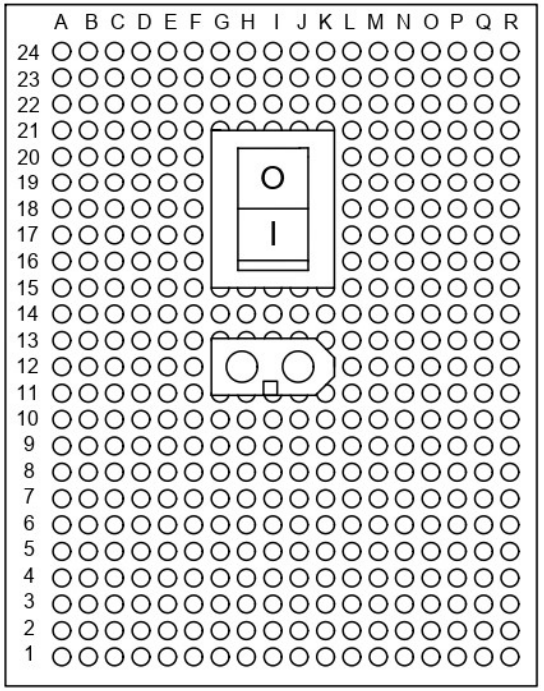
Titulo del proyecto: Diseño, montaje e implementación del control de altura de un cuadricóptero Titular:		Fecha: 17/05/2023 Escala: SE
Autores: M. Fontalba	Denominación del plano Simbología electrónica	Número del plano 03

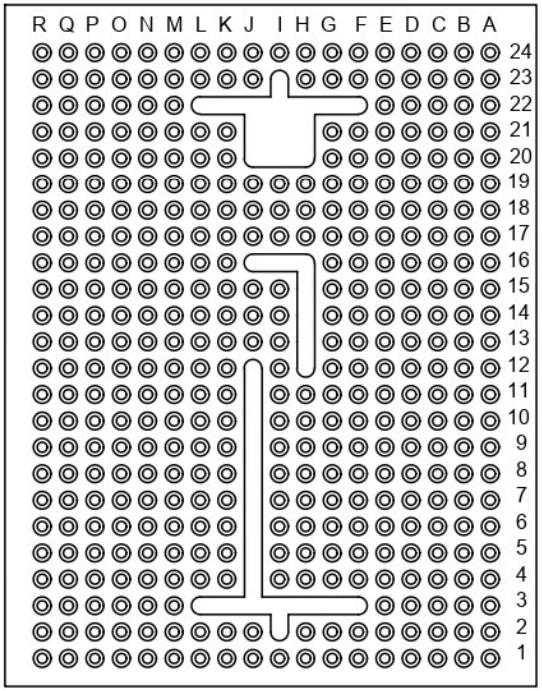




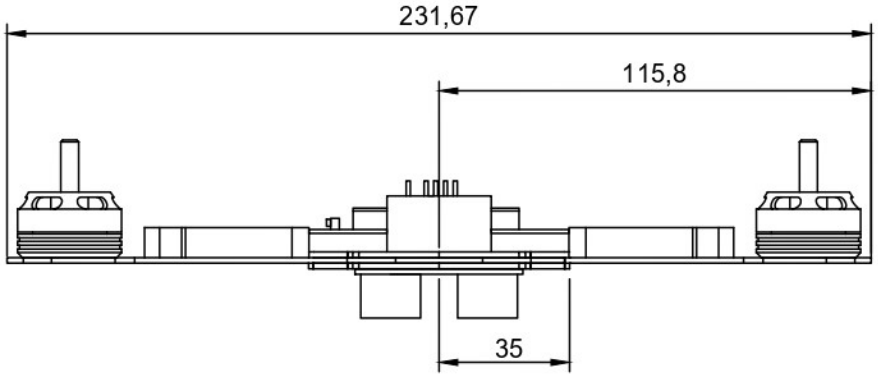


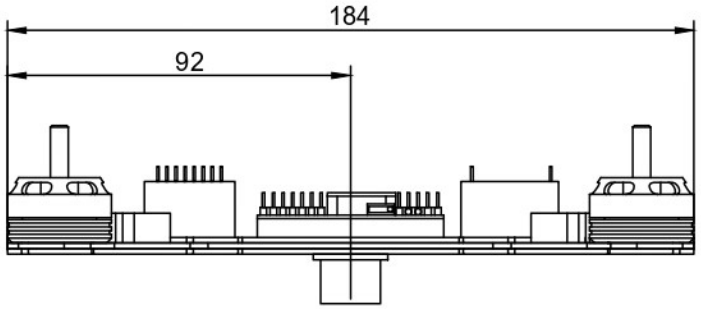
Titulo del proyecto: Diseño, montaje e implementación del control de altura de un cuadricóptero Titular:		Fecha: 17/05/2023 Escala: 1:2
Autores: M. Fontalba	Denominación del plano: Placa superior	Número del plano: 04

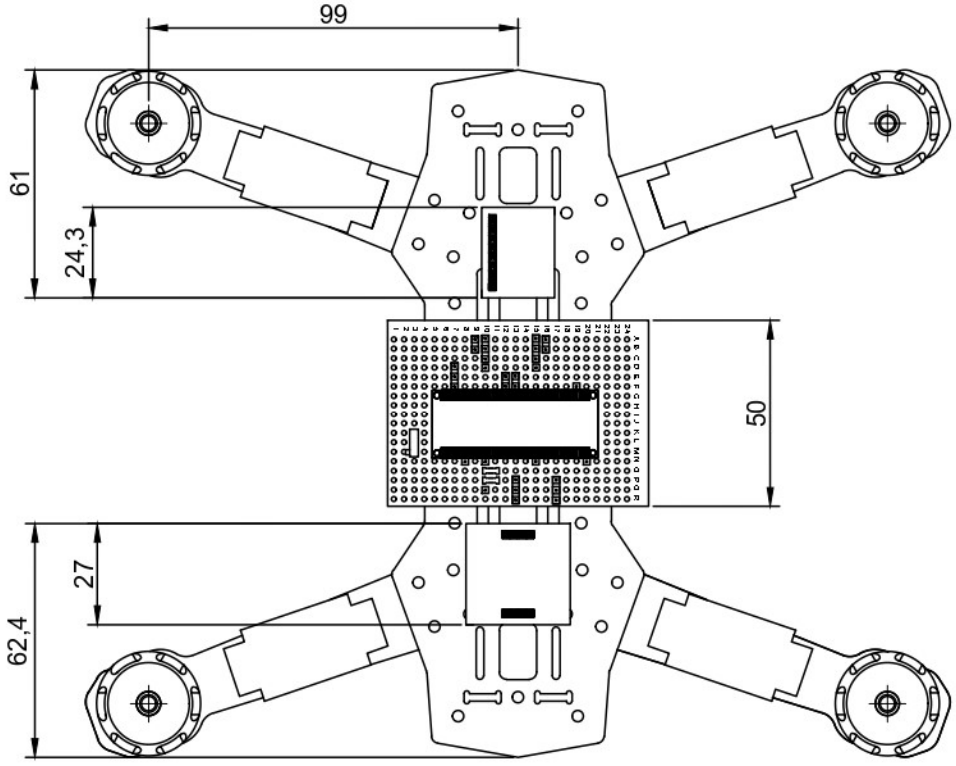




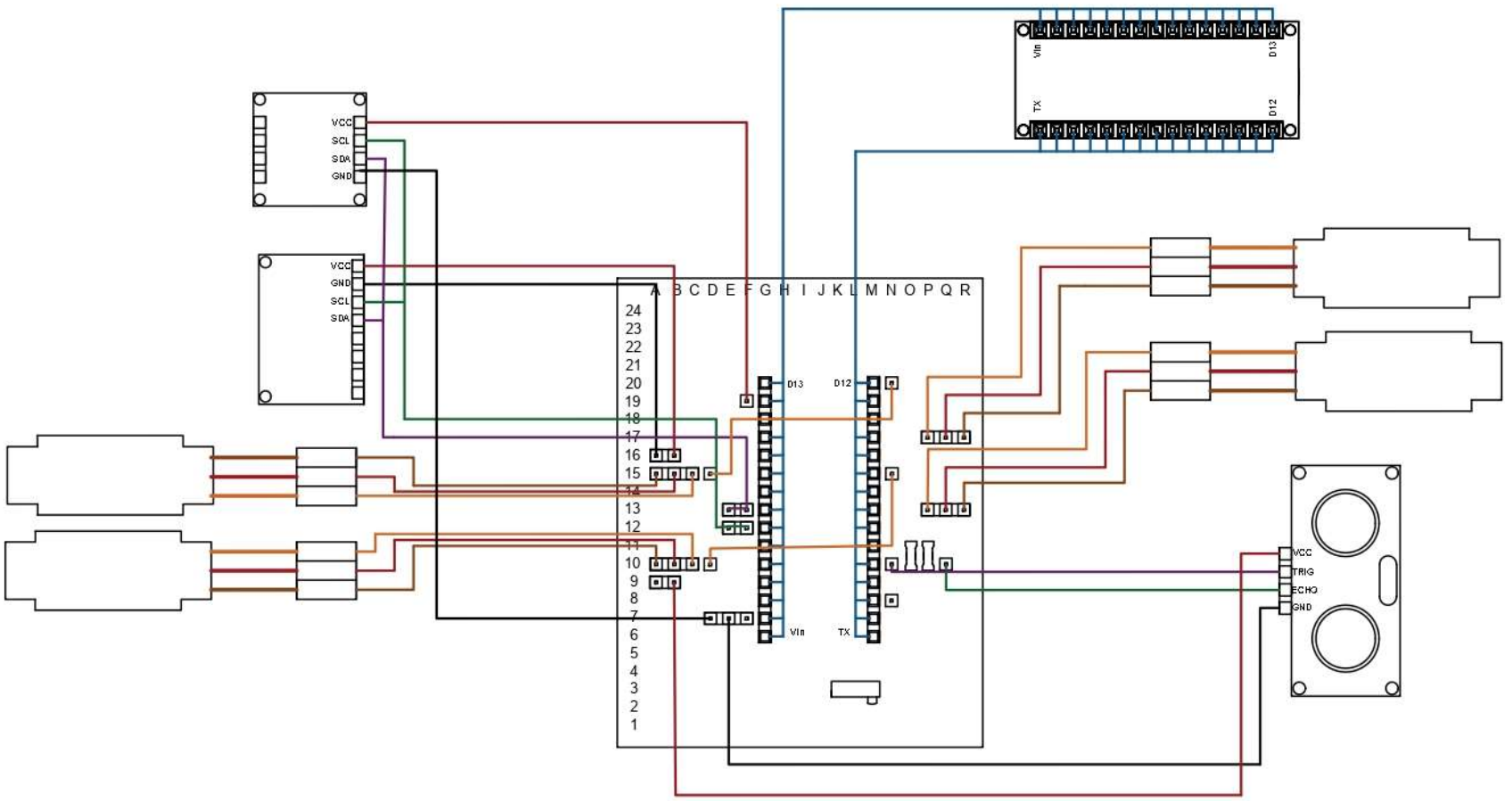
Titulo del proyecto: Diseño, montaje e implementación del control de altura de un cuadricóptero Titular:		Fecha: 17/05/2023 Escala: SE
Autores: M. Fontalba	Denominación del plano Esquema Placa de alimentación	Número del plano 05



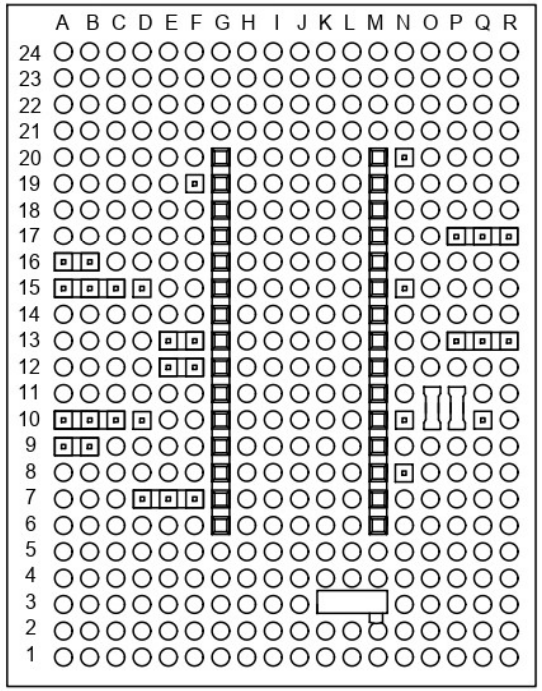


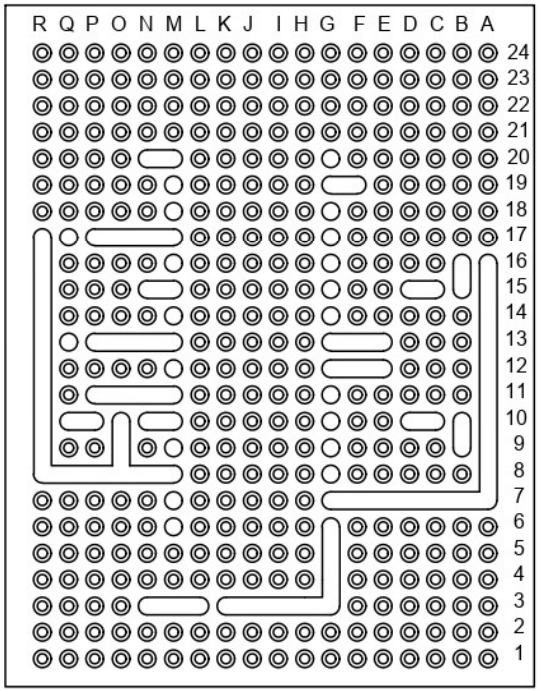


Titulo del proyecto: Diseño, montaje e implementación del control de altura de un cuadricóptero Titular:		Fecha: 17/05/2023 Escala: 1:2
Autores: M. Fontalba	Denominación del plano Placa inferior	Número del plano 06



Titulo del proyecto: Diseño, montaje e implementación del control de altura de un cuadricóptero Titular:		Fecha: 17/05/2023 Escala: SE
Autores: M. Fontalba	Denominación del plano Diagrama de conexiones placa principal	Número del plano 07





Titulo del proyecto: Diseño, montaje e implementación del control de altura de un cuadricóptero Titular:		Fecha: 17/05/2023 Escala: SE
Autores: M. Fontalba	Denominación del plano Esquema placa principal	Número del plano 08

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje e implementación del control de altura de

un cuadricóptero

Documento Nº3: PLIEGO

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Fontalba Roca, Marc

Tutor/a: González Sorribes, Antonio

CURSO ACADÉMICO: 2022 / 2023

Índice del pliego:

1. Definición y alcance del pliego

El objeto de este documento es fijar las condiciones técnicas mínimas que se deben cumplir en la fabricación y montaje de un cuadricóptero.

El ámbito de aplicación del documento se extiende a todos los sistemas mecánicos y electrónicos que forma parte de la aeronave. En determinados supuestos se podrán adoptar, por la propia naturaleza de este o del desarrollo tecnológico, soluciones diferentes a las exigidas en este documento, siempre que quede justificada su necesidad y no implique una disminución de las exigencias mínimas de calidad especificadas en el mismo.

2. Condiciones y normas de carácter general

En lo referente al ámbito legislativo, la normativa aplicada a la fabricación de drones es el reglamento delegado en el BOE 2019/945 que estandariza los requerimientos y especificaciones para fabricantes de UAS. Además, incluye una clasificación de aeronaves por peso donde el dispositivo diseñado se ubica en las aeronaves de tipo C1. Por lo tanto, se deberán cumplir con las especificaciones establecidas por la normativa para los vehículos aéreos de este grupo:

- No deberá superar los 900 gramos de masa máxima de despegue.
- No deberá superar la velocidad de 19 m/s en vuelo horizontal.
- No deberá poder superar los 120 metros de altura de vuelo.
- Deberá ser controlable de manera segura por lo que respecta a estabilidad, maniobrabilidad y rendimiento siguiendo con las instrucciones del fabricante.
- Tener la resistencia mecánica exigida para resistir toda tensión a la que este sometida durante su uso.
- Deberá estar fabricado de manera que reduzca al mínimo las lesiones a las personas durante su funcionamiento.
- Disponer de un método fiable y predecible para recuperar la conexión con la aeronave cuando se pierda el enlace de datos con esta.
- Deberá tener un nivel de potencia sonora ponderado A garantizando una lwa determinado e indicado en la propia aeronave o embalaje.
- Deberá estar alimentado con electricidad y a una tensión nominal no superior a los 24 V de CC o su equivalente en CA.
- Deberá tener un número de serie físico que cumpla con la norma ASIN/CTA-2063.
- Deberá tener una identificación a distancia directa que permita cargar el número de registro del operador de la aeronave y garantice en tiempo real durante su funcionamiento la difusión periódica directa una serie de datos especificados.
- Estar equipado con un sistema de geoconsciencia.
- Deberá transmitir una señal de alerta cuando la batería alcance un nivel bajo de manera que le piloto tenga tiempo a aterrizarla.
- Debe comercializarse con un manual de usuario.

Por otro lado, deberá ceñirse a la siguiente normativa europea 2014/30/UE sobre la compatibilidad electromagnética, la directiva 2014/53/UE sobre la comercialización de equipos radioeléctricos y la directiva 2011/65/UE sobre restricciones de uso de sustancias peligrosas en aparatos electrónicos.

3. Condiciones Técnicas

En este apartado se recogen condiciones técnicas para la construcción o compra de productos que permitan el correcto cumplimiento de las necesidades que requiere el dispositivo diseñado para su fabricación.

En el caso de que los materiales no fueran los especificados, los que se utilicen deberán cumplir los requisitos mínimos de funcionamiento y tolerancia que se requiere, siendo obligatorio que sean normalizados y sometidos a la aprobación del director del proyecto.

Todos los trabajos se ejecutarán con estricta sujeción al proyecto que ha servido de base a la contratación y a las modificaciones que hayan sido aprobadas. En caso de dudas u omisiones, o con motivo de reforma del presupuesto, se formará un comité entre proyectistas, Director del proyecto y, si se cree oportuno, el contratista, para decidir la solución más adecuada y económica.

4. Materiales

El dispositivo consta de dos tipos de materiales descritos en el organigrama del proyecto, estos son los materiales mecánicos y los materiales electrónicos.

4.1 Materiales mecánicos

4.1.1 Frame

El frame deberá tener características similares al QAV250 del fabricante Dilwe ref Dilwe79r0ews5np-01. Es decir, deberá estar fabricado de fibra de carbono, además deberá ser capaz de resistir las tensiones que se generen durante el vuelo. Por otro lado, deberá tener una dimensión diagonal de 250 mm y un peso no mayor a los 250 gramos. A su vez, la montura de los motores ubicados en los brazos del frame deberá ser compatibles con motores para monturas de 12 a 16 mm con M2 o 19 mm con M3.

4.1.2 Hélices

Las hélices deberán ser de triple aspa con 12,7 cm de diámetro y 11,4 cm de paso. Su peso no deberá ser mayor a los 6 gramos y deberá tener un orificio de montura de 5

mm. Además, deberá estar fabricada en policarbonato y se deberán adquirir cuatro hélices iguales pero un par deberá ser para giros CW y el otro par deberá ser para giro CCW. Por otro lado, las hélices deberán garantizar que con los cuatro motores seleccionados se pueda conseguir un empuje máximo superior al doble del peso de la aeronave. Además, cada hélice deberá llevar grabado el sentido de giro de esta.

4.2 Materiales electrónicos

4.2.1 Batería

La betería deberá ser tipo LIPO de 3 células con una tensión nominal de 11,1 V. Por otro lado, su capacidad no deberá ser inferior a los 2200 mAh y su velocidad de descarga no deberá ser inferior a los 35 C. Además, no deberá superar las dimensiones de 10,3 x 3, 6 x 2,5 cm y su peso no deberá ser mayor a los 200 gramos. Además, deberá contar con un terminal XT60 y una etiqueta con las especificaciones de tensión, capacidad, velocidad de descarga y marca del fabricante.

4.2.2 Motores

Los motores deberán ser brushless y de peso no superior a los 30 gramos. La montura de los motores deberá ser compatible con la del frame y junto con las hélices han de poder generar un empuje en conjunto no inferior a al doble del peso de la aeronave. Además, han de poder soportar la tensión de 11.1 V. Por lo que respecta al peso, deberá tener un peso no superior a los 30 gramos. En cuanto a las dimensiones, ha de ser de una altura de 31.5 mm y 27.9 mm de ancho. Además, deberá llevar una indicación del sentido de giro y modelo del motor.

4.2.3 ESC

El ESC debe ser capaz de soportar los 12A de corriente nominal y los 11.1 V de tensión nominal. Además, ha de ser programable y contar con un BEC lineal integrado cuya salida deberá ser no superior a los 5 V. Por otro lado, el variador debe contar con protección de bajo voltaje, contra sobrecalentamiento y contra estrangulamiento. En cuanto a dimensionamiento, no ha de ser superior a los 42 mm de largo, 20 mm de ancho y 8 de alto. Por lo que respecta al peso, no deberá superar los 30 gramos. Además, deberá llevar una etiqueta que indique la corriente nominal e información del fabricante.

4.2.4 Microcontrolador

El microcontrolador deberá tener al menos cuatro pines capaces de enviar señales PWM de al menos 2 ms de duración. Por otro lado, ha de contar con pines capaces de

establecer una comunicación por el protocolo I2C. Su reloj interno deberá ser capaz de trabajar por encima de los 16 MHz y deberá contar con un módulo interno que permita la comunicación por Wi-Fi. Además, deberá ser capaz de soportar una tensión de alimentación de 11.1. Por lo que respecta al dimensionamiento no deberá ser mayor a los 48 mm de largo y 18 mm de ancho.

4.2.5 Ultrasonidos

El sensor de ultrasonidos deberá soportar los 5 V de alimentación y deberá ser capaz de tomar medidas de entre 2 cm a los 2 m pudiendo superar este último valor. Además, su señal de accionamiento deberá estar por debajo de los 15 us de duración. Por lo que respecta a las dimensiones, deberá no ser mayor a los 45 mm de ancho, 20 mm de largo y 15 alto. En cuanto al peso, no deberá ser mayor a los 20 gramos.

4.2.6 IMU

La IMU deberá poder soportar los 5 V de alimentación nominal. Por otro lado, deberá poder establecer comunicación por el protocolo I2C y ha de poder medir tanto aceleración como velocidad angular en tres ejes. Además, deberá contar con un rango programable de entre 250 a 2000 º/s para la velocidad y 2 a 16 g para la aceleración. El módulo, no deberá ser superior a los 25 mm de largo y 17 mm de ancho. También deberá ser capaz de medir la posición angular y deberá contar con un filtro Kalman implementado a bajo nivel. Por otro lado, su peso no será superior a los 5 gramos. Además, deberá tener indicado en la PCB los ejes a medir y el sentido positivo del giro de cada eje.

4.2.7 Giroscopio

El giroscopio deberá soportar una tensión nominal de 5 V y poder establecer comunicación por el protocolo I2C. Además, deberá ser capaz de tomar medidas en tres ejes con un rango programable de entre 250 a 2000 º/s. Por otro lado, deberá tener una anchura de 21 mm y un largo de 20 mm. En referencia al peso, no deberá superar los 8 gramos. Además, deberá tener indicado en la PCB los ejes a medir y el sentido positivo del giro de cada eje.

4.2.8 Interruptor

Interruptor capaz de soportar 16 A y 11.1 V de tensión nominal en CC. La carcasa deberá estar fabricada de Nylon 66 y han de estar fabricados de óxido de estaño de plata. Sus dimensiones han de ser 21 mm de largo y 15 mm de ancho. Deberá llevar indicado los valores de protección y la marca.

4.2.9 Interruptor micro

El interruptor micro deberá ser capaz de soportar 2 A y un voltaje nominal superior 11.1 V. Sus dimensiones deberán no superar los 12.7 mm de largo, 6.6 mm de alto y 13.2 mm de ancho. Deberá llevar indicado los valores de protección y la marca.

5. Condiciones de ejecución

La fabricación del dispositivo se dividirá en tres frases, la primera que tratará de la preparación de la electrónica y la soldadura de las placas de alimentación y principal, una segunda fase que corresponde con el montaje de la base inferior fuselaje con la electrónica correspondiente y una última fase que corresponde con el montaje de la base superior con su electrónica junto con la unión de la base superior inferior y superior.

Para la preparación de la electrónica, se deberán soldar mediante estaño headers macho a cada uno de los sensores y el microcontrolador cuando estos no vengan soldados de fábrica. Posteriormente, se realizará la fabricación de la placa principal, en una placa perforada de 7 cm de largo y 5 de cm de ancho se deberán soldar con estaño los headers macho, hembra y el interruptor micro correspondiente con el plano 08. A su vez, se deberán soldar las vías en la placa perforada siguiendo con el mismo plano 08, una vez soldadas la vía se colocará el Arduino a la placa conforme al esquema de conexiones, plano 07. Seguidamente, se fabricará la placa de alimentación en una placa perforada igual que la anterior, donde se deberán soldar el interruptor y el conector hembra XT60 conforme al plano 05. Así mismo, se deberán soldar las vías correspondientes con el plano y los cables de salida de la placa que deberán tener una sección 1,5 mm. A continuación, se deberá recubrir las vías de ambas placas con silicona aislante. Además, se adherirá la cara inferior de ambas placar a una base de espuma aislante. Por último, se deberán crimpar los terminales tipo bala a los cables de los motores, entradas y salidas de ESC, salidas de la placa de alimentación y entradas de la placa principal. Respecto a los cables de los motores y las salidas de la placa de alimentación se deberán emplear conectores tipo bala hembra MPV1-156. Por otro lado, para los cables de los ESCs y la placa principal se deberán emplear conectores tipo macho MPV1-156.

En cuanto a la fabricación de la base inferior del UAV se procederá de la siguiente forma. En primer lugar, se deberán colocar ocho tornillos de M3 y 6 mm en una de las dos bases inferior iguales. Estos tornillos, se deberán colocar en los orificios correspondientes para sostener las varillas de aluminio tal como indica el fabricante del frame. En segundo lugar, se deberá colocar los cuatros brazos del UAV entre las dos bases inferiores iguales tal como se describe en el manual de ensamblaje de forma que la base con los ocho tornillos quede en la parte superior con la cabeza de los tornillos queden boca abajo. A continuación, se deberán atornillar los brazos atravesando los orificios de las dos placas y brazos mediante tornillos de M3 y 10 mm, por cada brazo

se deberán utilizar cuatro tronillos. En tercer lugar, se deberá colocar el tren de aterrizaje en cada uno de los brazos como indica en el fabricante. En cuarto lugar, se adherirá la IMU encapsulada en espuma de forma que quede ubicada conforme el plano 06 y deberá quedar colocada de manera que el eje Y dibujado en la IMU apunte hacía fuera de la aeronave. A continuación, se adherirá con silicona el giroscopio con su encapsulado conforme al plano 06 y deberá quedar colocado de forma que los ejes dibujados en el módulo estén igual que en la IMU. Posteriormente, se deberá adherir la placa principal con una base de espuma en conforme el plano 06. Seguidamente, se deberá adherir el sensor de ultrasonidos como queda descrito en el plano 06. A continuación, se deberán atornillar mediante tornillos de M3 6 mm los motores a cada uno de los brazos de forma que los que giren en el mismo sentido queden ubicados en esquinas diagonalmente opuestas. Posteriormente, se deberá colocar los ESC en cada uno de los brazos de forma que los cables que van al motor queden la más cercanos a estos. A continuación, se deberán asegurar los ESC a los brazos mediante bridas que no dejen holgura y queden bien sujetos a los brazos. Además, se deberán unir los cables con terminales tipo bala de motores con los correspondientes a los de los ESC. Posteriormente, se deberán cablear los sensores mediante cablecillos de longitud de 5 cm a la placa principal y los ESC mediante los cables con terminal tipo servo conforme al plano 07. Finalmente, se colocarán las varillas en cada uno de los ochos tornillos que se han colocado antes de colocar los brazos.

Por lo que respecta a la base superior, se procederá de la siguiente forma. En primer lugar, se deberán pasar ocho tornillos de métrica 3 y 6 mm por los orificios correspondientes a la tornillería para las varillas de aluminio como indica el fabricante. En segundo, lugar se deberá atornillar la base a las varillas de la base inferior. En tercer lugar, se deberá colocar la batería conforme al plano 04 y sobre esta, se deberá colocar la placa de alimentación con una base de espuma. A continuación, se deberá pasar dos correas por debajo de la base superior y por encima de la placa principal. Estas correas, se deberán apretar de forma que aseguren la batería y la placa de alimentación a la base superior. Posteriormente, se colocarán las hélices en cada uno de los motores de forma que las hélices y el giro del motor sean del mismo sentido CW o CCW. A continuación, se deberán unir los conectores tipo bala de la placa de alimentación a los conectores tipo bala de la placa principal y ESCs.Por último, quedará calibrar el centro de gravedad de la aeronave. Para ello, se accionarán todos los motores a una velocidad suficiente como para elevar el dispositivo. Si en el despegue presenta derivas hacia un lado, se deberá añadir algo de peso en el lado opuesto ya que estas derivas son causa principalmente de un centro de gravedad alejado del centro simétrico de la aeronave.

6. Pruebas de servicio

-Se deberán quitar las hélices de los motores y posteriormente se deberán atar cuerdas finas en cada uno de ellos de forma que cuando estos gires se enrollen al eje del motor. A continuación, se accionará cada uno de los motores y se comprobará que el sentido de giro es el correcto. Finalmente, se deberá detener los motores y corroborar que el

sentido de giro es el adecuado en función de en qué sentido se han enrollado las cuerdas.

-Se deberán desconectar todos los dispositivos de la placa principal pero mantenido la conexión con la placa de alimentación. A su vez, deberá desconectar los ESCs de esta última. Posteriormente, con la batería conectada a la placa de alimentación y con un multímetro se deberá comprobar que las tensiones de ambas placas sean correctas y no se hayan producido cortocircuitos o circuitos abiertos erróneos en la fase de soldadura.

-Se deberá comprobar que la IMU se ha colocado y mide correctamente leyendo los valores de cabeceo y alabeo. Para ello, se deberá emplear un código que utilice únicamente la función de lectura del sensor del código principal. A continuación, se deberá mover la aeronave comprobando que los ángulos medidos y el movimiento coincide. Para esta prueba no se deberán emplear los motores.

-Se deberá comprobar que el sensor de ultrasonidos mide y funciona correctamente. Para ello se deberá, cargar un programa que devuelva los valores de distancia medidos por el sensor. Posteriormente, se deberán tomar medidas respecto al suelo subiendo y bajando el UAV. Para esta prueba no se deberán emplear los motores.

-Se deberá colocar el dispositivo sobre una base plana, de peso superior a la tracción del UAV y con orificios que permitan amarrar la aeronave a la base mediante bridas, permitiendo unos pocos centímetros de holgura. Posteriormente, se deberán poner todos los motores en marcha a la misma tensión. Si la aeronave despega de forma perpendicular al suelo sin presentar desviaciones en los ángulos de cabeceo y alabeo la prueba se habrá realizado con éxito. Si no es el caso, se deberá compensar la deriva de la aeronave agregando contrapesos para corregir la posición del centro de masas del UAV al centro simétrico de esta

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje e implementación del control de altura de un cuadricóptero

Documento Nº4: PRESUPUESTO

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Fontalba Roca, Marc

Tutor/a: González Sorribes, Antonio

CURSO ACADÉMICO: 2022 / 2023

Índice de presupuestos:

1. Capítulo 1: Base inferior

1.1 Precios Unitarios

1. Precios Unitarios				
Ref	**Ud**	**Descripción**	**Precio**	
Materiales				
m1	ud.	Arduino Nano IoT33	33,14 €	
m2	ud.	IMU BNO055	39,13 €	
m3	ud.	Giroscopio L3G4200d	5,99 €	
m4	ud.	Ultrasonidos HC-SR04	5,49 €	
m5	ud.	Fuselaje QAV250	30,88 €	
m6	ud.	Motor	12,50 €	
m7	ud.	ESC	23,44 €	
m8	ud.	PerfBoard	0,80 €	
m9	ud.	Interruptor micro	0,20 €	
m10	ud.	Headers Hembra	0,01 €	
m11	ud.	Headers Macho	0,01 €	
m12	cm.	Cable 1,5 mm^2	0,01 €	
m13	ud.	Terminal tipo bala	0,16 €	
m14	ud.	Resistencia 5kΩ	0,05 €	
m15	ud.	Resistencia 10kΩ	0,05 €	
m16	ud.	Bridas	0,01 €	
m17	ud.	Hélices	1,25 €	
M.O.D.				

h1	h	Oficial 1ª mecánica	20,00 €	(€/h)
h2	h	Ayudante mecánica	15,90 €	(€/h)
h3	h	Oficial 1ª electrónica	20,00 €	(€/h)
h4	h	Ayudante electrónica	12,82 €	(€/h)
Secciones				
s1	ch	Montaje	25,20 €	(€/h)
s2	ch	Soldadura	20,40 €	(€/h)
s3	ch	Crimpado	24,00 €	(€/h)
s4	ch	Adhesión	22,20 €	(€/h)

1.2 Cuadro de precios auxiliares

2. Cuadro de precios Auxiliares					
Ref	**Ud**	**Descripción**	**Precio**	**Cantidad**	**Parcial**
d11	ud.	Placa principal, formada por placa perforada a la que se han soldado headers macho, hembra, cable de 1,5 mm^2 y terminales tipo bala crimpados.			
Materiales					
m8	ud.	PerfBoard	0,80 €	1	0,80 €
m9	ud.	Interruptor micro	0,20 €	1	0,20 €
m10	ud.	Headers Hembra	0,01 €	30	0,25 €
m11	ud.	Headers Macho	0,01 €	24	0,20 €
m12	cm.	Cable 1,5 mm2	0,01 €	10	0,05 €
m13	ud.	Terminal tipo bala	0,16 €	2	0,31 €
m14	ud.	Resistencia 5kΩ	0,05 €	1	0,05 €
m15	ud.	Resistencia 10kΩ	0,05 €	1	0,05 €

Secciones					
s2	ch	Soldadura	20,40 €	0,5	10,20 €
s3	ch	Crimpado	24,00 €	0,08	2,00 €
M.O.D.					
h3	h	Oficial 1ª electrónica	20,00 €	0,58	11,67 €
h4	h	Ayudante electrónica	12,82 €	0,58	7,48 €
				TOTAL	33,27 €
d12	ud.	Base inferior montada compuesto por brazos, placas inferiores, motores y varillas atornilladas.			
Materiales					
m5	ud.	Fuselaje QAV250	30,88 €	0,5	15,44 €
m6	ud.	Motor	12,50 €	4	49,99 €
Secciones					
s1	ch	Montaje	25,20 €	0,25	6,30 €
M.O.D.					
h2	h	Ayudante mecánica	15,90 €	0,25	3,98 €
h4	h	Ayudante electrónica	12,82 €	0,25	3,21 €
				TOTAL	78,91 €

1.3 Cuadro de precios descompuestos

3. Cuadro de precios Descompuestos					
D1	ud.	Base inferior montada con la placa principal, los ESCs, la IMU, el giroscopio y los ultrasonidos adheridos a la base y hélices montadas.			
Materiales					
d11	ud.	Placa principal, formada por placa perforada a la que se han soldado headers macho, hembra, cable de 1,5 mm2 y terminales tipo bala crimpados.	33,27 €	1	33,27 €
d12	ud.	Base inferior montada compuesto por brazos, placas inferiores, motores y varillas atornilladas.	78,91 €	1	78,91 €
m1	ud.	Arduino Nano IoT33	33,14 €	1	33,14 €
m2	ud.	IMU BNO055	39,13 €	1	39,13 €
m3	ud.	Giroscopio L3G4200d	5,99 €	1	5,99 €
m4	ud.	Ultrasonidos HC-SR04	5,49 €	1	5,49 €
m7	ud.	ESC	23,44 €	4	93,76 €
m16	ud.	Bridas	0,01	4	0,04 €
Secciones					
s1	ch	Montaje	25,2	0,13	3,36 €
s4	ch	Adhesión	22,20 €	0,17	3,70 €
M.O.D.					
h1	h	Oficial 1ª mecánica	20,00 €	0,17	3,33 €

h2	h	Ayudante mecánica	15,90 €	0,17	2,65 €
CD02					
%	Porcentaje	Costes Directos	0,02		6,06 €
				TOTAL	308,83 €

2. Capítulo 2: Base superior

2.1 Precios Unitarios

1. Precios Unitarios				
Ref	**Ud**	**Descripción**	**Precio**	
Materiales				
m1	ud.	Fuselaje QAV250	30,88 €	
m2	ud.	Batería 3S RoarignTop	19,40 €	
m3	ud.	XT60 Hembra	0,70 €	
m4	ud.	Interruptor	3,50 €	
m5	cm.	Cable 1,5 mm^2	0,01 €	
m6	ud.	Correas	7,00 €	
m7	ud.	Terminal tipo bala	0,16 €	
m8	ud.	PerfBoard	0,80 €	
M.O.D.				
h1	h	Oficial 1ª mecánica	20,00 €	(€/h)
h2	h	Ayudante mecánica	15,90 €	(€/h)

h3	h	Oficial 1ª electrónica	20,00 €	(€/h)
h4	h	Ayudante electrónica	12,82 €	(€/h)
Secciones				
s1	ch	Montaje	25,20 €	(€/h)
s2	ch	Soldadura	20,40 €	(€/h)
s3	ch	Crimpado	24,00 €	(€/h)

2.2 Cuadro de precios auxiliares

2. Cuadro de precios Auxiliares					
Ref	**Ud**	**Descripción**	**Precio**	**Cantidad**	**Parcial**
d21	ud.	Placa de alimentación, conformada por placa perforada a la que se sueldan el interruptor, conector XT60 hembra y cables 1,5 mm^2 crimpados con conectores bala.			
Materiales					
m1	ud.	Fuselaje QAV250	30,88 €	0,5	15,44 €
m3	ud.	XT60 Hembra	0,70 €	1	0,70 €
m4	ud.	Interruptor	3,50 €	1	3,50 €
m5	cm.	Cable 1,5 mm2	0,01 €	30	0,30 €
m7	ud.	Terminal tipo bala	0,16 €	5	0,78 €
m8	ud.	PerfBoard	0,80 €	1	0,80 €

Secciones					
s2	ch	Soldadura	20,40 €	0,25	5,10 €
s3	ch	Crimpado	24,00 €	0,17	4,00 €
M.O.D.					
h3	h	Oficial 1ª electrónica	20,00 €	0,42	8,33 €
h4	h	Ayudante electrónica	12,82 €	0,42	5,34 €
				TOTAL	44,30 €

2.3 Cuadro de precios descompuestos

D2	ud.	Base superior con la batería y la placa de alimentación mediante correas			
Materiales					
d21	ud.	Placa de alimentación, conformada por placa perforada a la que se sueldan el interruptor, conector XT60 hembra y cables 1,5 mm2 crimpados con conectores bala.	44,30 €	1	44,30 €
m2	ud.	Batería 3S RoarignTop	19,40 €	1	19,40 €
m6	ud.	Correas	7,00 €	2	14,00 €
Secciones					
s1	ch	Montaje	25,20 €	0,17	4,20 €
M.O.D.					

h1	h	Oficial 1ª mecánica	20,00 €	0,17	3,33 €
h2	h	Ayudante mecánica	15,90 €	0,17	2,65 €
CD02					
%	Porcentaje	Costes Directos	0,02		1,76 €
				TOTAL	89,64 €

3. Capítulo 3: Programación

3.1 Precios Unitarios

1. Precios Unitarios				
Ref	**Ud**	**Descripción**	**Precio (€)**	
M.O.D.				
h1	h	Programador	24,00 €	(€/h)
Secciones				
s1	ch	Programación del codigo	15,00 €	(€/h)

3.2 Cuadro de precios descompuestos

2. Cuadro de precios Descompuestos					
Ref	**Ud**	**Descripción**	**Precio**	**Cantidad**	**Parcial**
D3	ud.	Programación del codigo del dispositivo			

M.O.D.					
h1	h	Programador	24,00 €	6	144,00 €
Secciones					
s1	ch	Programación del codigo	15,00 €	1	15,00 €
CD02					
%	Porcentaje	Costes Directos	0,02		2,88 €
				TOTAL:	161,88 €

4. Capítulo 4: Pruebas de servicio

4.1 Precios Unitarios

1. Precios Unitarios				
Ref	**Ud**	**Descripción**	**Precio (€)**	
M.O.D.				
h1	h	Técnico Superior de Laboratorio	20,00 €	(€/h)
h2	h	Ayudante de Laboratorio	12,00 €	(€/h)
Transporte	km	Coche	0,15 €	
Secciones				
s1	servicio	Prueba de sentido de rotación de motores	10,00 €	(€/h)
s2	servicio	Prueba de tensión en la placa principal	14,00 €	(€/h)
s3	servicio	Prueba de lectura de la IMU	5,00 €	(€/h)

s4	servicio	Prueba de lectura del ultrasonido	4,30 €	(€/h)
s5	servicio	Prueba de estabilidad de la aeronave	12,00 €	(€/h)

4.2 Cuadro de precios descompuestos

2. Cuadro de precios Descompuestos					
Ref	**Ud**	**Descripción**	**Precio**	**Cantidad**	**Parcial**
D4	ud.	Prueba de la rotación de los motores en banco de pruebas			
Secciones					
s1	servicio	Prueba de sentido de rotación de motores	10,00 €	1,00	10,00 €
M.O.D.					
h1	h	Técnico Superior de Laboratorio	35,00 €	0,08	2,92 €
h2	h	Ayudante de Laboratorio	20,00 €	0,08	1,67 €
Transporte	km	Coche	0,15 €	2,00	0,30 €
CD02					
%	Porcentaje	Costes Directos	0,02		0,29 €
				TOTAL:	15,18 €

D5	ud.	Prueba de las tensiones en la placa de alimentación mediante multímetro			
Secciones					
s2	servicion	Prueba de tensiónes en la placa principal	14,00 €	1,00	14,00 €
M.O.D.					
h1	h	Técnico Superior de Laboratorio	35,00 €	0,08	2,92 €
h2	h	Ayudante de Laboratorio	20,00 €	0,08	1,67 €
Transporte	km	Coche	0,15	2,00	0,30 €
CD02					
%	Porcentaje	Costes Directos	0,02		0,37 €
				TOTAL:	19,26 €
D6	ud.	Prueba de lectura de la IMU y ejes Yaw, Pitch, Roll			
Secciones					
s3	servicio	Prueba de lectura de la IMU	5,00 €	1,00	5,00 €
M.O.D.					
h1	h	Técnico Superior de Laboratorio	35,00 €	0,08	2,92 €
h2	h	Ayudante de Laboratorio	20,00 €	0,08	1,67 €
Transporte	km	Coche	0,15	2,00	0,30 €
CD02					
%	Porcentaje	Costes Directos	0,02		0,19 €
				TOTAL:	10,08 €

D7	ud.	Prueba de mediciones y funcionamiento del sensor de ultrasonidos en banco de pruebas			
Secciones					
s4	servicio	Prueba de lectura del ultrasonido	4,30 €	1,00	4,30 €
M.O.D.					
h1	h	Técnico Superior de Laboratorio	35,00 €	0,08	2,92 €
h2	h	Ayudante de Laboratorio	20,00 €	0,08	1,67 €
Transporte	km	Coche	0,15	2,00	0,30 €
CD02					
%	Porcentaje	Costes Directos	0,02		0,18 €
				TOTAL:	9,36 €
D8	ud.	Prueba de estabilidad del centro de masa de la aeronave en banco de pruebas			
Secciones					
s5	servicio	Prueba de estabilidad de la aeronave	12	1,00	12,00 €
M.O.D.					
h1	h	Técnico Superior de Laboratorio	35,00 €	0,08	2,92 €
h2	h	Ayudante de Laboratorio	20,00 €	0,08	1,67 €
Transporte	Km	Coche	0,15	2,00	0,30 €
CD02					
%	Porcentaje	Costes Directos	0,02		0,33 €
				TOTAL:	17,22 €

5. Estado de mediciones

4.3 Estado de mediciones			
Ref	**Ud**	**Descripción**	**Cantidad**
Capítulo 1			
D1	ud.	Base inferior montada con la placa principal, los ESCs, la IMU , el giroscopio y el ultrasonidos adheridos a la base y hélices montadas.	1
Capítulo 2			
D2	ud.	Base superior con la batería y la placa de alimentación mediante correas	1
Capítulo 3			
D3	ud.	Programación del código del dispositivo	1
Capítulo 4			
D4	ud.	Prueba de la rotación de los motores en banco de pruebas	1
D5	ud.	Prueba de las tensiones en la placa de alimentación mediante multímetro	1
D6	ud.	Prueba de lectura de la IMU y ejes Yaw, Pitch, Roll	1
D7	ud.	Prueba de mediciones y funcionamiento del sensor de ultrasonidos en banco de pruebas	1
D8	ud.	Prueba de estabilidad del centro de masa de la aeronave en banco de pruebas	1

6. Presupuesto de ejecución material

Ref	Ud	Descripción	Precio	Cantidad	Parcial
Capítulo 1					
D1	ud.	Base inferior montada con la placa principal, los ESCs ,la IMU , el giroscopio y el ultrasonidos adheridos a la base y hélices montadas.	308,83 €	1	308,83 €
Capítulo 2					
D2	ud.	Base superior con la batería y la placa de alimentación mediante correas	89,64 €	1	89,64 €
Capítulo 3					
D3	ud.	Programación del código del dispositivo	161,88 €	1	161,88 €
Capítulo 4					
D4	ud.	Prueba de la rotación de los motores en banco de pruebas	15,18 €	1	15,18 €
D5	ud.	Prueba de las tensiones en la placa de alimentación mediante multímetro	19,26 €	1	19,26 €
D6	ud.	Prueba de lectura de la IMU y ejes Yaw, Pitch, Roll	10,08 €	1	10,08 €
D7	ud.	Prueba de mediciones y funcionamiento del sensor de ultrasonidos en banco de pruebas	9,36 €	1	9,36 €

D8	ud.	Prueba de estabilidad del centro de masa de la aeronave en banco de pruebas	17,22 €	1	17,22 €
				TOTAL:	631,43 €

7. Resumen del presupuesto

RESUMEN DEL PRESUPUESTO					
Ref	**Ud**	**Descripción**	**Precio**	**Cantidad**	**IMPORTE**
Capítulo 1	ud.	Montaje base inferior	308,83 €	1	308,83 €
Capítulo 2	ud.	Montaje base superior	89,64 €	1	89,64 €
Capítulo 3	ud.	Programación	161,88 €	1	161,88 €
Capítulo 4	ud.	Pruebas de servicio	71,08 €	1	71,08 €
Presupuesto de ejecución material	ud.	Presupuesto de ejecución material	631,43 €	1	631,43 €
Gastos Generales	%		0,06	1	37,89 €
Beneficio industrial	%		0,13	1	82,09 €
IVA	%		0,21	1	132,60 €

Porcentaje de honorarios y trámites	%		0,10	1	63,14 €
			PPRESUPUESTO DE EJECUCIÓN GENERAL:		**947,14 €**