CENTRO DE INGENIERÍA Y DESARROLLO INDUSTRIAL

INFORME DE PROYECTO INDUSTRIAL

CUADRICÓPTERO CONTROLADO POR PID

PARA OBTENER LA

ESPECIALIDAD DE TECNÓLOGO EN MECATRÓNICA

PRESENTA

ALUMNA: LUCÍA LIMONES PÉREZ

ASESOR: M.C. ROBERTO SOSA CRUZ

CO-ASESOR: M.C. J. ARTURO VELARDE SÁNCHEZ

SANTIAGO DE QUERÉTARO, QUERÉTARO; AGOSTO 2015

TABLA DE CONTENIDO

AGRADECIMIENTOS

Les agradezco a mis padres José Limones Vázquez y María del Socorro Pérez Luna así como a mi hermana Ana Marcela Limones Pérez por todo el apoyo brindado en cada momento de mi vida.

También agradezco al CIDESI por permitirme la oportunidad de formar parte de la generación 2015 de la Especialidad de Tecnólogo en Mecatrónica. Así mismo agradezco a mis compañeros de trabajo de CIDESI Rafael de los Santos, Ricardo Vega, Jaime Trejo, Diego Martínez, Oscar Rodríguez y Julio Gavito y a mis asesores Roberto Sosa y Arturo Velarde por todo el apoyo y la motivación brindada. Finalmente agradezco a mi asesor en control Ottmar Rafael Uriza Gosebruch.

RESUMEN

El avance tecnológico con respecto a cuadricópteros ha aumentado con el paso del tiempo. Hoy en día suelen ser de tamaños más compactos y cuentan con sistemas de control avanzados. Una de sus aplicaciones es la evaluación de cultivos agrícolas por medio de redes de cuadricópteros.

El siguiente reporte expone los componentes para la construcción de un cuadricóptero así como el desarrollo de un sistema de control PID cuyas variables a controlar son Roll, Pitch, Yaw. También se llevó a cabo el diseño y construcción de una tarjeta de control que permite la implementación de sensores para la obtención de la retroalimentación del sistema. El sistema cuenta con la posibilidad de comunicarse vía inalámbrica por medio del uso de un módulo XBee PRO S3B con una computadora.

Palabras clave: Cuadricóptero, Roll, Pitch, Yaw, control PID, tarjeta de control, módulo XBee PRO S3B.

ABSTRACT

The technologic advance about quadcopters has been increased with the time. Currently these aircrafts are more compacts and have advanced control systems. One of their applications it's the evaluation of agricultural crops through quadcopters networks.

This report exposes the devices for the quadcopters construction as well as the development of a PID control system implemented, where the variables to control are Roll, Pitch, Yaw. Also, an electronic control board where the sensors were implemented was design and constructed. The system is able to communicate by using wireless communication with an XBee PRO S3B module with any computer.

Key Words: Quadcopter, Roll, Pitch, Yaw, PID control, control board, XBee PRO S3B module.

CAPÍTULO I.- INTRODUCCIÓN

En los últimos años el campo de la electrónica ha tenido un desarrollo importante en sensores, microprocesadores y otros dispositivos, lo que permite la apertura de todo tipo de proyectos y líneas de investigación. El desarrollo de vehículos aéreos no tripulados también denominados *UAV* (por sus siglas en inglés "*Unmanned Aerial Vehicle*") se encuentran basados en sistemas de control especialmente en sistemas de despegue y aterrizaje autónomo. Existen diversos tipos de *UAVs*, entre los más característicos se encuentran los cuadricópteros o *quadcopter*.

Los UAVS pueden ser controlados de forma remota o autónoma y son capaces de realizar exploraciones en lugares de difícil acceso o de peligro para un operador humano, por lo que tienen muchas aplicaciones, por ejemplo:

- Fotografía aérea.
- Agricultura.
- Guardia costera.
- Monitoreo de ganado
- Monitoreo de tráfico vehicular.
- Servicios meteorológicos.
- Búsqueda de personas.

1.1 PLANTEAMIENTO DEL PROBLEMA

El control de la estabilidad de vuelo se realiza mediante la variación de las velocidades de cada uno de sus motores para modificar parámetros de orientación y posición. Lo que resulta difícil de maniobrar en un rango de tiempo de milisegundos debido al número de grados de libertad (D*OF*, por sus siglas en inglés "*Degrees Of Freedom*") con los que cuenta (*Roll, Pitch, Yaw*, X, Y, Z), debido a esto es preciso el desarrollo de un sistema de control que permita el correcto funcionamiento del vehículo y así mismo que facilite el manejo de éste al operador.

Por lo que es indispensable el monitoreo de las variables mencionadas con anterioridad. Esto puede lograrse a través de una interfaz de usuario sencilla que permita al operador conocer con exactitud la posición del vehículo facilitando el control de éste en áreas de difícil acceso.

1.2 JUSTIFICACIÓN

El interés en el desarrollo de vehículos aéreos no tripulados (*UAV*) ha incrementado considerablemente en los últimos años ya que poseen características como de tener un tamaño compacto y ser de fácil maniobrabilidad. Esto los hace perfectos para diversas aplicaciones como el acceso a terrenos de difícil acceso, inspección remota. Una de sus mayores ventajas es el despegue y aterrizaje de forma vertical en áreas limitadas así como la posibilidad de mantenerse estático en el aire.

En cuanto a sus aplicaciones puede ser utilizado en el campo de la agricultura para el monitoreo de sistemas de riego, fotografía aérea, inspección de ambientes complejos y/o peligrosos e incluso para uso militar y la guardia costera, detección de incendios, etc

Para esto se propone utilizar un controlador PID debido a que el acoplamiento entre las fuerzas y momentos generados por los motores y las hélices, el rozamiento con el aire, etc. ejercen ciertas perturbaciones importantes que pueden afectar la estabilización del cuadricóptero. La suma de las acciones del PID serán utilizadas para ajustar el proceso de control de PWM aplicado en los motores y con esto poder validar el funcionamiento del cuadricóptero que se posee en CIDESI.

1.3 OBJETIVO GENERAL

“Diseñar e implementar un sistema de control PID para la estabilidad de un cuadricóptero que permita al usuario su fácil maniobrabilidad”.

1.4 OBJETIVOS ESPECÍFICOS

- Diseñar y construir una tarjeta electrónica de control donde se implementen los componentes a utilizar como el controlador, los sensores y las salidas a los motores.
- Desarrollar un sistema de control inalámbrico utilizando un módulo XBee PRO S3B
- Diseñar e implementar el control PID del vehículo a través de Arduino.

1.5 ALCANCE

Se proporcionará un sistema de control PID que permita obtener la estabilidad de vuelo del cuadricóptero, así mismo diseñar y desarrollar una tarjeta electrónica que se encuentre integrada por los sensores e instrumentos a utilizar de tal forma que sea de fácil montaje en la estructura del cuadricóptero.

CAPÍTULO II.- FUNDAMENTO TEÓRICO

Un cuadricóptero es una aeronave que se eleva y se desplaza por la acción del control de cuatro motores, los cuales son instalados en las extremidades del cuadricóptero. Dichas aeronaves gozan de una gran maniobrabilidad con un incremento en la dificultad del software y la electrónica para su regulación. Una de sus principales ventajas es que pueden ser controlados de forma remota, por lo que son elegidos para aplicaciones en el campo de la investigación, inspección remota, aplicaciones militares, etc. con el propósito de evitar accidentes en zonas de difícil acceso ya que son capaces de despegar y aterrizar en áreas limitadas.

2.1 ESTADO DEL ARTE

Cuadricóptero Autónomo de Arquitectura Abierta, QA3

Para el control del QA3 se implementó un control PID para ajustar el proceso de salida del PWM aplicado en los motores. Para establecer la inclinación y la posición angular se utilizaron acelerómetros MEMS ADXL345 donde se determinó la dirección y el sentido del vector de aceleración y giroscopios MEMS ITG3200. Para la tarjeta de control fue utilizada una tarjeta madre diseñada por CIII (Centro de Investigación en Informática para la Ingeniería). La plataforma contiene el microcontrolador LPC2114/24 con núcleo ARM7. La estructura del QA3 (Figura 1) fue de Aluminio compuesta por dos tubos rectangulares (0.2x60x5) encastrados en el centro formando un ángulo de 90° donde se localiza la placa de control montada sobre gomas para amortiguar las vibraciones de los motores [1].

Figura 1.- Estructura de QA3.

Diseño e implementación de un sistema de control para un cuadricóptero.

El vehículo cuenta con una computadora de la marca Gumstix como controlador principal a bordo con Linux embebido y Xenomai, una plataforma para dar soporte en tiempo real. Se basó en controladores PID (Figura 2) con ajuste fino utilizando el algoritmo de evolución diferencial para hacer un mejor ajuste. El cuadricóptero puede comunicarse con una computadora (utilizada para configuración y apoyo en la evolución diferencial) y una tablet-PC con Android para el control en tierra a través de una red Wi-Fi. El cuadricóptero cuenta con una IMU y un sensor ultrasónico para la altitud.

El control automático de postura se subdivide en 3 controladores PID: uno para pitch, roll y yaw y un controlador PID para el control de posición (x,y,z). A continuación se presenta un diagrama a bloques con los sensores implementados (Figura 3) y el protocolo de comunicación utilizado así como los voltajes de operación de cada uno [2].

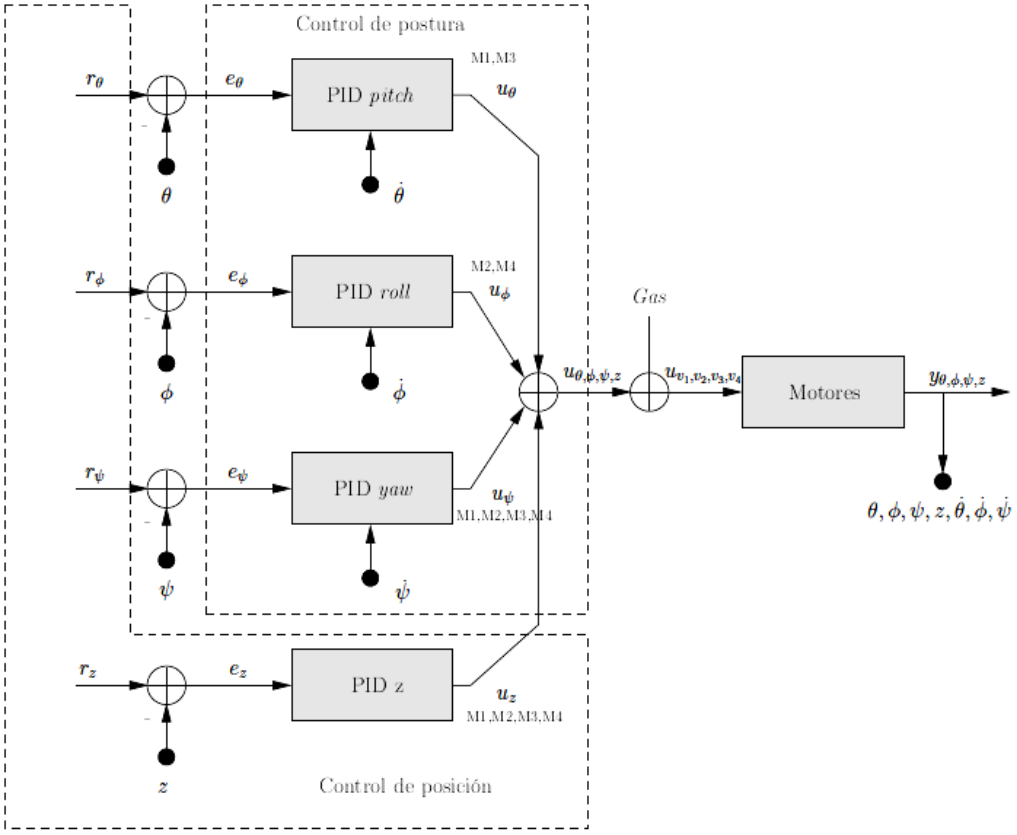


Figura 2.- Diagrama de control general del sistema.

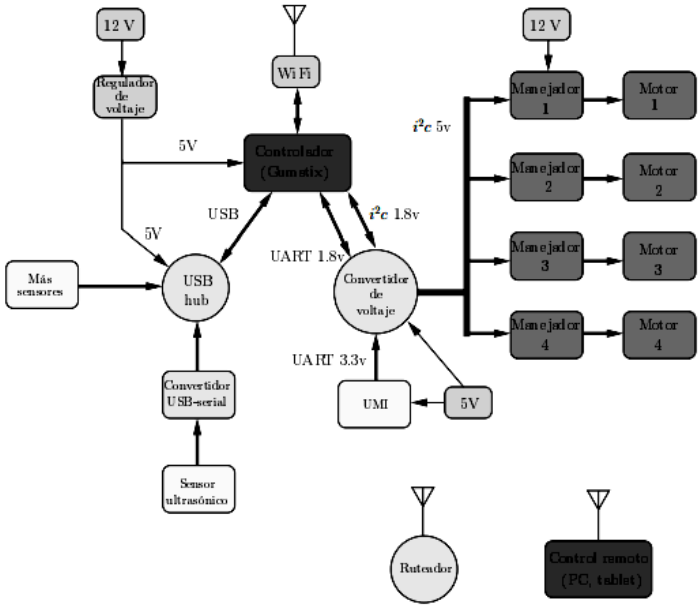


Figura 3.- Diagrama del hardware del sistema de control del cuadricóptero.

Modelado y control de un cuadricóptero.

El cuadricóptero cuenta con una tarjeta controladora Crius con procesador ATMega 2560 de Arduino y una IMU. Además se desarrollaron un conjunto de controladores basados en el modelo simplificado (dinámicas de orientación) para tener un mayor rango de posibilidades y posteriormente el uso del modelos no lineal completo (comportamiento en vuelo) para el sistema de control y así permitir el vuelo estacionario y la navegación autónoma mediante sensores de distancia (Figura 4). Los algoritmos de control diseñados fueron: [3]

- Control PID con ponderación diseñado por respuesta en frecuencia para evitar la acumulación del error en la integración.
- Control PID incremental para limitar los incrementos del mando y simplificar la lógica del mecanismo anti-*windup*.
- Control por realimentación de estado por posicionamiento de polos para optimizar la respuesta del sistema.

- Regulador LQ por realimentación de estado para obtener la matriz de ganancias de realimentación para minimizar el error de seguimiento de la referencia y el esfuerzo de control.

- Control LQ por realimentación de estado con acción integral (LQI) para corregir perturbaciones en carga que presenta el sistema.

Figura 4.- Cuadricóptero con transmisor.

Modelo de control predictivo para cuadricóptero: estudios experimentales de actitud, altitud y posición.

El control para un cuadricóptero no tripulado en un ambiente cerrado es más complejo ya que hay escasez de datos de localización, por lo que este dispositivo se encuentra basado en la unión de una IMU, un sensor ultrasónico y un sensor de flujo óptico. Se diseñó un modelo de controlador predictivo novedoso conmutable (SMPC) con el propósito de alcanzar la trayectoria precisa de control bajo la presencia de ráfagas de viento forzosas. A continuación se muestra un diagrama de Hardware que indica los sensores utilizados y los datos de salida de cada uno (Figura 5). [4]

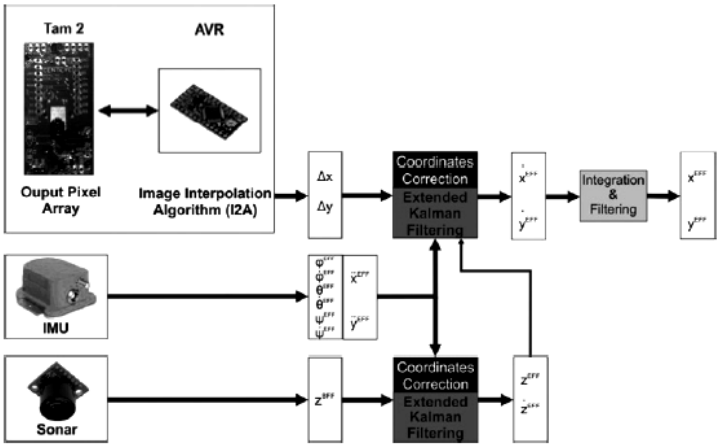


Figura 5.- Sistema de sensores de interior integrados para la estimación del estado del quadricoptero.

Navegación 3D de un sistema de vuelo autónomo de tipo quadrotor.

Se encarga de seguir puntos en el espacio por lo que los algoritmos se basaron en el cuaternión unitario para el seguimiento de posición en el espacio mediante modos deslizantes por bloques usando el algoritmo de súper *twisting*. Usa un modelo basado en la matriz de cosenos directores para la estabilización de la aeronave. El control con retroalimentación de cuaterniones se compara con su análogo de rotación por ángulos de Euler, ambos usan un control robusto sumado con un estimador de perturbaciones no presentes en el espacio de controlabilidad. La principal ventaja es la eficiencia en términos de procesamiento de datos. [5]

Sistema de control para la estabilidad y orientación de un helicóptero quadrotor.

Se presenta el diseño e implementación de un sistema de control para la estabilidad y orientación de un helicóptero tipo quadrotor. El cual tiene como unidad de procesamiento central un Arduino Mega 2560 en compañía de una unidad inercial Arduimu V3. Los controladores abordados en este trabajo son, control en cascada, control PID por ganancia programada y PID de sintonización automática por lógica difusa. Los cuales fueron sintonizados mediante un enfoque de una entrada y una salida "*fifo*" (en inglés "*first in first out*") para lo que se requirió el diseño de una serie de pruebas con la finalidad de restringir el movimiento de la aeronave a un solo grado de libertad.

Para cada controlador se realizó una prueba de estabilidad (Figura 6), comportamiento en estado transitorio y comportamiento ante perturbaciones. Para seleccionar los controladores que posteriormente se integrarían para realizar un vuelo estacionario. [6]

Figura 6.- Base para pruebas.

2.2 PRINCIPIO DE OPERACIÓN

Un cuadricóptero es un sistema de 6 grados de libertad (*Roll, Pitch, Yaw, X, Y, Z*), las fuerzas y los momentos que actúan sobre él son producidos por las variaciones en la potencia entregada a cada motor. Dos pares de motores rotan en direcciones opuestas para balancear el torque total. A través de un sistema de comunicación inalámbrico (XBee, bluetooth, radio frecuencia, etc.) se realiza la comunicación con el controlador para enviar nuevas direcciones junto con las mediciones de los sensores (acelerómetro, giroscopio, barómetro, etc.) así como mediciones externas de posición (GPS). [7]

El sistema de referencia utilizado es el B fijo al vehículo (Figura 7), dependiendo de la potencia mandada a los motores se genera una fuerza de empuje en cada uno el cuál determina el comportamiento del cuadricóptero en vuelo.

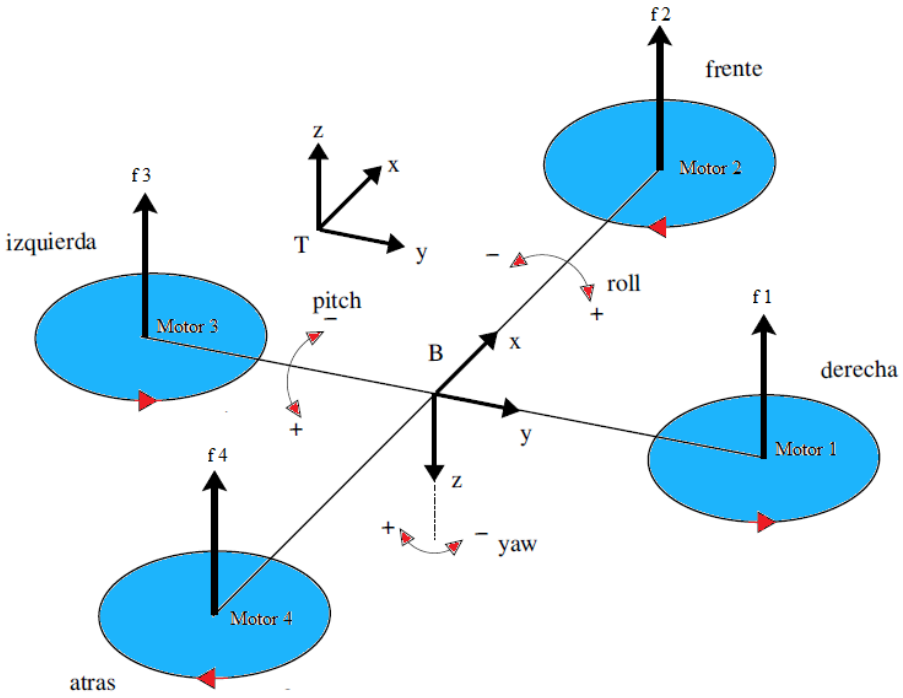


Figura 7.- Diagrama esquemático del funcionamiento de un cuadricóptero.

2.2.1 Movimiento Y*aw*

Movimiento producido en el eje "Z" por lo que se debe de aumentar por igual la potencia de un par de motores y disminuir por igual la del otro par. Al decrementar la potencia aumenta el torque, por lo que el cuadricóptero gira en sentido contrario a las hélices que están rotando con mayor potencia (Figura 8).

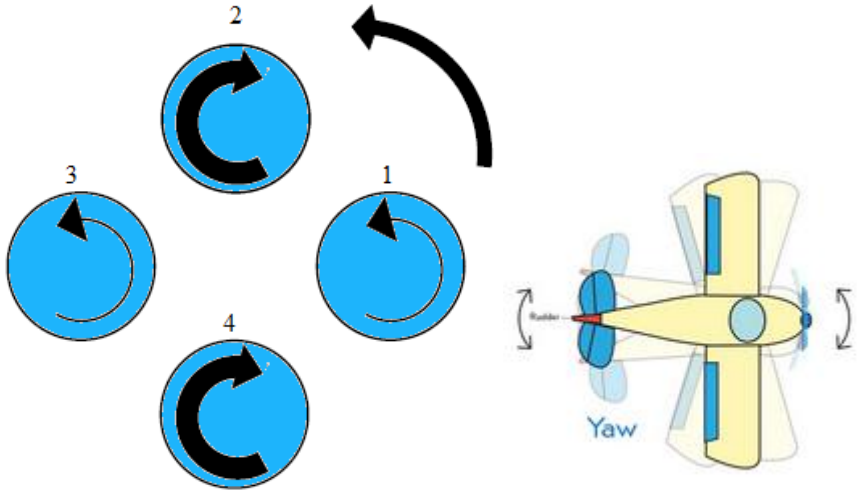


Figura 8.- Potencia en los motores para el control de Yaw.

2.2.2 Movimiento *Pitch*

Movimiento producido en el eje "Y" que corresponde al movimiento hacia adelante y atrás. Se mantiene la potencia en el motor 1 opuesto al sentido deseado, reduce al mínimo la del motor 3 y deja a los motores 2 y 4 con potencia media, ya que de esta forma el cuadricóptero se inclina a favor del sentido deseado (Figura 9).

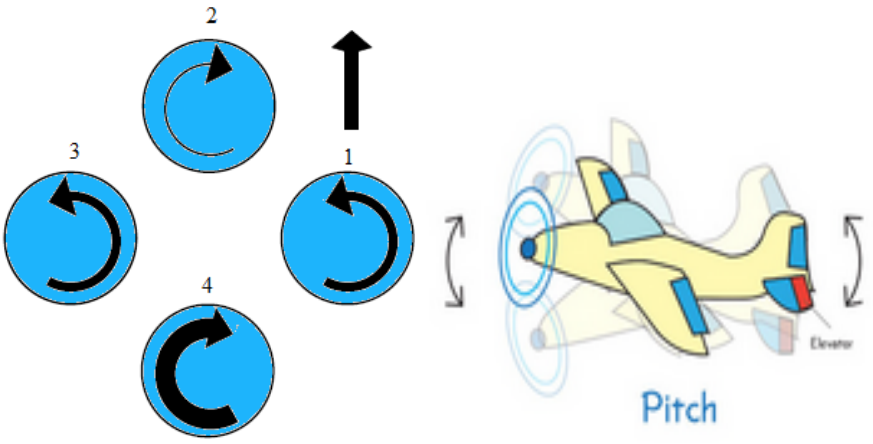


Figura 9.- Potencia en los motores para el control de Pitch.

2.2.3 Movimiento *Roll*

Movimiento producido en el eje "X", por lo que el vehículo gira a la izquierda o derecha. Para ser manipulado se aplica un empuje diferencial en los motores opuestos (Figura 10).

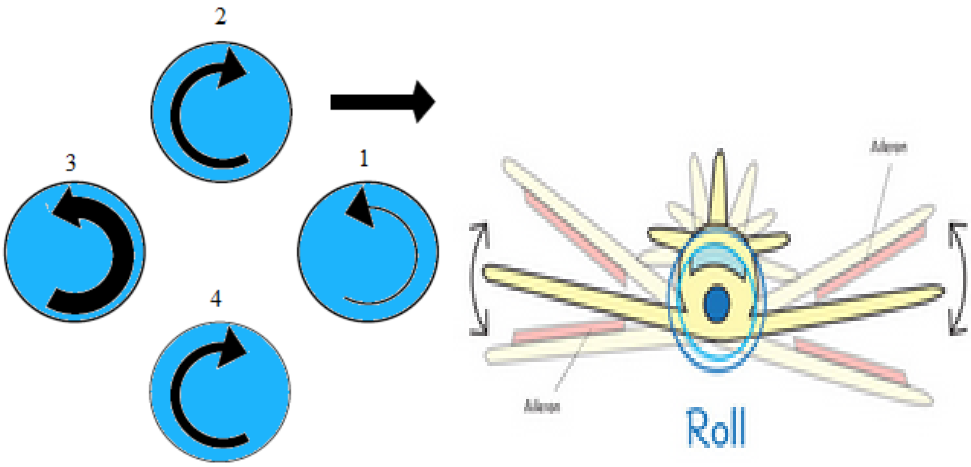


Figura 10.- Potencia en los motores para el control de Roll.

2.3 ESTRUCTURA

Para la estructura es recomendable que el marco sea rígido para soportar las fuerzas opuestas de las hélices y disminuir las vibraciones de los motores. Consta de una parte central donde es posicionada la tarjeta de control. Las extremidades se encuentran unidas a la placa central donde están conectados los motores y las hélices (Figura 11).

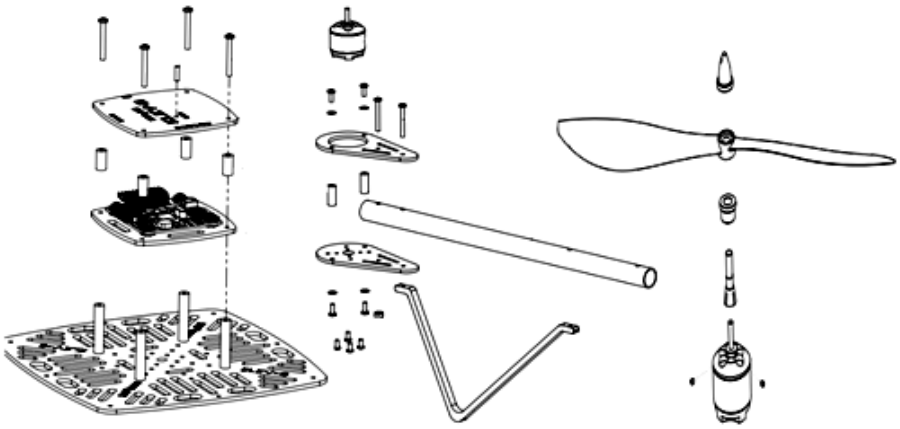


Figura 11.- Ensamblaje de piezas de un cuadricóptero.

2.3.1 Motores

Se utilizan motores sin escobillas (en inglés "*brushless*") para aeromodelismo, ya que presentan mayores ventajas como menor peso, menor costo, no requieren tanto mantenimiento, son más eficientes y generan menor interferencia electromagnética (Figura 12). Generalmente giran a un mayor número de revoluciones que los de corriente continua además de que su consumo de potencia es menor y generan menos pérdidas de potencia al no existir escobillas. Entre las características de estos motores se encuentran los siguientes parámetros.

- **Factor Kv:** Indica el número de revoluciones por minuto (rpm) a las que es capaz de girar el motor por cada volt de tensión que se le aplica. Preferentemente debe de ser elevado ya que indica una respuesta en velocidad y aceleración mayor.
- **Par proporcionado por el motor.** Si el cuadricóptero es de mayor peso el par debe de ser mayor, por lo que el valor de Kv debe de disminuir.

Figura 12.- Motor *brushless Turnigy*.

Debido a que para el funcionamiento de estos motores es necesaria la generación de tres señales se utilizan controladores electrónicos de velocidad.

2.3.2 Controlador Electrónico de Velocidad

Es el controlador de velocidad de giro del motor ESC (por sus siglas en inglés "*Electronic Speed Controller*", Figura 13) que proporciona una velocidad estable en el rotor. Permiten regular la potencia suministrada a cada motor y para su funcionamiento son alimentadas por baterías NiMH (Níquel – MetalHidruro) o Li-Po (Litio – Polímero).

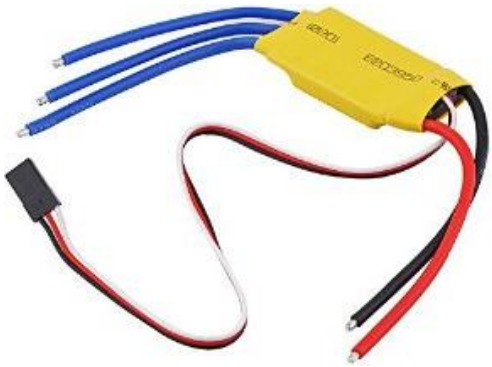


Figura 13.- Controlador electrónico de velocidad ESC.

Cada controlador está regulado por una señal PWM, cuya frecuencia para cuadricópteros es conveniente usar medio-altas. La tensión en los bornes del motor será el valor medio del tren de pulsos que estará entre 1 y 2 ms.

2.3.3 Batería

Se encarga de suministrar la alimentación a todo el sistema. Suelen estar compuestas por un número de celdas conectadas en paralelo para aumentar la corriente. Los

parámetros a considerar de una batería son la tensión de salida (relacionada con el número de celdas de la batería) y la capacidad eléctrica (cantidad total de carga producida haciendo referencia a los tiempos de carga y descarga en Amperes/Hora (Ah). Los tipos de baterías más utilizadas en el desarrollo de UAVS se exponen en la tabla 2.

Tabla 1.- Tipos de baterías para un UAV

Batería	Acrónimo	Descripción
Níquel - Cadmio	NiCd	• El voltaje por celda es de 1.2V. • Poseen efecto memoria, lo que reduce la capacidad de carga de la batería. • Son altamente tóxicas.
Níquel - Metal-Hidruro	NiMH	• El voltaje por celda es de 1.2V. • Tienen mayor capacidad que la NiCd. • Poseen menor efecto memoria. • Alta resistencia interna, lo que las limita para alimentar cargas de alta potencia.
Iones de litio	Li-ion	• El voltaje por celda es de 3.7V. • Tienen el doble de capacidad que NiCd. • No poseen efecto memoria. • Requieren de un circuito para limitar la corriente de cada celda.
Polímero de litio	Li-Po	• El voltaje por celda es de 3.7V. • Tienen una capacidad de carga de entre 5 a 12 veces la capacidad de NiCd. • No poseen efecto memoria. • Son de poco peso. • Tiempo de carga menor que NiMH.

Las baterías requieren de cargadores especiales (Figura 14) ya que no deben de ser sobrecargadas o descargadas por completo. El conector permite acceder a cada celda para una carga balanceada y así mantener la diferencia entre cada celda al mínimo.

Figura 14.- Cargador/Balanceador para baterías Li-Po.

Para el cálculo de tiempo aproximado de vuelo se calcula la potencia máxima producida por los motores sin tomar en cuenta a los demás elementos tal como se muestra en las ecuaciones 1, 2 y 3.

$$P = 4 * (Pmáx_{motores}) \quad \textbf{(ec.1)}$$

$$E = Núm_{Baterias} * V * I \quad \textbf{(ec.2)}$$

Por lo que el tiempo de vuelo se puede expresar de la siguiente manera (ecuación 3):

$$Tiempo = \frac{E}{P} * 60 \quad \textbf{(ec.3)}$$

2.3.4 Hélices

Para la construcción del cuadricóptero se necesitan 4 hélices, dos que giren en un sentido y otras dos en otro para cancelar el torque generado. El tipo de hélice utilizada en aeronaves son las llamadas "*pusher*" que generan su empuje en la cola. Existen dos parámetros importantes: el paso y el diámetro (Figura 15). [8]

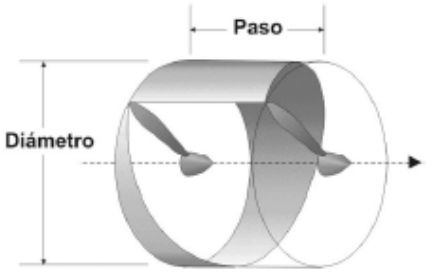


Figura 15.- Parámetros utilizados para la selección de una hélice.

El paso se calcula como la distancia que recorre la hélice en una vuelta y el diámetro se genera por medio del círculo que forma al girar e indica el área de aire que entra en el cálculo del empuje por lo que determina la eficiencia de la hélice.

2.3.5 Unidad de Medición Inercial (IMU)

Es un dispositivo electrónico que mide velocidad y orientación por medio de la unión de acelerómetros, giroscopios y en ocasiones magnetómetros. En español es conocido como Unidad de Medición Inercial (UMI o en inglés IMU "*Inertial Measurement Unit*").

Las medidas obtenidas de los sensores son procesadas y utilizadas para calcular los cambios en las velocidades de los motores. Para una nueva medición son utilizados 3 acelerómetros y 3 giroscopios que permitan obtener las medidas en las 3 dimensiones. (Figura 16).

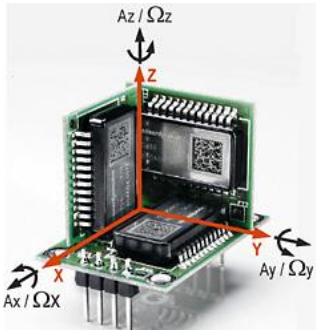


Figura 16.- Estructura interna de una IMU.

2.3.6 Giroscopio

Es un objeto esférico o en forma de disco, montado de manera que pueda girar en cualquier dirección. Se utiliza para medir la orientación y con ello, mantenerla estable. Determina la velocidad angular con respecto a cada eje. Cuando es sometido a un momento cambia la orientación de su eje girando respecto a un tercer eje (Figura 17).

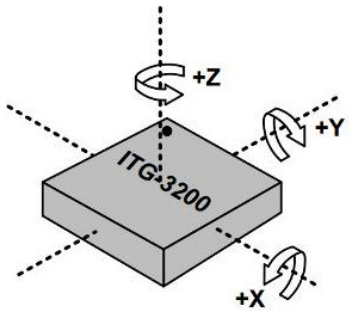


Figura 17.- Principio de funcionamiento de un giroscopio.

2.3.7 Acelerómetro

Como su nombre lo indica es un dispositivo electrónico que mide la aceleración dada por el movimiento de una masa situada en el extremo de un muelle (Figura 18), la cual dependiendo del movimiento se puede contraer o elongar generando una fuerza. La masa es un valor conocido por lo que la fuerza es la que se calcula. Por medio de la segunda ley de Newton es posible conocer la aceleración, mostrada en la ecuación 4.

$$a = \frac{F}{m} \qquad \textbf{(ec.4)}$$

Donde:

a = Aceleración (m/s^2) F = Fuerza ($N = Kgm/s_2$) M = Masa (Kg)

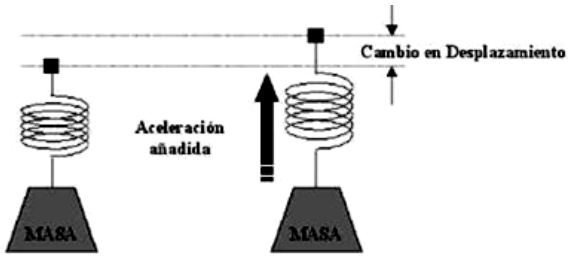


Figura 18.- Principio de funcionamiento de un acelerómetro.

La unión de 3 acelerómetros permite conocer la aceleración en cada eje (X, Y, Z); sin embargo, estos dispositivos son sensibles a las vibraciones, pudiendo generar errores de medición. Debido a ello son utilizados junto con giroscopios para obtener una medida más precisa, pudiendo distinguir entre movimientos y vibraciones.

2.3.8 GPS

Sistema de Posicionamiento Global GPS (por sus siglas en inglés "*Global Position System*") permite obtener una referencia con mayor precisión hasta centímetros de la posición actual del dispositivo. Cuenta con la ventaja de que no afectan en él los campos magnéticos externos y permite monitorizar la trayectoria del vehículo (Figura 19).

Figura 19.- Sistema de Posicionamiento Global GPS.

El GPS funciona mediante una red de 24 satélites en órbita, cuando se desea determinar la posición el receptor localiza y se conecta automáticamente a un mínimo de

3 satélites de los que recibe señales indicando la identificación y la hora. Los tipos de datos enviados por el GPS se denominan NMEA (por sus siglas en inglés "*National Marine Electronics Association*") los cuales son:

- GPGGA: Posicionamiento global de datos fijos del sistema.
- GPVTG: Velocidad respecto al suelo.
- GPGSA: GPS DOP (calidad de la señal) y satélites activos.
- GPGSV: Información de cada satélite.
- GPGLL: La posición geográfica, latitud / longitud.
- GPRMC: Hora, fecha, posición, dirección y velocidad.

La trama que más se usa es la GPGGA, ya que porta la información más relevante. Para interpretarla se toma en cuenta la numeración con la posición separada por comas:

$GPGGA, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14*15

1. Hora UTC (Tiempo Universal Coordinado) en formato hhmmss.
2. Latitud en formato ggmm.ssss.
3. Orientación en latitud: N (norte) o S (sur).
4. Longitud en formato: gggmm.ssss.
5. Orientación en longitud: E (este) o W (oeste).
6. Indicación de calidad GPS: 0 = nula, 1 = GPS fija.
7. Número de satélites visibles por el receptor: xx.
8. Dilución horizontal de posición: xx.x.
9. Altitud de la antena por encima / por debajo del niel del mar (geoidal): xxxxx.x.
10. Unidades de altitud: M (metros).
11. Separación geoidal: xxx.x.
12. Unidades de separación: M (metros).
13. Tiempo en segundos desde la última actualización: xx.
14. ID de referencia de la estación.
15. Checksum: *xx.

2.3.9 Barómetro

Mide la presión por metro cuadrado que ejerce la atmósfera sobre una superficie (Figura 20). La unida de presión atmosférica es el pascal (Pa). Las zonas con más presión son aquéllas donde las precipitaciones no son frecuentes.

Figura 20.- Barómetro de uso comercial.

Su funcionamiento está basado en la relación entre presión y altitud, la presión atmosférica desciende con la altitud aproximadamente 1 hPa (hectopascal, equivalente a 100 Pascales) por cada 8.2 metros de altitud. Toman como base de referencia el nivel del mar.

2.3.10 Microcontrolador

Es el cerebro del cuadricóptero ya que toma las medidas de sensores y las procesa para modificar las velocidades de los motores y así mantener o cambiar la orientación del cuadricóptero. Para una buena selección del controlador (Figura 21) es necesario conocer las características necesarias como los tipos de buses de comunicación con los que cuenta, velocidad de CPU, número de salidas PWM para la manipulación de la velocidad de los motores, etc.

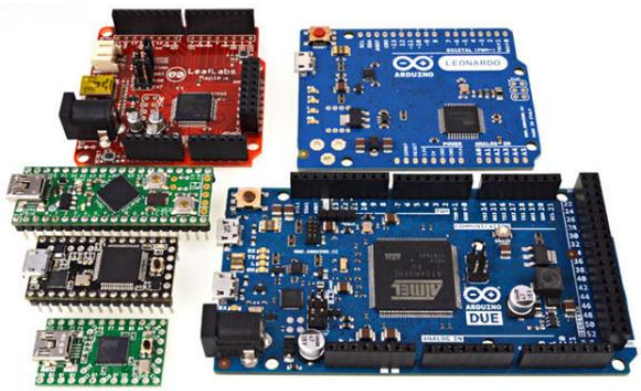


Figura 21.- Microcontrolador, el cerebro del cuadricóptero.

2.3.11 Sensor ultrasónico de distancia

El emisor genera un pulso de alta frecuencia que se va propagando hasta chocar con un objeto, una parte de la señal enviada rebota y la otra es absorbida. La parte que rebota es captada por el receptor. El sensor cuenta con un reloj que permite determinar el intervalo de tiempo de ida y vuelta del pulso para posteriormente aplicar un algoritmo y determinar la distancia a la que se encuentra el objeto (Figura 22).

Figura 22.- Funcionamiento de un sensor ultrasónico.

Los sensores de proximidad ultrasónicos cuentan con las siguientes ventajas y desventajas:

- Rango relativamente amplio (varios metros).
- Detección del objeto independientemente del color y/o material.
- Detección segura de objetos transparentes (botellas de vidrio o plástico).
- Relativamente insensibles a la suciedad y al polvo.
- Posibilidad de aplicaciones al aire libre.
- Si se utilizan para objetos con superficies inclinadas el sonido se desvía.
- Reaccionan con lentitud. El rango de frecuencia de conmutación es de 1-125 Hz.
- Generalmente son más costosos que los de proximidad óptica.

2.3.12 Sensor de distancia infrarrojo

Se encuentran compuestos por LEDS IR (Diodo Emisor de Luz Infrarroja) que producen luz en el espectro infrarrojo (no visible para el ojo humano), que puede ser detectada por una variedad de dispositivos electrónicos.

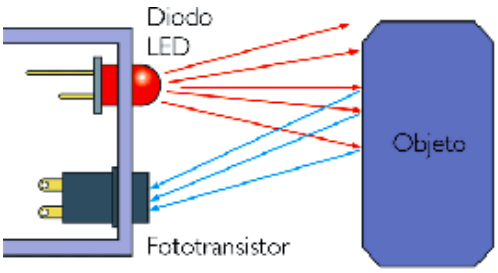


Figura 23.- Funcionamiento de un sensor infrarrojo reflexivo.

Los sensores infrarrojos pueden ser reflexivos o de barrera dependiendo de la posición del receptor. Para el caso de un cuadricóptero son utilizados los sensores infrarrojos reflexivos (Figura 23). Cuando el haz de luz infrarroja es enviado y choca contra un objeto el haz es reflejado hasta llegar al receptor de manera que se pueda convertir la luz infrarroja entrante en una corriente eléctrica. La corriente se envía a un convertidor análogico-digital (ADC) para ser interpretada por el microcontrolador. Entre sus ventajas y desventajas se encuentran las siguientes:

- Bajo voltaje de operación.
- El emisor y receptor deben de estar correctamente alineados para comunicarse.
- Corto alcance.
- No es posible realizar la detección con objetos transparentes.
- Sensible a luz exterior, clima y polvo.
- Velocidad de transmisión de datos menor.

2.3.13 Comunicación inalámbrica XBee

Permite establecer una comunicación inalámbrica (Figura 24). El XBee está basado en el estándar de comunicaciones para redes inalámbricas IEEE_802.15.4 como se muestra en la tabla 2. Cada módulo tiene una dirección única. Proveen dos formas de comunicación: modo AT (transmisión a través de un puerto serial) y modo API, con el que normalmente se trabaja es el AT. Para su uso en modo AT es necesario modificar ciertos parámetros como:

- **PAN ID:** Proporciona la identificación de toda la red, por lo que tanto el dispositivo emisor como el receptor deben de tener la misma PAN ID.
- **DH y DL o bien DT:** Determinan la dirección del dispositivo a comunicarse.
- **MY:** Dirección propia de cada modulo.

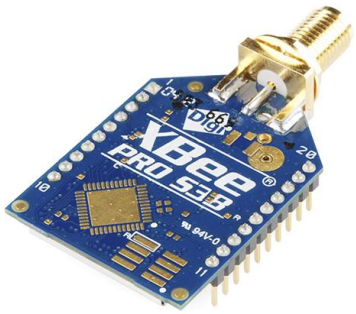


Figura 24.- Modulo XBee PRO.

Tabla 2.- Estándar de comunicación IEEE 802.15.2

IEEE 802.15.4					
Banda de Operación (MHz)	Vel. de transmisión (Kbps)	Región	distancia en función de potencia y vel. de transmisión		
			1 mW	10 mW	100 mW
868.3	20	Europa	23 m	54 m	154 m
902 - 928	40	América	-	-	-
2405 - 2480	250	Mundial	13 m	29 m	66 m

2.3.14 Filtro complementario

Debido a que un acelerómetro es muy sensible a todas las vibraciones y desplazamientos lineales por lo que los parámetros de salida serán muy ruidosos y con errores durante los desplazamientos, por lo que es conveniente filtrar la señal del acelerómetro (con un filtro pasa bajo) y el giroscopio (con un filtro pasa alto). A continuación se muestra el diagrama de determinación de los ángulos de Euler y la velocidad angular a través del filtro complementario (Figura 25).

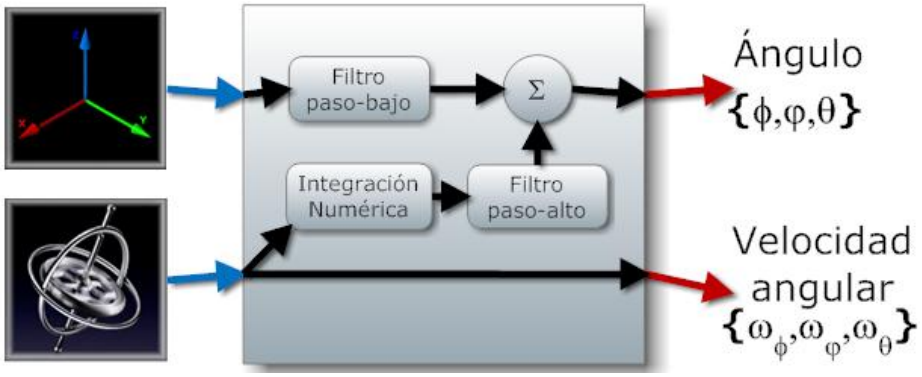


Figura 25.- Determinación de los ángulos y velocidad angular por medio de filtro complementario.

El filtro se encarga de eliminar los ruidos y las aceleraciones lineales; sin embargo, produce un retardo en la percepción del ángulo, lo cual puede provocar problemas en la estabilización ya que se pueden utilizar los datos del giroscopio lentamente (integrados en el tiempo para pasar de velocidad angular a ángulo) dado que este tiene una deriva temporal que los inutiliza. Por lo que la fórmula se expone en la ecuación 5:

$$\acute{A}ngulo = \alpha \cdot \acute{A}ngulo + giroscopio \cdot dt + (1 - \alpha) \cdot acelerómetro \qquad \textbf{(ec. 5)}$$

2.4 Método de control PID

Un sistema de control está definido como un conjunto de componentes que pueden regular su propia conducta o la de otro sistema para lograr un funcionamiento predeterminado.

Un controlador PID es un controlador de tres términos: Proporcional, integral y derivativo (Figura 26). Es un método de control de lazo cerrado utilizado en la industria para el control de sistemas; procesa el error calculado a partir de una salida deseada menos la salida medida. [9]

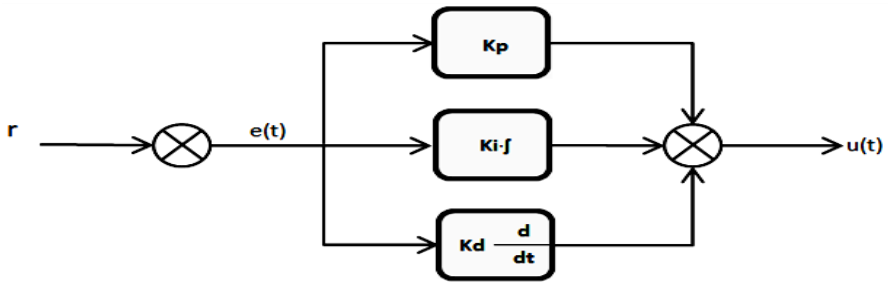


Figura 26.- Diagrama de un controlador PID.

- **Proporcional (P)**: Respuesta proporcional al error. Está acción no siempre es capaz de eliminar el error en régimen permanente y es sensible a ruidos.

$$u\ t\ = Kpe\ t\ = Kp(r\ t\ - y\ t\ ) \quad \textbf{(ec. 6)}$$

- **Integral (I)**: Da una respuesta proporcional a la integral del error. Esta acción elimina el error en régimen estacionario. Por el contrario, se obtiene un mayor tiempo de establecimiento, una respuesta más lenta y el periodo de oscilación es mayor que en el caso de la acción proporcional.

$$u\ t\ = ki\ \int_0^t e\ T\ dT \quad \textbf{(ec. 7)}$$

- **Derivativa (D)**: Da una respuesta proporcional a la derivada del error (velocidad de cambio del error). Añadiendo esta acción de control a las anteriores se aumenta la velocidad de respuesta del sistema aunque se amplifican los ruidos en el mando a altas frecuencias.

$$u\ t\ = kd\ \frac{de(t)}{dt} \quad \textbf{(ec. 8)}$$

Si la ganancia P es demasiado baja entonces el incremento del empuje no es lo suficientemente fuerte como para nivelar el cuadricóptero por lo que termina volteándose. Sí es ligeramente baja puede ser controlable aunque le tomará más tiempo llegar a un valor establecido. En cambio sí es demasiado alta el cuadricóptero oscilará, lo que hará que intente nivelarse rápidamente y pierda el equilibrio.

La ganancia I se asocia con el tiempo, por lo que determina el tiempo en que se encuentra fuera de nivel el cuadricóptero; cuando es demasiado baja el cuadricóptero no será capaz de volver a nivel. En cambio sí es demasiado alta oscilará bruscamente pudiendo dañar el sistema o terminar volteándolo completamente.

La ganancia D se asocia a la velocidad y trabaja en contra de la proporcional e integral. Evita oscilaciones producidas cuando ganancias P e I son demasiado altas, por lo que es un factor de amortiguación. Sí es muy baja entonces el cuadricóptero se desestabilizará cuando P e I sean altas; en cambio, sí es demasiado alta se mantendrá estabilizado; sin embargo, tardará mucho en responder a los mandos.

CAPÍTULO III.- IMPLEMENTACIÓN DE HARDWARE Y SOFTWARE

3.1 DESCRIPCIÓN DEL HARDWARE

Para llevar a cabo el control del cuadricóptero es necesario conocer el funcionamiento de los sensores a utilizar y el tipo de comunicación de cada uno de ellos para elegir correctamente un microcontrolador. Un UAV cuenta con una gran variedad de sensores como altímetros barométricos (Barómetro BMP180), sistemas de posicionamiento global (GPS LS20031), sensores de proximidad ultrasónicos (HC-SR04) e infrarrojos (Sharp gp2y0a02yk) ya sea para evadir obstáculos o para un aterrizaje autónomo, un sensor de medición inercial (ADIS16362) para determinar los ángulos de rotación pitch, yaw, roll y un sistema de comunicación inalámbrica (XBee PRO S3B). El cerebro del cuadricóptero será el microcontrolador (Teensy ++ 2.0), en él se leerán los datos provenientes de los sensores. Estos datos representan las entradas a los controles PID para generar una salida a los motores. La estructura de la tarjeta de control se observa en el siguiente diagrama a bloques (Figura 27).

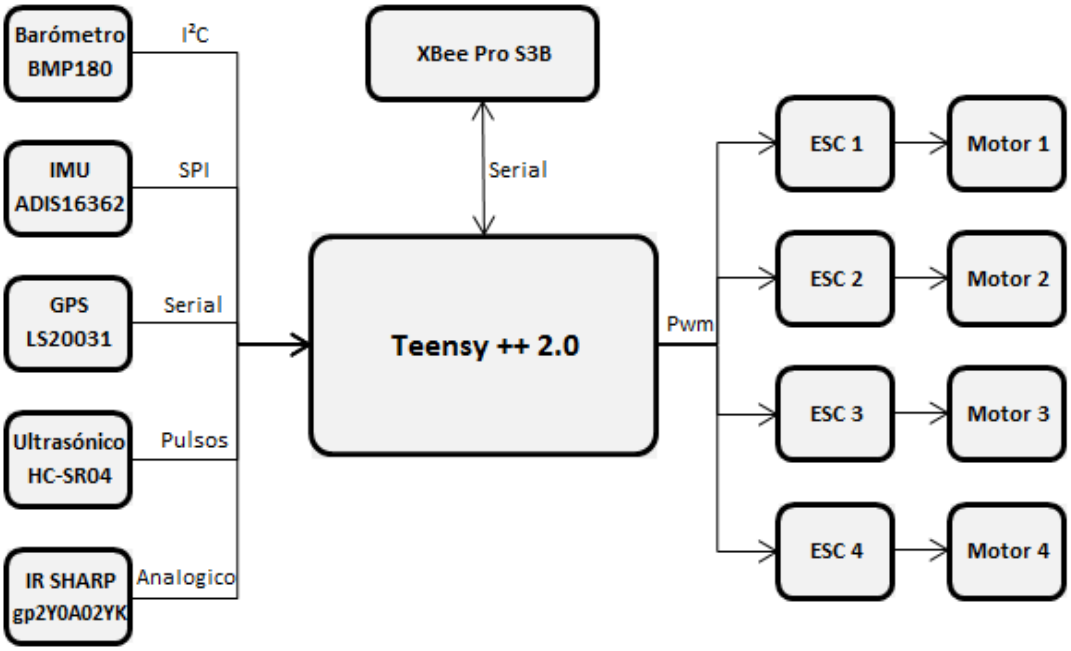


Figura 27.- Diagrama a bloques de los elementos que componen la tarjeta de control.

El marco está hecho a base de fibra de carbono el cual es un material rígido que absorbe las vibraciones producidas por los motores, además de que tiene un peso ligero. El peso total de la nave sin incluir la batería es de aproximadamente 1.13Kg (Figura 28).

Figura 28.- Estructura final del Cuadricóptero.

3.1.1 Controlador Teensy

Se realizaron pruebas con el Teensy ++ 2.0 y Teensy 3.1. En la tabla 3 se presenta una comparación de sus características principales.

Tabla 3.- Tabla comparativa de Teensy ++ 2.0 y Teensy 3.1

Especifición	Teensy ++ 2.0	Teensy 3.1
Procesador	AT90USB1286 8 bits AVR 16 MHZ	MK20DX256 32 bits ARM Cortex - M4 72 MHz
Memoria Flash	130048	262144
Memoria RAM	8192	65536
EEPROM	4096	2048
I / O	46 5 Volts	34 3.3 / 5 Volts
Analog In	8	21
PWM	9	12
UART, I^2C, SPI	1,1,1	3,2,1

3.1.2 Motor *Brushless A2212-13 1000Kv*

Se utilizó el motor *brushless Outrunner* A2212-13 1000Kv (Figura 29) para aeromodelismo con las siguientes características:

- Peso: 52.7g.
- Máximo empuje: 650 – 915g.
- Máxima corriente: 4 – 10 A.
- Potencia máxima: 150 W.
- RPM: 1000 Kv.
- Batería: 2 ~ 4 Celdas / 7.4 ~ 14.8 V.
- Hélices recomendadas: 11 x 7 / 10 x 5.

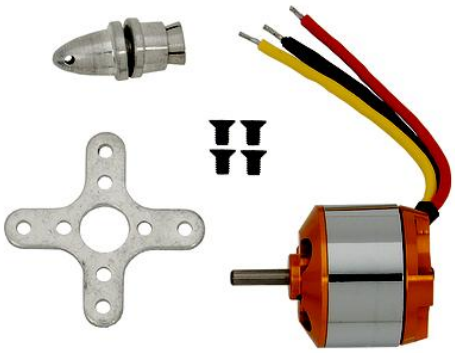


Figura 29.- Motor *brushless* A2212-13 1000 Kv.

3.1.3 Controladores ESC

Como se mencionó con anterioridad los ESC (Figura 30) permiten regular la potencia suministrada a cada motor. El ESC proporcionado es el *GemFan 30 A ESC, Pre-Programmed*. Se alimentan con baterías LiPo de 11.1V.

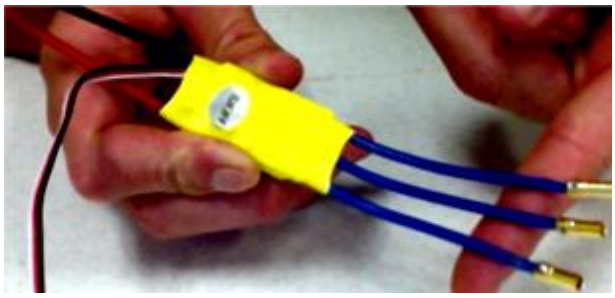


Figura 30.- Controlador de velocidad electrónico ESC implementado.

Algunas características específicas del ESC son:

- Voltaje de entrada: DC 7.4 -11.1 V (2-3S Lixx)
- Corriente: 30A(Salida: 30A continuos, Pico 40A por 10 segundos)
- Tamaño: 48.26 x 25.4x 7.62 mm
- Peso: 27 g.

3.1.4 Baterías Li-Po (Litio – Polímero)

Proporciona una corriente de 3300 mAh. Está formada por 3 celdas que brindan un voltaje de 11.1 VDC (Figura 31). Para su carga se cuenta con el cargador balanceador síncrono Team Tenergy (Figura 32) para no dañar la batería al momento de recargarla ya que el balanceador integrado mantiene una diferencia de carga entre las celdas al mínimo. Entre algunas recomendaciones se aconseja no descargar la batería a más del 10 a 20% de su capacidad total y siempre monitorear la carga de la betería.

Figura 31.- Bateria LiPo ELEV-8.

Figura 32.- Balanceador Team Tenergy.

De las ecuaciones 1, 2 y 3 se obtienen los siguientes resultados para el cálculo del tiempo de vuelo con la batería adquirida.

$$P = 4 * 150W = 600\ W$$

$$E = 1 * 11.1V * 3300mAh = 36.63Wh$$

E = 1 Batería LiPo 11.1V * 3300mAh = 36.63 Wh

Por lo tanto el tiempo de vuelo se puede expresar de la siguiente manera:

$$Tiempo = \frac{36.63\ Wh}{600\ W} * 60\ = 0.06105 * 60 = 3.663\ min$$

3.1.5 Hélices

Propeller 10x4.5 Pusher y Propeller 10x4.5 Slow Flyer (Figura 33). La selección de las hélices se encuentra relacionada por el tipo de motor utilizado. Las implementadas son de 10 x 4.5 (10'' diámetro, 4.5'' paso). La tabla 4 expone las características de cada hélice.

a) b)

Figura 33.- a) Hélice 10x4.5 Slow Flyer y Pusher. b) Hélice Modelo 1055

Tabla 4.- Características generales de las hélices utilizadas

Características	Hélice Pusher 10x4.5	Hélice Slow Flyer 10x4.5	Hélice Modelo 1055	Hélice Modelo 1055
Diámetro	10''	10''	10''	10''
Paso	4.5	4.5	5.5	5.5
RPM máximo	6500	6500	--	--
Giro	CW	CCW	CCW	CCW
Material	Fibra de carbono	Fibra de carbono	Alta gama de fbra de carbono	Alta gama Fibra de carbono

3.1.6 Unidad de Medición Inercial IMU ADIS16362

Para la medición de orientación se optó por el sensor de Analog Devices ADIS16362 que incluye 3 giroscopios y 3 acelerómetros. Este sensor utiliza el protocolo de comunicación SPI; su tamaño es de (23 x 23 x 23 mm) además de que cuenta con un conector flexible. Funciona con un rango de voltaje de entre 4.75 y 5.25 V. A continuación se muestra el sensor y un diagrama a bloques de su estructura interna (Figura 34).

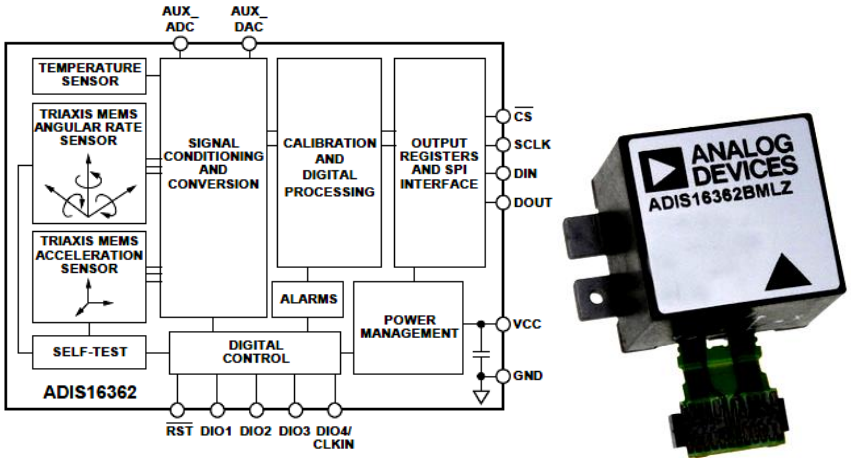


Figura 34.- Estructura interna del ADIS16362.

Cuenta con 24 pines de los cuales solo 6 son utilizados para la comunicación SPI. Se diseñó una tarjeta especial para su montaje en la tarjeta de control. El diseño se llevó a cabo en el software ARES. La siguiente imagen muestra el rutado y placa en 3D (Figura 35).

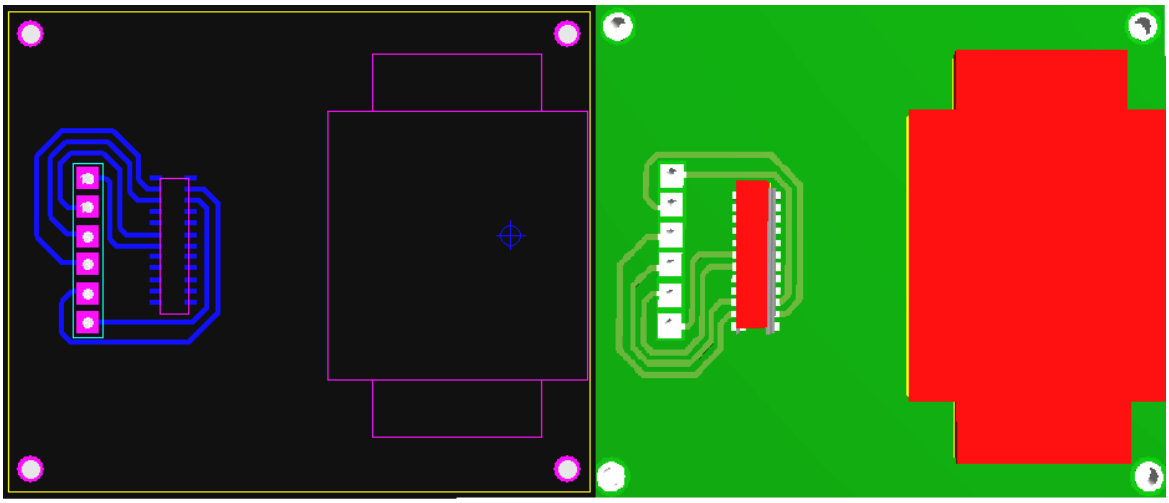


Figura 35.- PCB de la tarjeta para el ADIS16362 y vista en 3D.

SPI por sus siglas en inglés “Serial Peripheral Interface” utiliza líneas de reloj y datos separados, además de una línea de selección para elegir el dispositivo al que se desea comunicar. Los pines para el SPI son SCLK (Señal de reloj para la comunicación síncrona), DOUT (salida de datos del dispositivo), DIN (entrada de datos), CS (línea de selección).

3.1.7 Barómetro

El BMP180 (Figura 36) es un sensor de alta presión barométrica que trabaja con un rango de bajo voltaje (1.8V - 3.6 V) y cuenta con una comunicación tipo I²C, se utiliza para medir la presión absoluta del aire a su alrededor, la cual varía con el clima y la altitud.

Figura 36.- Sensor de presión barométrica BMP180 de Sparkfun.

Ya que utiliza comunicación I²C requiere únicamente de sólo dos cables para su conexión (CL y DA) además de los de alimentación los cuales pueden soportar diversos dispositivos esclavos y maestros para comunicarse con todos los dispositivos del bus. La mayoría de los dispositivos con I²C pueden comunicarse a 100kHz o 400kHz.

3.1.8 GPS LS20031

El receptor LS20031 GPS (Figura 37) incluye una antena integrada y circuitos receptores de GPS, emite información acerca de la posición de un dispositivo 5 veces por segundo, la antena hace un seguimiento de hasta 66 satélites. En la tarjeta se incluye una microbatería con el propósito de preservar los datos del sistema de adquisición de satélites rápida. Cuenta con una comunicación serial Tx-Rx y con un led indicador que parpadea al enlazarse con un satélite, su velocidad de transmisión es de 9600 bps.

Figura 37.- *Pinout* de GPS LS20031.

3.1.9 XBee PRO S3B

Los XBee PRO S3B son dispositivos inalámbricos fabricados por Digi International [9], tienen su propio protocolo de comunicación por radio frecuencia. La versión PRO tiene un mayor alcance (1.6 Km línea vista) con un mayor consumo de potencia. Entre sus principales características se encuentran las siguientes:

- Alcance entre 100 m y 10 km entre sus distintos modelos.
- Potencia de salida del transmisor: 24 dBm (250 MW).
- Voltaje de alimentación: 2.1 a 3.6V.
- Peso: de 5 a 8 g.
- Banda de frecuencia de operación: 902 a 928 MHz.

El XBee PRO S3B consiste en un microcontrolador EFM32G230F128, un emisor de Analog Devices ADF7023 y un amplificador de potencia de RF. Para su configuración es necesario el uso del software X-CTU (Figura 38).

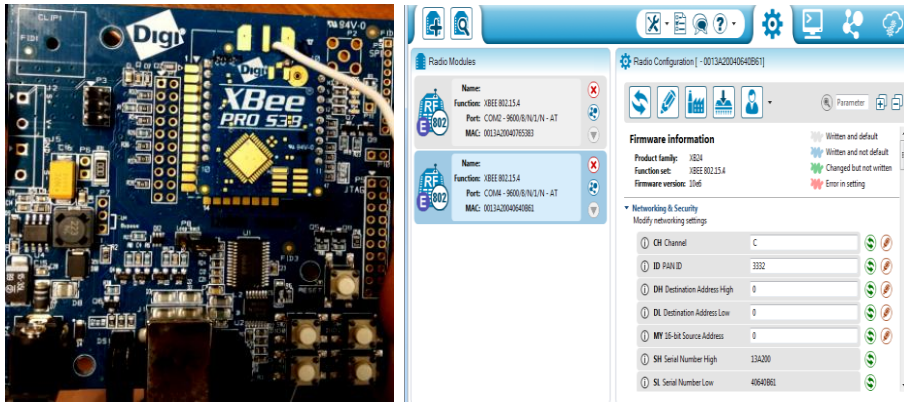


Figura 38.- *Pinout* XBee PRO S3B y XCTU para su configuración.

3.1.10 Sensor ultrasónico HC-SR04

Es un sensor que puede ser utilizado para determinar la distancia en la que se encuentra cierto objeto gracias los pulsos emitidos, el alcance de este sensor va desde los 2cm hasta los 4m.

El HC-SR04 (Figura 39) es un módulo con un emisor y un receptor, su modo de configuración es activar el pin TRIG para que comience a enviar pulsos, una vez enviados se desactiva y el pin ECHO espera el momento en recibir el pulso, debido a que cuenta con un reloj interno se mide el tiempo que tardo el pulso en regresar, por lo que el tiempo es un equivalente de la distancia recorrida. Sus principales características son las siguientes:

- Voltaje de operación: 5V.
- Frecuencia de operación: 40KHz.
- Rango de medición: 2cm - 4m.
- Señal Trigger: 10us min. Pulso TTL.
- Señal Echo: Señal TTL proporcional a la distancia.

Figura 39.- Sensor ultrasónico HC-SR04.

3.1.11 Sensor infrarrojo Sharp GP2Y0A02YK

Es un sensor infrarrojo reflexivo (Figura 40) que proporciona una lectura continua de la distancia medida como una tensión analógica dentro de un rango de 20 a 150 cm. La tensión de alimentación es de 5V y la tensión de salida varía unos 2 volts de diferencia entre el margen mínimo y el máximo de la distancia medida.

El encapsulado presenta una mayor distancia entre la lente y el sensor con el fin de aumentar el rango de trabajo. La conexión se realiza mediante un conector JST de 3 vías, 2 para la alimentación y una para la salida la cual es actualizada cada 39ms. Normalmente

se conecta salida a la entrada de un ADC el cual por medio del microcontrolador es convertido en distancia.

Figura 40.- Sensor infrarrojo reflexivo Sharp gp2Y0A02YK.

3.1.12 Diseño de la placa de control

El diseño de la tarjeta de control se basó en el microcontrolador Teensy ++ 2.0 (Figura 41 y Figura 42). El tamaño final de la tarjeta fue de 9.2 x 9.5 cm la cual va montada en la parte central del cuadricóptero. La tarjeta cuenta con cuatro salidas PWM hacia los motores, pines de conexión para el sensor ultrasónico y el infrarrojo así como dos salidas a tiras de leds. Se implementó un circuito de regulación de voltaje para obtener una salida de 5V y de 3.3V.

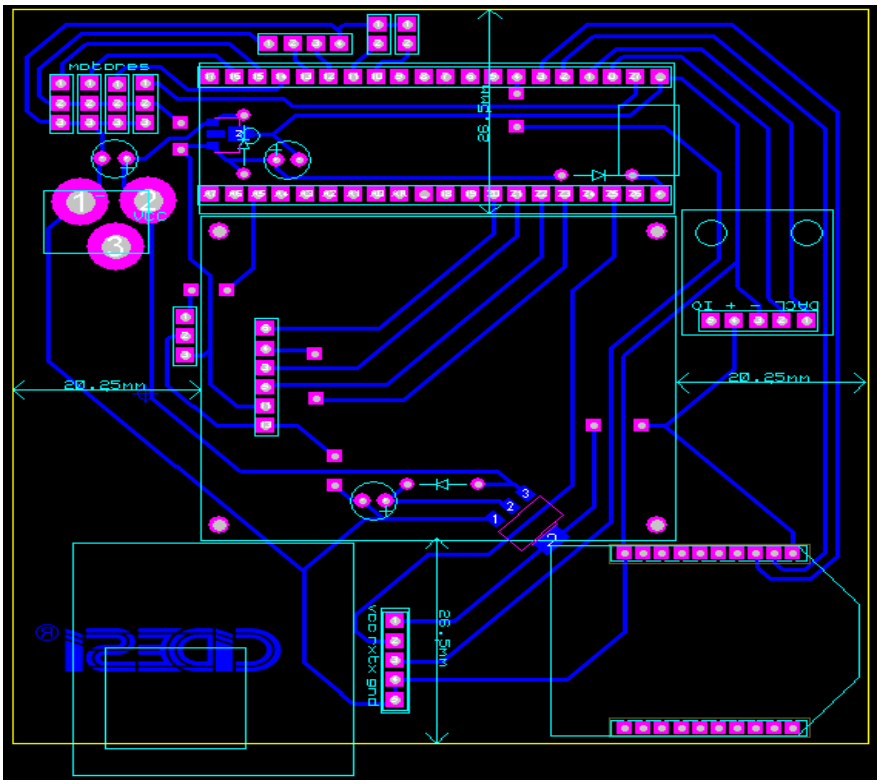


Figura 41.- PCB de la tarjeta de control para el Teensy ++ 2.0.

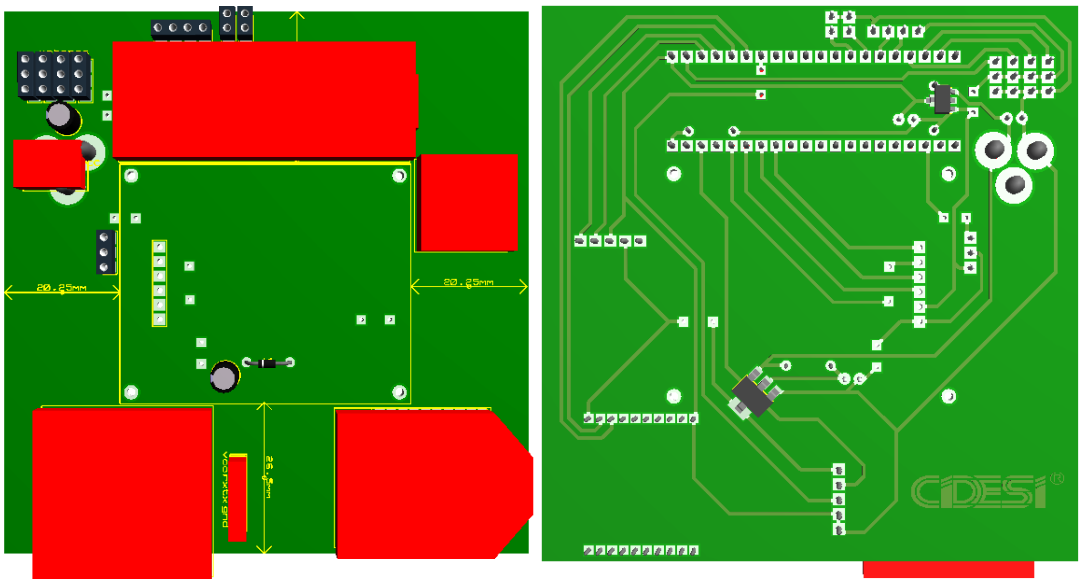


Figura 42.- Diseño con Teensy ++ 2.0 en 3D vista frontal y posterior.

Para el circuito de alimentación se utilizaron dos reguladores de voltaje, el 78L05 y el AMS1117, ambos de montaje superficial (Figura 43). Se agregaron capacitores para realizar un filtrado tanto del voltaje de salida como el de entrada con el objetivo de eliminar los rizos producidos por la señal de entrada. En la Figura 44 se expone el diagrama esquemático del Teensy ++ 2.0 con sus respectivas salidas hacia los sensores.

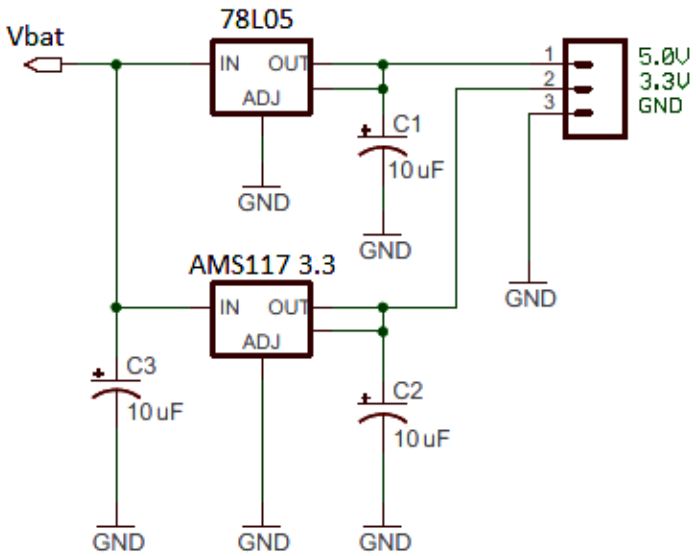


Figura 43.- Diagrama esquemático de la fuente de alimentación para la tarjeta de control.

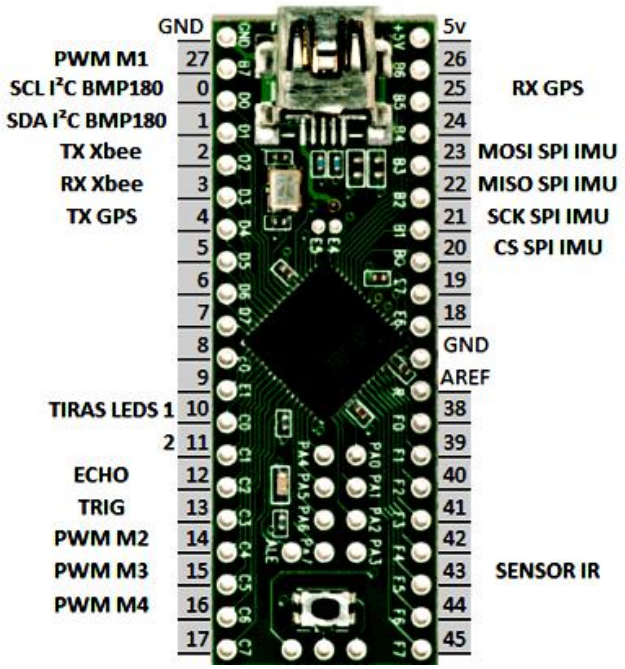


Figura 44.- Esquema de conexión de pines del Teensy ++ 2.0 con los sensores utilizados.

3.2 DESCRIPCIÓN DE SOFTWARE

En esta sección se describirá el código implementado para el funcionamiento de cada sensor así como el control PID, su funcionamiento e implementación con el teensy ++ 2.0.

3.2.1 Teensyduino

Los microcontroladores Teensy cuentan con un micro pulsador para el BootLoader; sin embargo, para poder cargar los programas de forma automática es necesaria la descarga del Teensyduino y del cargador de archivos para cualquier Teensy. A continuación se muestra el cargador de códigos para Teensy así como el instalador del Teensyduino (Figura 45).

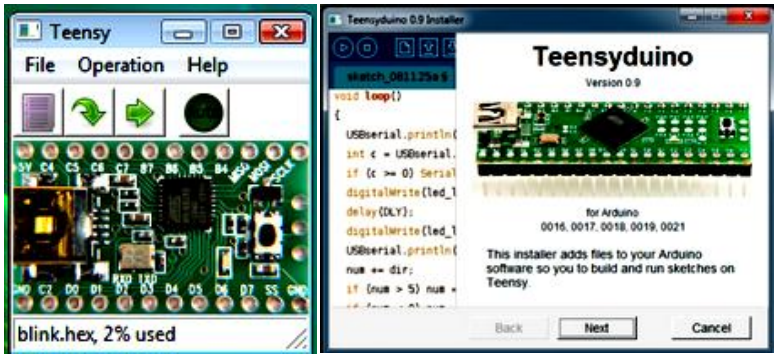


Figura 45.- BootLoader y Teensyduino.

Para la programación se utilizó plataforma de código libre “Arduino” ya que puede ser utilizado para programar el Teensy a través de Teensyduino y el cargador de códigos. El tipo de programación es C lo que lo hace sencillo de usar.

3.2.2 Configuración del GPS LS20031

Para el GPS es necesario determinar el tipo de microcontrolador a utilizar, en el caso de ser el Teensy 3.1 sólo basta con declarar el número de serial a usar ya que cuenta con 3 seriales extra. Por otro lado si se utiliza el Teensy ++ 2.0 con un Serial extra (en este caso se utilizará para la comunicación con XBee) es necesario convertir dos pines digitales para trabajar como seriales por medio de la librería “AltSoftSerial”. La configuración para cada microcontrolador se expondrá en la sección de anexos.

3.2.3 Configuración de la IMU ADIS16362

Se incluye la librería SPI para la comunicación con el sensor. Para la lectura de datos es necesaria la hoja de especificaciones (*Datasheet*) ya que en él se especifican las direcciones de la salida para el giroscopio y el acelerómetro en cada eje. Se implementaron dos filtros, uno de ellos fue el complementario para la obtención de los ángulos y por otro lado se configuró el filtro interno del sensor.

Para el Teensy ++ 2.0 no existe error alguno con la comunicación por SPI; sin embargo, se descubrió que el Teensy 3.1 tiene problemas en la transmisión de datos por SPI. Por lo que durante el resto del proyecto se optó por utilizar el microcontrolador Teensy ++ 2.0.

3.2.4 Configuración del *XBee PRO*S3B

Se realizó una prueba de control de motores por medio del XBee PROS3B. Para la prueba se realizaron cambios en la configuración del XBee por medio del programa XCTU. Los cambios se realizaron en el ID, DT y MY (Figura 46).

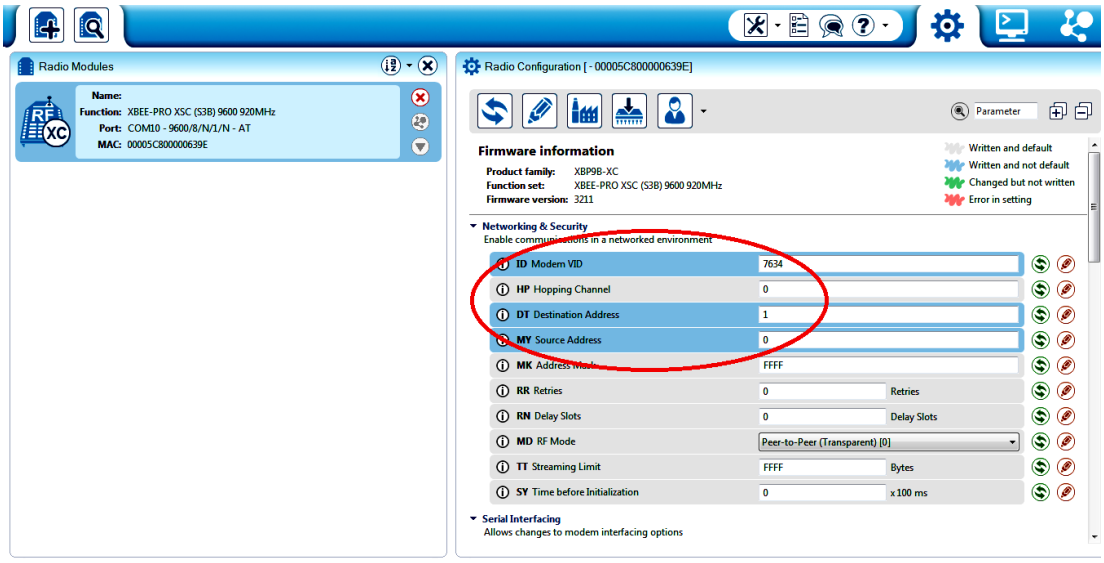


Figura 46.- Configuración para el XBee PRO S3B a través del XCTU.

3.2.5 Controlador PID

Debido a que el control del cuadricóptero tiene 6 grados de libertad (Roll θ, Pitch Φ, Yaw Ψ, X, Y, Z) donde X, Y, Z son la posición del centro de masa del vehículo con respecto a la tierra y θ, Φ, Ψ son los ángulos de Euler que representan la postura se puede llevar a cabo un control PID para postura y posición deseada de forma autónoma (Figura 47).

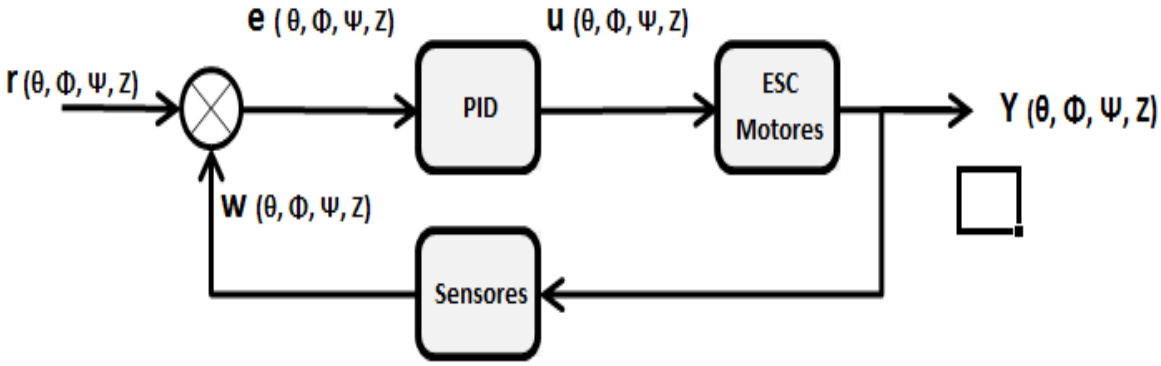


Figura 47.- Estructura PID para los ángulos de Euler y altura.

En donde:

- r(θ, Φ, Ψ, Z) son los datos deseados.
- w(θ, Φ, Ψ, Z) son las mediciones de los sensores.
- e(θ, Φ, Ψ, Z) son los errores adquiridos de la resta de r – w.

- u(θ, Φ, Ψ, Z) son las salidas de cada PID las cuales actúan sobre los motores.
- y(θ, Φ, Ψ, Z) son los datos generados por el cambio en la potencia de los motores.

Es necesario implementar tres controladores PID para cada ángulo a controlar; sin embargo, para el control de la altura es necesario variar la velocidad de todos los motores con la misma magnitud.

3.2.5.1 Control de Roll

Roll es la rotación sobre el eje "X", lo que corresponde a un movimiento izquierda a derecha. El control de Roll implica el incremento/decremento de dos motores, para este caso los motores 1 y 3. Por ejemplo un giro negativo es el resultado de un incremento en la velocidad del lado derecho del cuadricóptero, para modificarlo se varía la velocidad en uno de los dos motores. Cabe mencionar que el valor de altura es un valor fijo ya que aún no se toma en cuenta el PID para controlar la altura del cuadricóptero.

$$Motor\ 1 = altura - Roll \quad \textbf{(ec. 9)}$$

$$Motor\ 3 = altura + Roll \quad \textbf{(ec. 10)}$$

3.2.5.2 Control de Pitch

Pitch es la rotación sobre el eje "Y", lo que corresponde a un movimiento hacia al frente y hacia atrás. Para este cuadricóptero los motores que actúan en su sentido de giro son los motores 2 y 4. Por ejemplo, para elevar la parte de atrás es necesario aumentar la velocidad en el motor 4 y disminuir en igual forma al motor 2 y viceversa para el giro positivo.

$$Motor\ 2 = altura - Pitch \quad \textbf{(ec. 11)}$$

$$Motor\ 4 = altura + Pitch \quad \textbf{(ec. 12)}$$

3.2.5.3 Control de Yaw

Yaw es la rotación producida en el eje "Z". Es el único ángulo en donde es necesario el control de los 4 motores para su estabilización, ya que es requerido aumentar o disminuir en igual forma un par de motores para producir un cambio de giro. Por ejemplo, aumentar

la velocidad de los motores 1 y 3 y disminuirla en igual forma en los motores 2 y 4 produce un sentido de giro positivo y viceversa.

$$Motor\ 1 = altura + Yaw \quad \textbf{(ec. 13)}$$

$$Motor\ 2 = altura - Yaw \quad \textbf{(ec. 14)}$$

$$Motor\ 3 = altura + Yaw \quad \textbf{(ec. 15)}$$

$$Motor\ 4 = altura - Yaw \quad \textbf{(ec. 16)}$$

La ecuación para la salida a cada motor queda de la siguiente manera:

$$Motor\ 1 = altura + Roll + Yaw \quad \textbf{(ec. 17)}$$

$$Motor\ 2 = altura + Pitch - Yaw \quad \textbf{(ec. 18)}$$

$$Motor\ 3 = altura - Roll + Yaw \quad \textbf{(ec. 19)}$$

$$Motor\ 4 = altura - Pitch - Yaw \quad \textbf{(ec. 20)}$$

Se implementó el PID de dos maneras diferentes (utilizando la biblioteca de Arduino y otra basada en literatura consultada [10]). A continuación se muestra un ejemplo del PID según la literatura el cual es el mismo para cada ángulo (Figura 48) y el utilizado a través de la librería de Arduino (Figura 49).

```
//PID roll
roll2 = ang_x;
if (ang_x > 0) roll2 -= 179;
if (ang_x < 0) roll2 += 179;
 err = roll2;
 inr = (err + errl)/2;
inr2 = inr2 + inr;
 der = (err - errl);
errl = err;
salr = kpr*err + kir*inr2 + kdr*der;
```

Figura 48.- PID para el ángulo Roll

```
double kp = 0.4, ki = 0.01 , kd = 0.01;
#define PID_ROLLMM 20
double sp_roll = 0.0;
double err = 0.0;
double salr = 0.0;

PID roll_cont(&err, &salr, &sp_roll, kp, ki, kd, REVERSE);

imu.getRPY(rpy);
err = rpy[0];
if (rpy[0] > 0)  err = sp_roll - (err - 179);
if (rpy[0] < 0)  err = sp_roll - (err + 179);

roll_cont.Compute();
```

Figura 49.- PID para el ángulo roll a través de librería de arduino.

CAPITULO IV.- PRUEBAS

Se midieron las revoluciones por minuto (rpm) de cada motor para encontrar los valores de PWM mínimo y máximo aplicado a cada motor. Se colocó una cinta negra en cada motor como punto de referencia y se utilizó un sensor infrarrojo CNY70 (Figura 50) con la configuración ilustrada en la Figura 51.

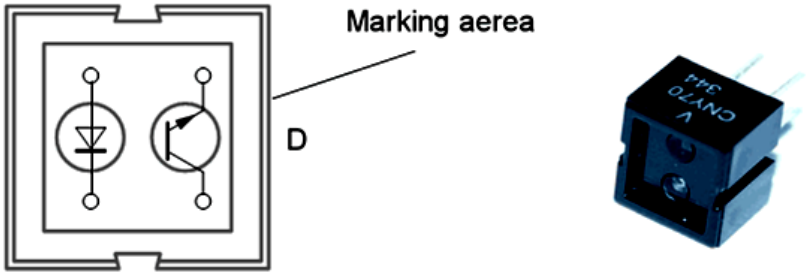


Figura 50.- Diagrama esquemático del sensor infrarrojo CNY70.

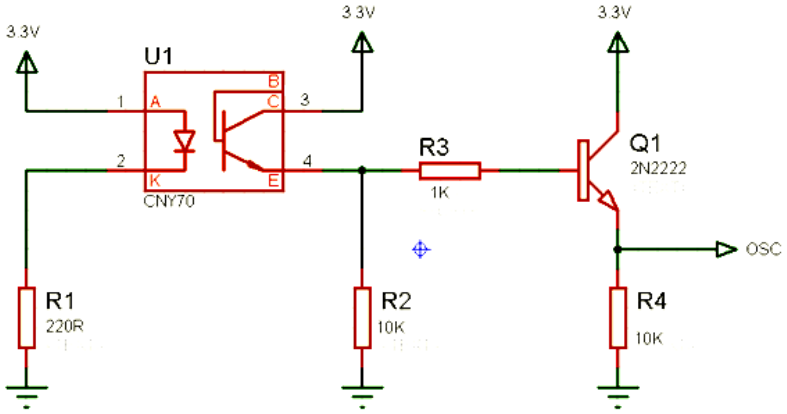


Figura 51.- Diagrama de conexión para la medición de PWM en cada motor.

En las figuras 52 y 53 se muestra el montaje y la alineación del sensor infrarrojo para la medición de rpm. Se utilizó un trozo de hule espuma para sostener al sensor mientras se incrementa el PWM en el motor, ya que si no se encontraba debidamente sostenido el sensor se desalineaba, teniendo respuestas de medición erróneas.

Figura 52.- Alineación del sensor CNY70 en la medición de PWM.

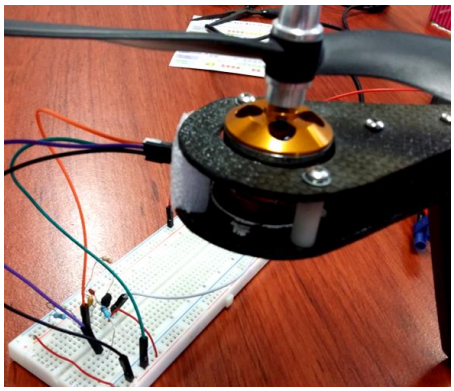


Figura 53.- Alineación del sensor CNY70 en la medición de PWM.

Las señales obtenidas del sensor infrarrojo fueron analizadas a través de un osciloscopio digital (Figura 54 y Figura 55). A través de él fueron tomados los valores de frecuencia para obtener el tiempo necesario para una vuelta y el número de rpm de manera confiable para cada motor. Los resultados obtenidos de las mediciones se muestran en la tabla 4. La gráfica generada donde se expone la relación PWM/Frecuencia se mostrará en la sección de resultados.

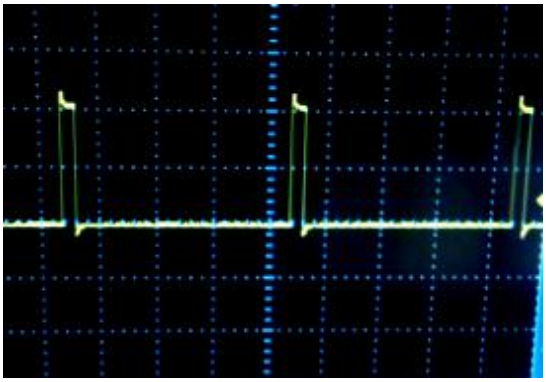


Figura 54.- Detección de la línea negra para la obtención del tiempo de una vuelta.

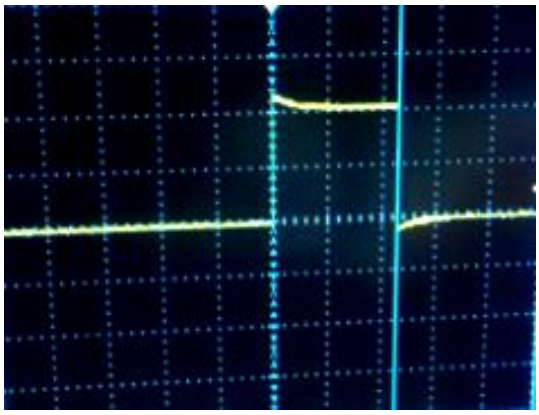


Figura 55.- Señal PWM producida por el microcontrolador con una frecuencia de aproximadamente 4KHz.

Tabla 5.- Obtención de valores mínimos y máximos de PWM y número de vueltas.

PWM	Tiempo PWM (ms)	Tiempo 1 vuelta (ms)				Núm. Vueltas 1 segundo				Núm. Vueltas 1 minuto			
		M1	M2	M3	M4	M1	M2	M3	M4	M1	M2	M3	M4
140	1	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------
141	1.14	50.0	43.6	50.0	52.0	20.0	22.9	20.0	19.2	1200.0	1376.1	1200.0	1153.8
145	1.16	27.6	27.6	25.6	27.6	36.2	36.2	39.1	36.2	2173.9	2173.9	2343.8	2173.9
150	1.2	19.2	17.8	18.8	19.2	52.1	56.2	53.2	52.1	3125.0	3370.8	3191.5	3125.0
155	1.24	15.8	15.0	15.2	15.6	63.3	66.7	65.8	64.1	3797.5	4000.0	3947.4	3846.2
160	1.28	13.6	12.8	13.2	13.6	73.5	78.1	75.8	73.5	4411.8	4687.5	4545.5	4411.8
165	1.32	11.5	11.3	11.0	11.4	87.0	88.5	90.9	87.7	5217.4	5309.7	5454.5	5263.2
170	1.36	10.1	9.9	9.8	10.0	99.0	101.0	102.0	100.0	5940.6	6060.6	6122.4	6000.0
175	1.4	9.2	8.9	8.8	9.2	108.7	112.4	113.6	108.7	6521.7	6741.6	6818.2	6521.7
180	1.44	8.5	8.2	8.2	8.4	117.6	122.0	122.0	119.0	7058.8	7317.1	7317.1	7142.9
185	1.48	8.1	7.8	8.0	8.0	123.5	128.2	125.0	125.0	7407.4	7692.3	7500.0	7500.0
190	1.52	8.1	7.8	8.1	8.0	123.5	128.2	123.5	125.0	7407.4	7692.3	7407.4	7500.0
195	1.56	8.1	7.9	8.0	8.2	123.5	126.6	125.0	122.0	7407.4	7594.9	7500.0	7317.1
200	1.6	8.1	7.9	8.0	8.2	123.5	126.6	125.0	122.0	7407.4	7594.9	7500.0	7317.1
205	1.64	8.1	7.8	8.0	8.2	123.5	128.2	125.0	122.0	7407.4	7692.3	7500.0	7317.1
210	1.68	8.1	7.8	8.0	8.2	123.5	128.2	125.0	122.0	7407.4	7692.3	7500.0	7317.1
215	1.72	8.2	7.9	8.0	8.2	122.0	126.6	125.0	122.0	7317.1	7594.9	7500.0	7317.1
220	1.76	8.1	7.9	8.0	8.2	123.5	126.6	125.0	122.0	7407.4	7594.9	7500.0	7317.1
225	1.8	8.2	7.9	8.0	8.2	122.0	126.6	125.0	122.0	7317.1	7594.9	7500.0	7317.1
230	1.84	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

De la tabla 5 se puede observar que para valores de PWM menores a 140 y mayores a 230 los ESC entran en saturación, por lo que no se genera movimiento en los motores. Los valores de PWM que se pueden aplicar a los motores deben de encontrarse dentro del rango de 141 – 225, donde 141 es el valor mínimo y 225 el valor máximo.

Una vez realizado el diseño en PCB de toda la placa se llevó a cabo la construcción de ésta por medio de una máquina CNC. En las siguientes figuras se expone la placa final así como la vista posterior (Figura 56 y Figura 57).

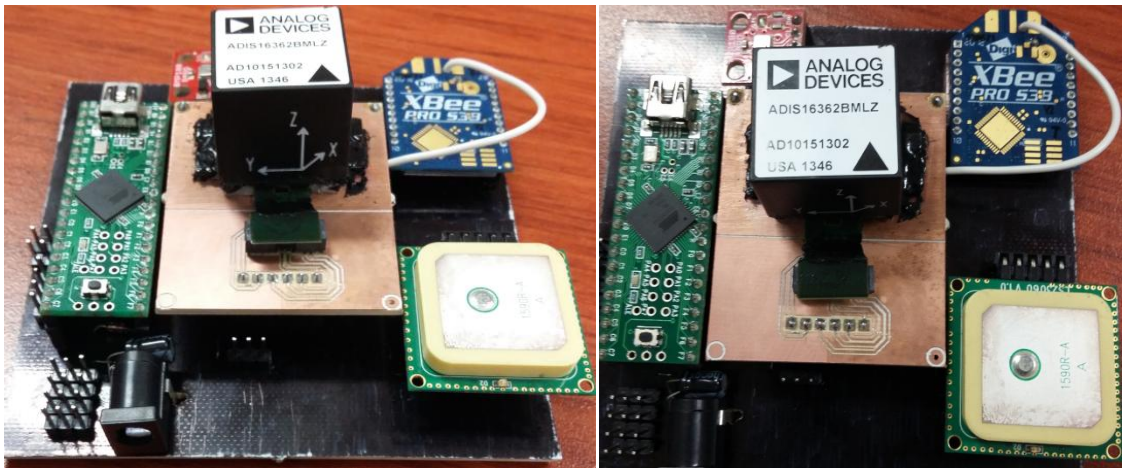


Figura 56.- Placa terminada con los sensores montados.

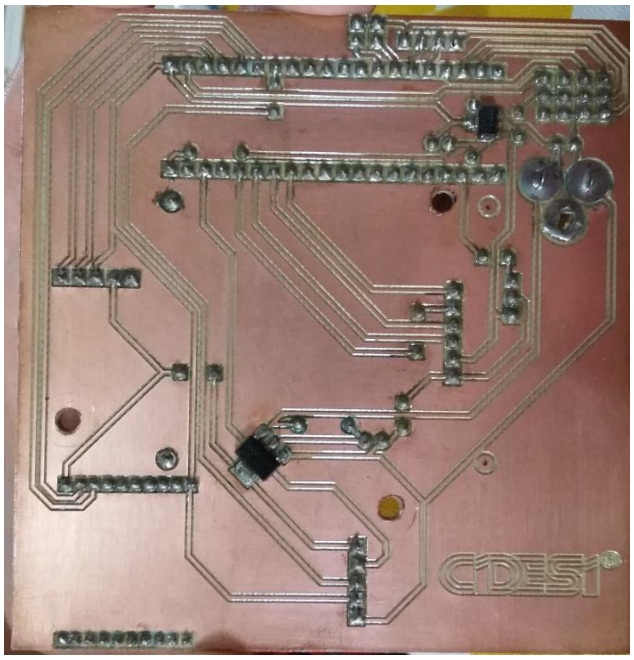


Figura 57.- Vista posterior de la placa terminada.

La placa cuenta con un conector *jack power* y dos reguladores (5V y 3.3V) cuyo voltaje de entrada máxima es de 15V, proporcionando una corriente de salida máxima de 1.5A. En la tabla 6 se muestran los voltajes de alimentación de cada dispositivo.

Tabla 6.- Voltajes de alimentación para cada dispositivo

Componente	5V	3.3V
ADIS 16362	X	
BMP180		X
GPS LS20031		X
Infrarrojo gp2Y0A02YK	X	
Teensy ++ 2.0	X	
Ultrasónico HC-SR04	X	
Xbee		X

Se realizó una prueba general de la placa donde se llevó a cabo la lectura de cada sensor al mismo tiempo para verificar la funcionalidad de ésta. Se obtuvieron las siguientes mediciones mostradas en la Figura 58. Cabe mencionar que para que el GPS envíe datos de posición es necesario que se conecte con al menos 3 satélites y que preferentemente se encuentre al aire libre.

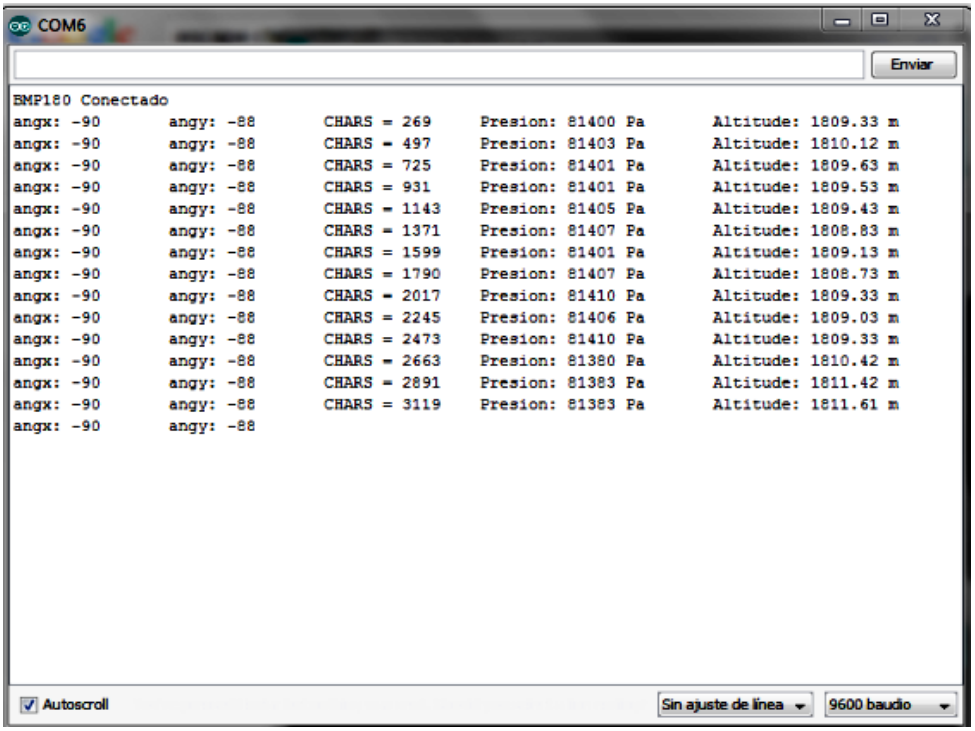


Figura 58.- Lectura de todos los sensores utilizados a través de la tarjeta de control.

CAPITULO V.- RESULTADOS

Se realizó la caracterización de los motores para encontrar una relación rpm entre cada uno de los motores. Esto con el propósito de que la velocidad de los motores sea cercada al momento de implementar el PID. De la caracterización realizada se obtuvieron dos gráficas, las cuales muestran la relación PWM/F (Figura 59) y PWM/rpm (Figura 60).

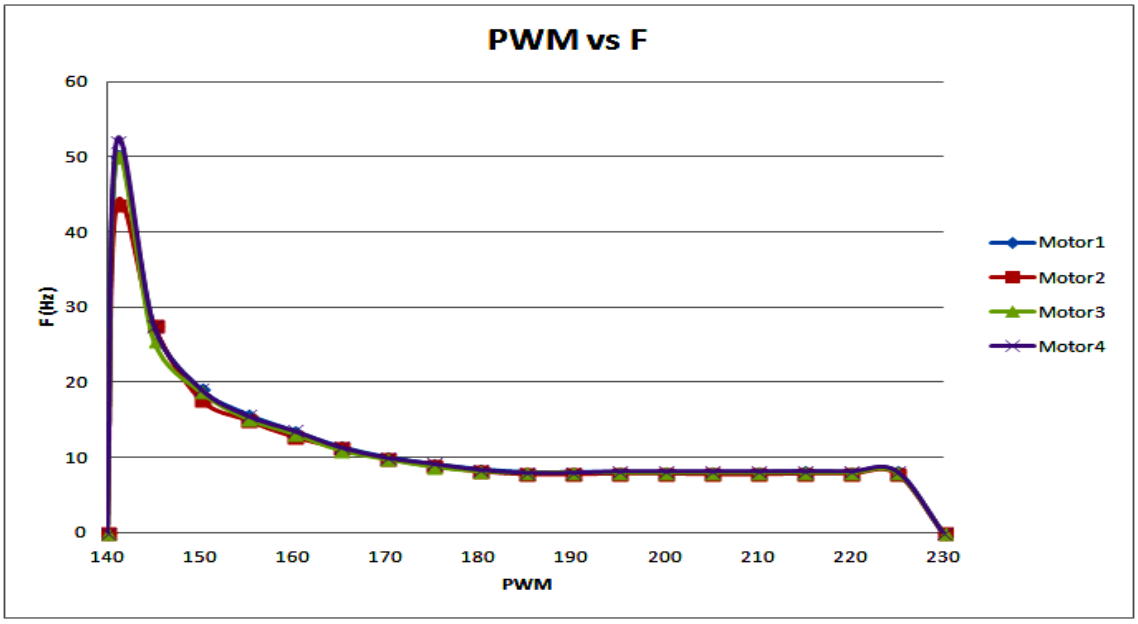


Figura 59.- Relación PWM vs F.

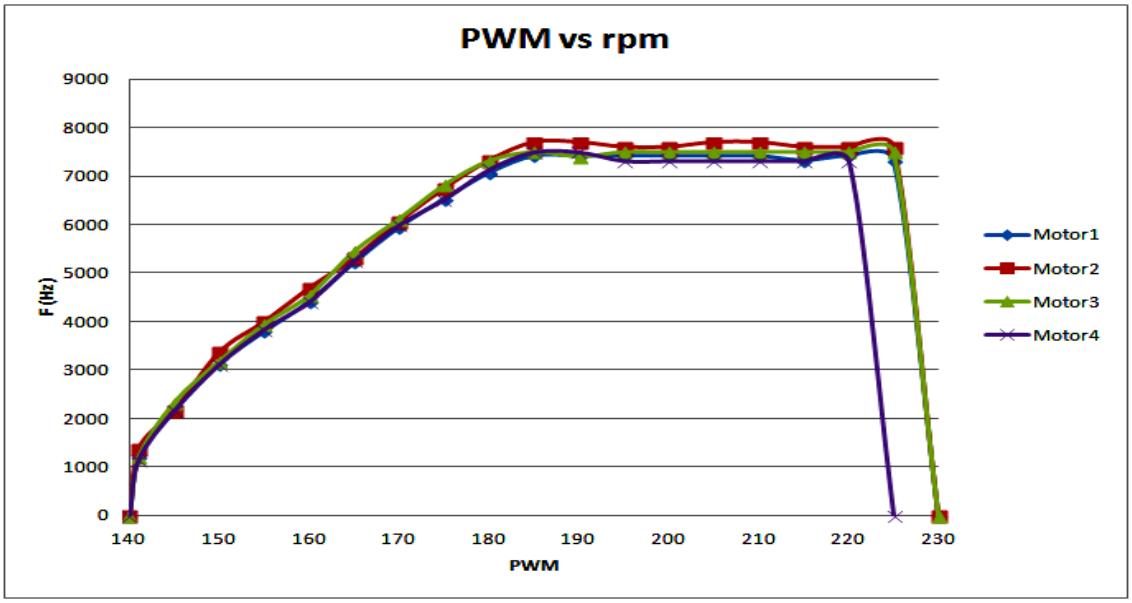


Figura 60.- Relación PWM vs rpm.

Se diseñó y construyó una estructura para realizar pruebas de velocidad de los motores. Dicha estructura cuenta con un tubo de aluminio y una rótula para sostener el cuadricóptero y que permite la movilidad de éste (Figura 61).

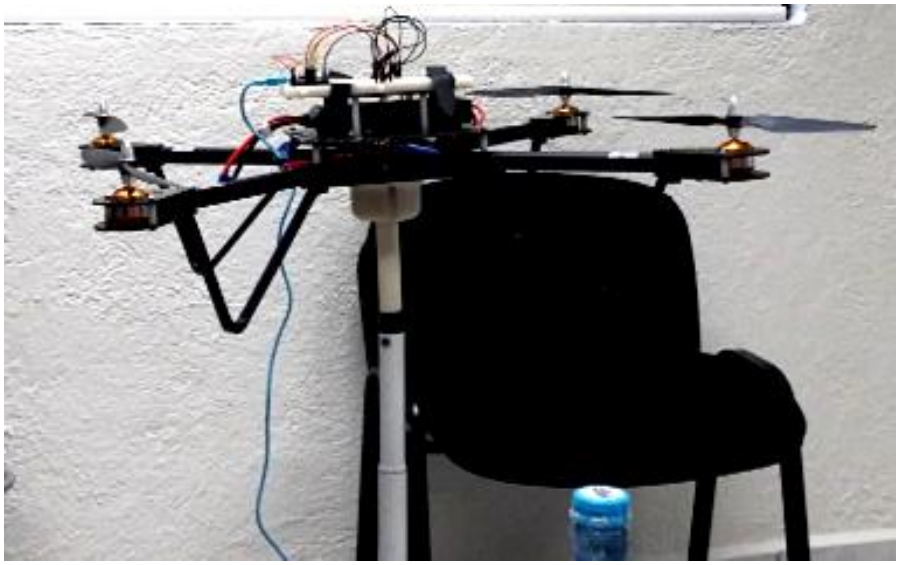


Figura 61.- Cuadricóptero montado sobre estructura de aluminio.

En las siguientes figuras se muestran las pruebas realizadas con el PID para la estabilización del cuadricóptero (Figura 62 y Figura 63).

Figura 62.- Estabilización del cuadricóptero por medio de PID.

Figura 63.- Estabilización del cuadricóptero por PID desde otro ángulo.

Cabe mencionar que la estructura de aluminio no era lo suficientemente fuerte para soportar todo el empuje generado por los motores. Debido a esto se rediseñó la estructura sustituyendo el tubo de aluminio por uno de acero (Figura 64).

Figura 64.- Estructura de acero galvanizado para pruebas con cuadricóptero.

Se reemplazaron las aspas de fibra de carbono por aspas de plástico de 11 x 5.5 (Figura 65) donde se observó una mayor fuerza de empuje en cada una de las extremidades del cuadricóptero, por lo que no se utilizó toda la potencia en los motores.

Figura 65.- Pruebas de velocidad con aspas (PONER MODELO DE ASPA).

Con las aspas nuevas se realizaron pruebas de estabilidad y velocidad de los motores para la variación del ángulo Roll modificando las constantes KP, KI, KD del PID (Figura 66).

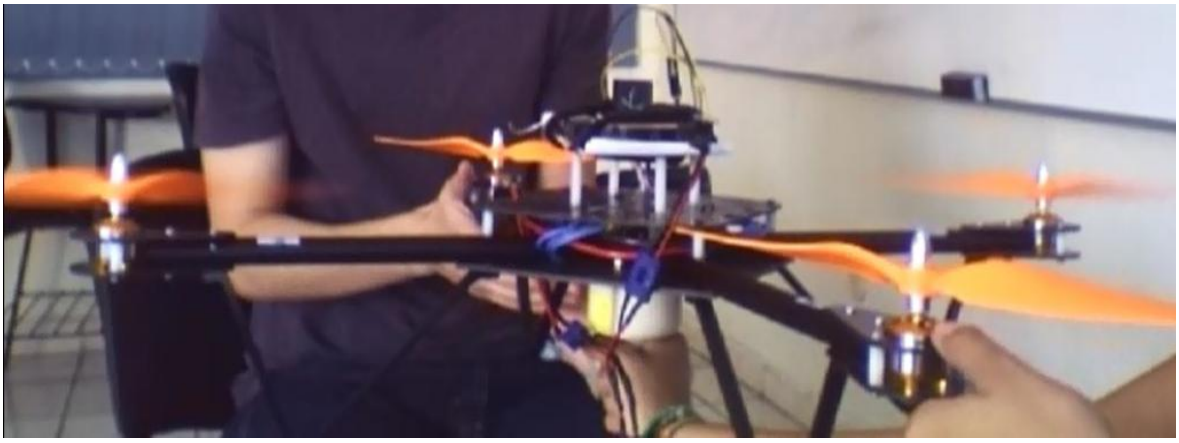


Figura 66.- Pruebas de estabilidad y velocidad para el ángulo Roll.

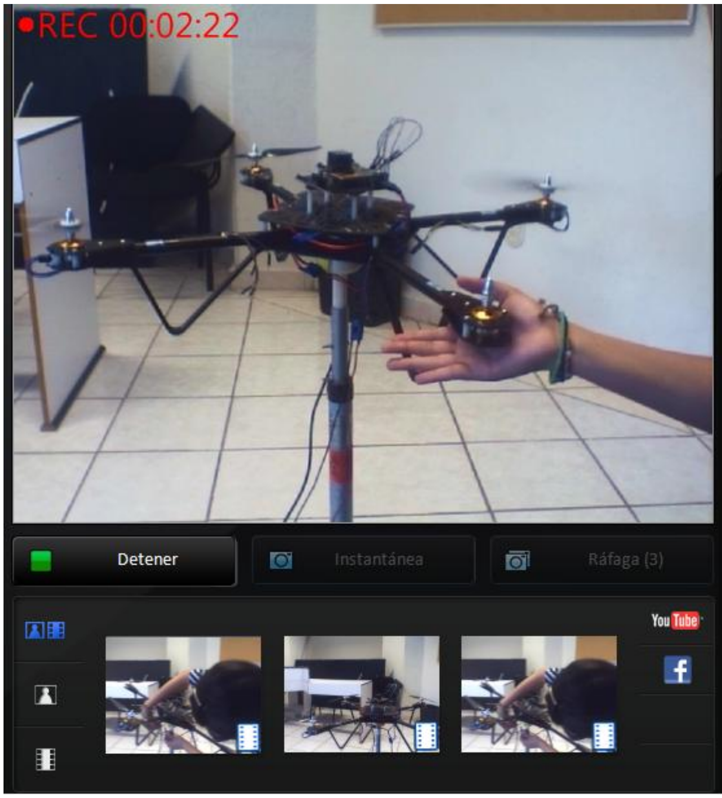
También se realizaron pruebas con aspas nuevas de fibra de carbono de 10 x 5.5 donde se observo mayor estabilidad al momento de implementar el PID. Los valores de

KP, KI y KD probados son los expuestos en la tabla 7 donde la prueba 8 mostró mejores resultados que todos los anteriores.

Tabla 7.- Observaciones de constantes de PID

P	KP	KI	KD	Observaciones
1	0.2	0	0	El control es muy lento. El aumento de velocidad de los motores es muy lento al inclinar por completo el cuadricóptero.
2	0.5	0	0	El control proporcional funciona correctamente. La salida del PID no incrementa conforme se mantiene el error. Cuando esta en equilibrio no permanece justamente en el centro.
3	0.4	0	0	Control muy lento. M3 no generá suficiente fuerza de empuje. Sí se voltea hacia M3 no logra levantarlo todo. La salida del PID sigue sin aumentar.
4	0.5	0.001	0	La salida del PID ya aumenta con forme permanece el error; sin embargo es muy lenta. Se mantiene más tiempo en equilibrio.
5	0.5	0.01	0	Reacciona bastante bien, el PID reacciona rápido. Se mantiene mayor tiempo en equilibrio. M1 y M3 reaccionan bien ante el error.
6	0.5	0.01	0.09	Pierde el control muy rápido, disminuir kd.
7	0.5	0.01	0.05	Se estabiliza pero de repente pierde el control. disminuir kd un poco más.
8	0.5	0.01	0.01	Responde muy bien, se le agregó una perturbación mínima. Permanece mayor tiempo en equilibrio pero en un momento lo pierde.

En las figuras 67 y 68 se ilustran las pruebas de estabilidad realizadas a diferentes valores de KP, KI y KD.



-179.00	0.00	0.00	0.92	69.08	67.92
-179.00	0.00	0.00	0.92	69.08	67.92
-179.00	0.00	0.00	0.92	69.08	67.92
-178.00	-1.00	0.00	0.52	69.48	67.52
-178.00	-1.00	0.00	0.52	69.48	67.52
-177.00	-2.00	0.00	0.12	69.88	67.12
-178.00	-1.00	0.00	0.52	69.48	67.51
-179.00	0.00	0.00	0.92	69.08	67.91
178.00	1.00	0.00	1.32	68.68	68.32
178.00	1.00	0.00	1.32	68.68	68.32
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
-179.00	0.00	0.00	0.92	69.08	67.92
178.00	1.00	0.00	1.32	68.68	68.32
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
-179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92
179.00	0.00	0.00	0.92	69.08	67.92

Figura 67.- Control PID con Kp = 0.5, Ki = 0.01 y Kd = 0.0.

Figura 68.- Control PID con Kp = 0.5, Ki = 0.01 y Kd = 0.01.

CAPITULO VI.- CONCLUSIONES

Un cuadricóptero es un vehículo aéreo no tripulado capaz de realizar tareas específicas modificando las velocidades de giro de los motores. Debido a que ya se contaba con la estructura se realizó la caracterización de cada uno de los motores. Esto con el propósito de que las velocidades en cada motor sean relativamente iguales para evitar complicaciones al implementar el control. El PID para cada ángulo se programó de tal forma que aumenta o disminuye por igual la velocidad de los motores que actúan en él.

En función a las pruebas realizadas con aspas de fibra de carbono y de plástico se encontró que las de fibra de carbono presentan mayor maniobrabilidad debido a que el diámetro y paso que presentan son menores a las de plástico. Por lo que generan menor fuerza de empuje y consumen menor cantidad de corriente. Esto permite que su manipulación sea más sencilla.

En cuanto al microcontrolador se encontró que el Teensy 3.1 tiene dificultades de comunicación por SPI, los cuales aún no han sido resueltos por el fabricante por lo que se sustituyó por el Teensy ++ 2.0. Como se necesitaba un puerto serial extra con el Teensy ++ 2.0 se recurrió al uso de la biblioteca “AltSoftSerial”, el cual simula un puerto serial a través de pines digitales específicos Rx, Tx (4, 25 para el Teensy ++ 2.0). Por lo que es necesario conocer el pin a utilizar para cada microcontrolador.

Existen otras mejoras para un control PID, los cuales se definen como PID con ponderación de la referencia y PID incremental con ponderación de la referencia. El primero ofrece una respuesta más rápida sin amplificar el ruido en el sistema mientras que el segundo es una modificación del anterior que permite limitar los incrementos para evitar cambios bruscos en la respuesta del sistema limitando la saturación de éste. O bien la implementación de controladores LQI para corregir las perturbaciones del sistema o LQR para proporcionar una respuesta robusta para errores grandes.

El control PID para éste caso funciona bien con rangos de error pequeños (-10 y 10 grados). No obstante para valores superiores el control de la nave se vuelve inestable. El control realizado se limitó a la estabilización de Roll donde actúan los motores 1 y 3. Ante perturbaciones el cuadricóptero mantuvo la estabilidad por un tiempo; sin embargo, llega a perderla y comienza a oscilar.

Cabe mencionar que a pesar de que no se logró la estabilidad total del cuadricóptero por falta de tiempo, análisis y estudios se adquirieron y reforzaron nuevos conocimientos en cuanto a programación. Por otro lado se puede concluir que el control total de un cuadricóptero resultó ser algo complejo por todas las variables y el nivel de programación que requiere. Es importante mencionar las precauciones que se deben de considerar al realizar las pruebas ya que si la estructura del cuadricóptero es débil o las aspas no cuentan con protección propia puede llegar a ser peligroso para el usuario.

TRABAJO A FUTURO

El trabajo presentado en este reporte puede ser mejorado en muchos aspectos, los puntos más importantes a mejorar son:

- Realizar pruebas de estabilidad y de vuelo con diferentes tipos de control, recomendando el PID con ponderación a referencia, PID integral con ponderación a referencia o bien el LQR.
- Lograr el desplazamiento del cuadricóptero controlando la posición por medio de un sistema GPS, el cual se encuentra integrado en la tarjeta de control.
- Desarrollar el control para el aterrizaje y despegue vertical.
- Construir un control remoto para establecer el tipo de vuelo o determinar la posición deseada del cuadricóptero.
- Implementar en la base del cuadricóptero una cámara de alta definición para aplicaciones de fotografía aérea.
- Aislar las vibraciones de los motores de la IMU ya que el giroscopio y el acelerómetro integrado son muy sensibles a éstas lo que ocasiona errores de medición al momento de implementar el control.

ANEXOS

A continuación se ilustra el *pinout* del Teensy ++ 2.0 (Figura 69) y el Teensy 3.1 (Figura 70) con el propósito de conocer las características de cada microcontrolador.

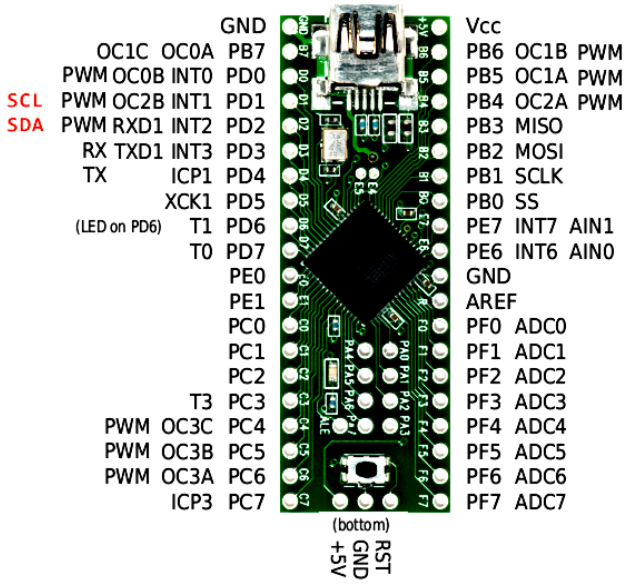


Figura 69.- *Pinout* del Teensy ++ 2.0.

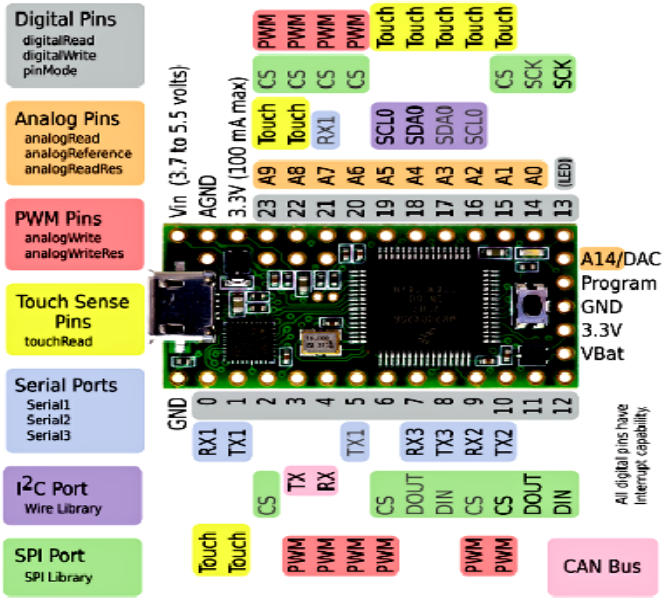


Figura 70.- *Pinout* del Teensy 3.1

Inicialmente se utilizó el Teensy 3.1 y se llevó a cabo el diseño y construcción de una tarjeta de control donde se realizaron pruebas con los sensores y de PWM; sin embargo,

se encontraron algunos problemas al momento de transmitir datos por SPI. En la siguiente figura se muestra el diseño de la placa (Figura 71) y su vista en 3D (Figura 72).

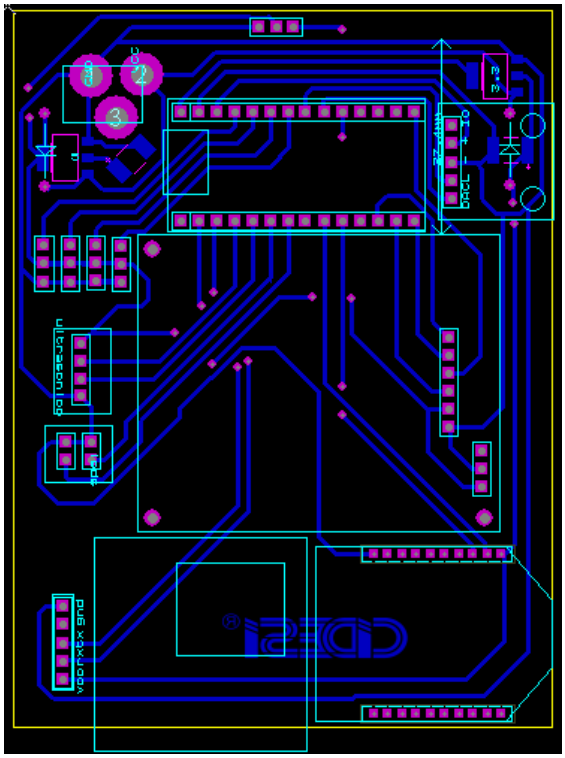


Figura 71.- Diseño de la tarjeta de control para el Teensy 3.1

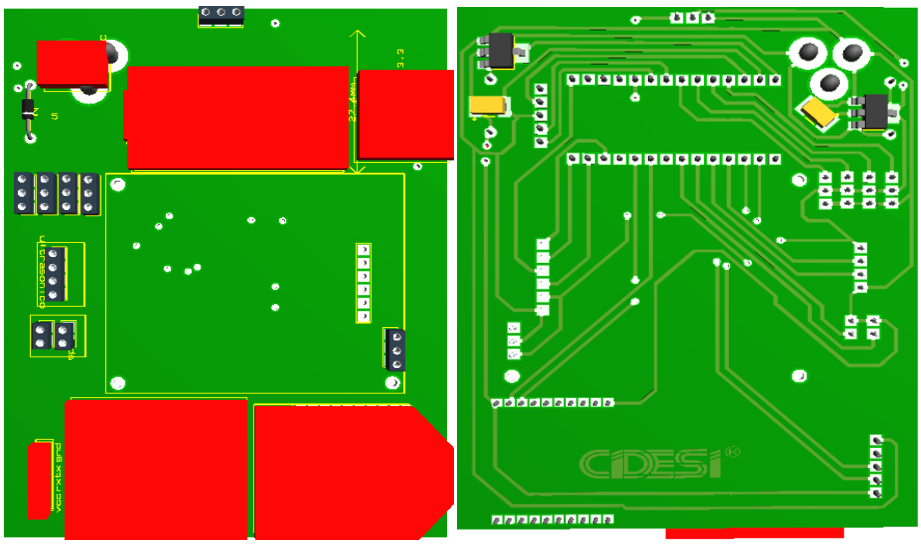


Figura 72.- Diseño con Teensy 3.1 en 3D vista frontal y posterior.

La distribución de pines para los sensores con el Teensy 3.1 se muestra en la Figura 73.

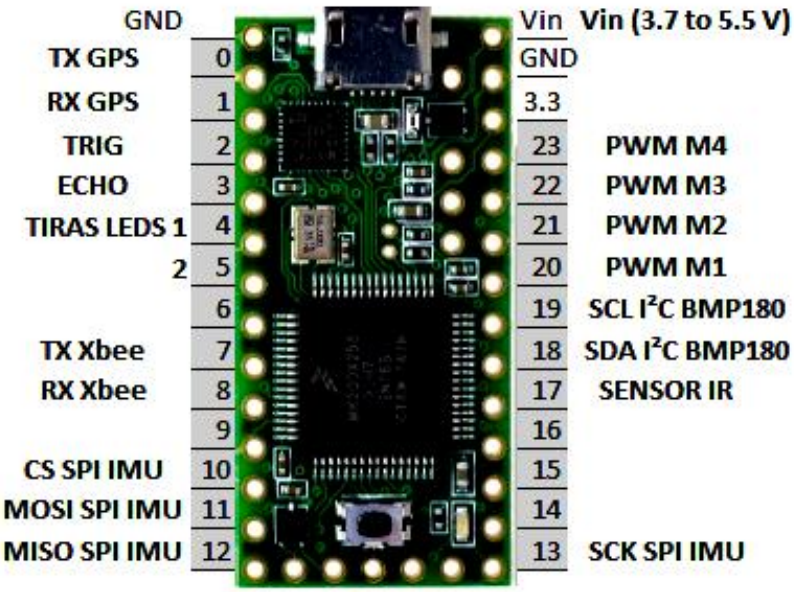


Figura 73.- Esquema de conexión de pines del Teensy 3.1 con los sensores utilizados.

En la Figura 74 se ilustra el código de configuración para el GPS utilizando el microcontrolador Teensy ++ 2.0, donde se incluye la biblioteca AltSoftSerial con pines específicos para Rx, Tx (4, 25).

```
#include<AltSoftSerial.h>
#include<TinyGPS.h>
AltSoftSerial gps_pines(4,25);
TinyGPS gps;
bool dato = false;
unsigned long chars;
unsigned short sentences, error;
float flat=0, flon=0;
void setup()  {
  Serial.begin(9600);
  gps_pines.begin(57600);
}
```

Figura 74.- Configuración de pines Tx y Rx para el Teensy ++ 2.0.

Para la lectura de la *IMU* se recurrió al análisis del *datasheet* (Figura 75) para determinar las direcciones de las salidas del giroscopio y del acelerómetro para cada uno de los ejes. Una vez teniendo la lectura final de la *IMU* se llevo a cabo el desarrolló una

biblioteca especial, la cual fue implementada en el código principal para el control PID. El código se ilustra en la Figura 76.

Name	R/W	Flash Backup	Address[1]	Default	Register Description
XGYRO_OUT	R	No	0x04	N/A	X-axis gyroscope output
YGYRO_OUT	R	No	0x06	N/A	Y-axis gyroscope output
ZGYRO_OUT	R	No	0x08	N/A	Z-axis gyroscope output
XACCL_OUT	R	No	0x0A	N/A	X-axis accelerometer output
YACCL_OUT	R	No	0x0C	N/A	Y-axis accelerometer output
ZACCL_OUT	R	No	0x0E	N/A	Z-axis accelerometer output

OUTPUT DATA REGISTERS

Each output data register uses the format in Figure 12 and Table 9. Figure 6 shows the positive direction for each inertial sensor. The ND bit is equal to 1 when the register contains unread data. The EA bit is high when any error/alarm flag in the DIAG_STAT register is equal to 1.

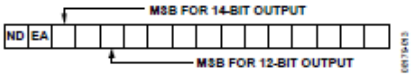


Table 9. Output Data Register Formats

Register	Bits	Scale	Reference
SUPPLY_OUT	12	2.418 mV	See Table 10
XGYRO_OUT[1]	14	0.05°/sec	See Table 11
YGYRO_OUT[1]	14	0.05°/sec	See Table 11
ZGYRO_OUT[1]	14	0.05°/sec	See Table 11
XACCL_OUT	14	0.333 mg	See Table 12
YACCL_OUT	14	0.333 mg	See Table 12
ZACCL_OUT	14	0.333 mg	See Table 12
XTEMP_OUT[2]	12	0.136°C	See Table 13
YTEMP_OUT[2]	12	0.136°C	See Table 13
ZTEMP_OUT[2]	12	0.136°C	See Table 13
AUX_ADC	12	805.8 µV	See Table 14

[1] Assumes that the scaling is set to ±300°/sec. This factor scales with the range.
[2] 0x0000 = 25°C (±5°C).

Table 10. Power Supply, Offset Binary Format

Supply Voltage	Decimal	Hex	Binary
5.25 V	2171 LSB	0x87B	XXXX 1000 0111 1011
5.002418 V	2069 LSB	0x815	XXXX 1000 0001 0101
5 V	2068 LSB	0x814	XXXX 1000 0001 0100
4.997582 V	2067 LSB	0x813	XXXX 1000 0001 0011
4.75 V	1964 LSB	0x7AC	XXXX 0111 1010 1100

Table 11. Rotation Rate, Twos Complement Format

Rotation Rate	Decimal	Hex	Binary
+300°/sec	+6000 LSB	0x1770	XX01 0111 0111 0000
+0.1°/sec	+2 LSB	0x0002	XX00 0000 0000 0010
+0.05°/sec	+1 LSB	0x0001	XX00 0000 0000 0001
0°/sec	0 LSB	0x0000	XX00 0000 0000 0000
−0.05°/sec	−1 LSB	0x3FFF	XX11 1111 1111 1111
−0.1°/sec	−2 LSB	0x3FFE	XX11 1111 1111 1110
−300°/sec	−6000 LSB	0x2890	XX10 1000 1001 0000

Table 12. Acceleration, Twos Complement Format

Acceleration	Decimal	Hex	Binary
+1.7 *g*	+5105 LSB	0x13F1	XX01 0011 1111 0001
+0.667 *mg*	+2 LSB	0x0002	XX00 0000 0000 0010
+0.333 *mg*	+1 LSB	0x0001	XX00 0000 0000 0001
0 *g*	0 LSB	0x0000	XX00 0000 0000 0000
−0.333 *mg*	−1 LSB	0x3FFF	XX11 1111 1111 1111
−0.667 *mg*	−2 LSB	0x3FFE	XX11 1111 1111 1110
−1.7 *g*	−5105 LSB	0x2C0F	XX10 1100 0000 1111

Table 13. Temperature, Twos Complement Format

Temperature	Decimal	Hex	Binary
+105°C	+588 LSB	0x24C	XXXX 0010 0100 1100
+85°C	+441 LSB	0x1B9	XXXX 0001 1011 1001
+25.272°C	+2 LSB	0x002	XXXX 0000 0000 0010
+25.136°C	+1 LSB	0x001	XXXX 0000 0000 0001
+25°C	0 LSB	0x000	XXXX 0000 0000 0000
+24.864°C	−1 LSB	0xFFF	XXXX 1111 1111 1111
+24.728°C	−2 LSB	0xFFE	XXXX 1111 1111 1110
−40°C	−478 LSB	0xE22	XXXX 1110 0010 0010

Table 14. Analog Input, Offset Binary Format

Input Voltage	Decimal	Hex	Binary
3.3 V	4095 LSB	0xFFF	XXXX 1111 1111 1111
1 V	1241 LSB	0x4D9	XXXX 0100 1101 1001
1.6116 mV	2 LSB	0x002	XXXX 0000 0000 0010
805.8 µV	1 LSB	0x001	XXXX 0000 0000 0001
0 V	0 LSB	0x000	XXXX 0000 0000 0000

Figura 75.- Datasheet ADIS16362

```
#include <SPI.h>
#include <math.h>

#define IMU_ADIS_GX   0x04
#define IMU_ADIS_GY   0x06
#define IMU_ADIS_GZ   0x08
#define IMU_ADIS_AX   0x0A
#define IMU_ADIS_AY   0x0C
#define IMU_ADIS_AZ   0x0E

typedef unsigned int uint;

class IMU_ADIS {
  // SPI
  uint cs;
  SPISettings config_imu;
  int lectura (byte cmd) {
    byte buf[] = {cmd, 0x00};
    enviarSPI(buf, 2);
    delay(10);

    SPI.beginTransaction (config_imu);
    digitalWrite(cs, LOW);
    byte b0 = SPI.transfer(cmd);
    byte b1 = SPI.transfer(cmd);
    digitalWrite(cs,HIGH);
    SPI.endTransaction();
    delay(10);
    b0 |= 0xC0;
    int zum = (((int)b0 << 8) | (int)b1);
    return zum;
  }

  void enviarSPI(byte buf[], int l)
  {
    SPI.beginTransaction (config_imu);
    digitalWrite(cs,LOW);
    for (int i = 0; i < l; i++)
    {
      SPI.transfer(buf[i]);
    }
    digitalWrite(cs,HIGH);
    SPI.endTransaction();
  }

  // Angulos
  int ang_x = 0;
  int ang_y = 0;
  int ang_z = 0;
  float conv_acel (int zum) {
    int temp = zum & 0x00002000;
    if (temp <= 0)
    return (zum & 0x00001FFF)*(0.333/1000);
    return zum*0.333/1000;
```

Figura 76.- Biblioteca creada para la lectura de la IMU ADIS16362.

En la segunda parte del código se realiza la conversión de la lectura de la *IMU* y se obtienen los ángulos para roll, pitch, yaw representado por angX, angY y angZ así como la implementación del filtro complementario (Figura 77).

```
float conv_gyro (int zum) {
  int temp = zum & 0x00002000;
  if (temp <= 0)
    return (zum & 0x00001FFF)*0.05;
  return zum*0.05;
}
float angX() {
  float accz = conv_acel(lectura(IMU_ADIS_AZ));
  float accy = conv_acel(lectura(IMU_ADIS_AY));
  return ((atan2(accy,accz))*180/PI);
}
float angY() {
  float accz = conv_acel(lectura(IMU_ADIS_AZ));
  float accx = conv_acel(lectura(IMU_ADIS_AX));
  return ((atan2(accx,accz))*180/PI);
}
float angZ() {
  float accy = conv_acel(lectura(IMU_ADIS_AY));
  float accx = conv_acel(lectura(IMU_ADIS_AX));
  return ((atan2(accy,accx))*180/PI);
}
// Filtro
int alfa = 0.99;
unsigned long t = 0, dt = 0;
void filtro (float ang_n, int &ang_v, float gyro, unsigned long
  ang_v = alfa*(ang_v + gyro*dt) + (1-alfa)*ang_n;
}

public:
/* Constructor
 * uint channel_selecct pin con el que se esclaviza a la IMU
 */
IMU_ADIS(uint channel_select) {
  cs = channel_select;
  config_imu = SPISettings(2000000, MSBFIRST, SPI_MODE3);
  SPI.begin();
  pinMode(cs, OUTPUT);
}
```

Figura 77.- Segunda parte del código para la lectura de datos de la IMU ADIS16362.

Finalmente se expone la tercera parte donde se obtienen las lecturas y se almacenan en un arreglo de longitud 3 denominado rpy[x] (Figura 78).

```
void getRPY(float *rpy) {
  t = millis();
  //angulo x
  float gyro_x = conv_gyro(lectura(IMU_ADIS_GX));
  float ang_xn = angX();
  filtro(ang_xn, ang_x, gyro_x, dt);

  //angulo y
  float gyro_y = conv_gyro(lectura(IMU_ADIS_GY));
  float ang_yn = angY();
  filtro(ang_yn, ang_y, gyro_y, dt);

  //angulo z
  float gyro_z = conv_gyro(lectura(IMU_ADIS_GZ));
  float ang_zn = angZ();
  filtro(ang_zn, ang_z, gyro_z, dt);

  dt = millis() - t;

  rpy[0] = ang_x;
  rpy[1] = ang_y;
  rpy[2] = ang_z;
}
```

Figura 78.- Tercera parte del código para la lectura de la IMU ADIS16362.

Fueron implementados dos tipos de PID para observar con cuál se obtiene mejor respuesta. El primer código a exponer fue desarrollado por medio de la literatura (Figura 79, 80, 81) y el segundo fue el proporcionado por la biblioteca de Arduino (Figura 82, 83).

```
//PID roll
float roll = 0.0;
float  kpr = 1.5, kir = 0.1, kdr = 0.3;
float errl = 0.0, erl = 0.0, der = 0.0, salr = 0.0, inr = 0.0, inr2 = 0.0;

//PID pitch
float pitch = 0.0;
float   kpp = 1.5, kip = 0.1, kdp = 0.3;
float  erpl = 0.0, erp = 0.0, dep = 0.0, salp = 0.0, inp = 0.0, inp2 = 0.0;

//PID yaw
float  yaw = 0.0;
float  kpy = 1.5, kiy = 0.1, kdy = 0.3;
float eryl = 0.0, ery = 0.0, dey = 0.0, saly = 0.0, iny = 0.0, iny2 = 0.0;

//variables
int p1 = 0, p2 = 0, p3 = 0, p4 = 0;
int p1eq = 0, p2eq = 0, p3eq = 0, p4eq = 0;
int alt = 100;
```

Figura 79.- Declaración de variables para el PID de cada ángulo.

```
void setup(){
  m1.attach(27);
  m2.attach(14);
  m3.attach(16);
  m4.attach(15);

  m1.write(0);
  m2.write(0);
  m3.write(0);
  m4.write(0);

  Serial1.begin(9600);
  Serial1.clear();

  SPI.begin();
  pinMode (cs, OUTPUT);
  pinMode (6, OUTPUT);

  digitalWrite (6, HIGH);
  delay (2500);
}

void loop()
{
  if (Serial1.available())  {
    dato = Serial1.read();
    if (dato == 'A')        {
        m1.writeMicroseconds(1000);
        m2.writeMicroseconds(1000);
        m3.writeMicroseconds(1000);
        m4.writeMicroseconds(1000);
        ban = 0;
        digitalWrite(6,LOW);
   if (dato == 'B') {
         ban = 1;
         digitalWrite(6,HIGH);
      }
   }// if
   if (ban == 1)  {
     imu.getRPY(rpy);
     //PID roll
     roll2 = rpy[0];
     if (rpy[0] > 0) roll2 -= 179;
     if (rpy[0] < 0) roll2 += 179;
      err = roll2;
      inr = (err + err1)/2;
     inr2 = inr2 + inr;
      der = (err - err1);
     err1 = err;
     salr = kpr*err + kir*inr2 + kdr*der;
```

Figura 80.- Conexión a motores e inicio de PID.

```
//PID pitch
pitch2 = rpy[1];
if (rpy[1] > 0) pitch2 -= 179;
if (rpy[1] < 0) pitch2 += 179;
 erp = pitch2;
 inp = (erp + erp1)/2;
inp2 = inp2 + inp;
 dep = (erp - erp1);
erp1 = erp;
salp = kpp*erp + kip*inp2 + kdp*dep;

//PID yaw
yaw = rpy[2];
if (rpy[2] > 0) yaw -= 179;
if (rpy[2] < 0) yaw += 179;
 ery = yaw;
 iny = (ery + ery1)/2;
iny2 = iny2 + iny;
 dey = (ery - ery1);
ery1 = ery;
saly = kpy*ery + kiy*iny2 + kdy*dey;

//PWM
p1 = constrain(alt + salr, -200,200);
p2 = constrain(alt + salp, -200,200);
p3 = constrain(alt - salr, -200,200);
p4 = constrain(alt - salp, -200,200);

p1eq = map(p1, -200, 200, 50, 166);
p2eq = map(p2, -200, 200, 47, 128);
p3eq = map(p3, -200, 200, 48, 169);
p4eq = map(p4, -200, 200, 92, 164);
m1.write(p1eq);
m2.write(p2eq);
m3.write(p3eq);
m4.write(p4eq);

if (err >= 30 || erp >= 30) {
m1.write(50);
m2.write(48);
m3.write(48);
m4.write(92);
}
```

Figura 81.- Salida PID a motores.

```
#include <SPI.h>
#include <IMU_ADIS.h>
#include <Servo.h>
#include <PID_v1.h>

double kp = 2.2, ki = 0.9 , kd = 1.3;
#define PID_ROLLMM 100

Servo m1, m3;
char dato;
int b = 0;
int altura = 1250;
double v1 = 0.0, v3 = 0.0;

IMU_ADIS imu(20);

float rpy[3] = {0.0f, 0.0f, 0.0f};
double sp_roll = 0.0;
double err = 0.0;
double salr = 0.0;

PID roll_cont(&err, &salr, &sp_roll, kp, ki, kd, REVERSE);

void setup() {
  Serial.begin(9600);
  Serial.clear();

  m1.attach(27);
  m3.attach(16);

  m1.write(0);
  m3.write(0);

  roll_cont.SetMode(AUTOMATIC);
  roll_cont.SetOutputLimits(-PID_ROLLMM, PID_ROLLMM);
}
 void loop() {
   if (Serial.available()){
   dato = Serial.read();
      if (dato == 'a'){
        Serial.println("a");
        m1.writeMicroseconds(1000);
        m3.writeMicroseconds(1000);
        b = 0;
      }

      if (dato == 'b'){
        Serial.println("b");
        delay(2000);
        m1.writeMicroseconds(1100);
        m3.writeMicroseconds(1100);
      }
```

Figura 82.- Implementación PID por medio del uso de biblioteca de Arduino.

```
    if (dato == 'c'){
      imu.getRPY(rpy);
      if (rpy[0] > 0) sp_roll=rpy[0] - 179;
      if (rpy[0] < 0) sp_roll=rpy[0] + 179;
      Serial.print(sp_roll);
      Serial.println("\tc");
      b = 1;
    }
    if (dato == 'd'){
      b = 0;
      Serial.println("d");
      m1.write(0);
      m3.write(0);
    }
  } //End if serial

  if (b == 1){
  imu.getRPY(rpy);
  err = rpy[0];
  if (rpy[0] > 0)  err -= sp_roll;
  if (rpy[0] < 0)  err += sp_roll;

  roll_cont.Compute();

  v1 = altura - salr;
  v3 = altura + salr;

   m1.writeMicroseconds(v1);
   m3.writeMicroseconds(v3);
   }
}
```

Figura 83.- Salida PID a motores.

REFERENCIAS BIBLIOGRÁFICAS

[1] **Quadricóptero Autónomo de Arquitectura Abierta QA3.** Redolfi Javier Andrés, Henze Agustín. Trabajo de Tesis. 14 de Marzo 2011.

[2] **Diseño e implementación de un sistema de control para un cuadricóptero.** Pico Villalpando, Antonio. Trabajo de tesis para obtener el grado de Maestro de Ciencias en Computación. Cinvestav. México. Diciembre 2012.

[3] **Modelado y control de un cuadricóptero.** Sevilla Fernández, Luis. ICAI Universidad Pontificia Comillas. Madrid. Junio 2014.

[4] **Model predictive quadrotor control: attitude, altitude and position experimental studies.** K. Alexis, et.al. IET Control Theory and Applications. Suecia. Marzo 2012.

[5] **Navegación 3D de un sistema de vuelo autónomo de tipo quadrotor.** Arellano Muro, Carlos Augusto. Trabajo de tesis para obtener el grado de Maestro en Ciencias. Cinvestav. México. Agosto 2014.

[6] **Sistema de control para la estabilidad y orientación de un helicóptero quadrotor.** Jaramillo Gómez Felipe, Gómez Yepes Álvaro. Trabajo de grado para el título de Ingeniero Mecatrónico. Escuela de Ingeniería de Antioquia. Colombia. 2013.

[7] **Quad-Rotor Control.** Eran Nissim Asaf Tal. Trabajo de tesis. Israel Institute of Technology. Agosto 2011.

[8] **Implementación de un helicóptero de dos grados de libertad con fines académicos.** Ramírez Villa Daniel, Henao Sánchez Santiago. Semillero de investigación en robótica móvil. Universidad de San Buenaventura Seccional Medellín. Colombia.

[9] **http://www.digi.com/products/xbee-rf-solutions/modules/xbee-pro-xsc**

[10] **Arduino PID – Guía de uso de la librería.** Traducción del trabajo de Brett Beauregard.http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

[11] **Quadcopter Final Report.** Adan Amarillas, et.al. Trabajo de tesis.